IPL (Indian Premier League) Tournament Scheduling through Backtrack Approach

Thesis submitted in partial fulfillment of the requirements for the award of degree of

Master of Engineering
in
Software Engineering

Submitted By
Trinetra Kumar Pathak
(Roll No. 801131029)

Under the supervision of:
Mr. Vinod K. Bhalla
Assistant Professor

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA – 147004

July 2013

CERTIFICATE

I hereby certify that the work which is being presented in the thesis entitled, **"IPL (Indian Premier League) Tournament Scheduling through Backtrack Approach"**, in partial fulfillment of the requirements for the award of degree of Master of Engineering in *Software Engineering* submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of *Mr. Vinod K. Bhalla* and refers other researcher's work which are duly listed in the reference section.

The matter presented in the thesis has not been submitted for award of any other degree of this or any other University.

Signature:
Trinetra Kumar Pathak

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

Mr. Vinod K. Bhalla
Assistant Professor
Computer Science and Engineering Department

Countersigned by

(Dr. Maninder Singh)
Head of Department
Computer Science and Engineering Department
Thapar University
Patiala

(Dr. S. K. Mohapatra)
Dean (Academic Affairs)
Thapar University
Patiala

ACKNOWLEDGMENT

No volume of words is enough to express my gratitude towards my guide, **Mr. Vinod K. Bhalla**, Assistant Professor, Computer Science and Engineering Department, Thapar University, who has been very concerned and has supervised the work presented in this thesis report. He has helped me to explore this vast field in an organized manner and provided me with all the ideas on how to work towards a research oriented venture.

I am also thankful to **Dr. Maninder Singh**, Head of Department, CSED and **Mr. Karun Verma**, P.G. Coordinator, for the motivation and inspiration that triggered me for the thesis work.

I would also like to thank the staff members and my colleagues who were always there in the need of the hour and provided with all the help and facilities, which I required, for the completion of my thesis.

Most importantly, I would like to thank my **parents**, **friends** and the **Almighty** for showing me the right direction out of the blue, to help me stay calm in the oddest of the times and keep moving even at times when there was no hope.

Trinetra Kumar Pathak

(801131029)

ABSTRACT

IPL (Indian Premier League) tournament scheduling is proving the many combinations of the IPL tournament which is useful for the IPL members to get the schedule according to them. In this schedule we have at least two matches gap between previous and next matches for each and every team. Though this approach we will give some relaxations time to each and every team and then each team perform well in each match.

IPL tournament schedules develop using backtracking approach which time complexity is define as an iterative backtracking, i.e. O (!n), where $n \geq 9$. Backtrack approach has been used on those combinations whose pairs are not rearrange simply according to our given constrains, means we have applied iterative backtracking approach on remains pairs. It must be n*(n-1) pairs after rearrangement. If rearrangement is not completed n*(n-1) pairs then we have used backtracking approach. We use iterative backtracking methods with our constraint.

This thesis comprise of 5 chapters, chapter 1 describes information retrieval, basic recursive and basic tournament techniques. Chapter 2 describes various tournament schedule and various techniques to develop a tournament with constraints. Chapter 3 specifies the problem statement, objective defined for this thesis and methodology to achieve these objectives. Chapter 4 describes the algorithm and its implementation with example. Chapter 5 concludes the overall thesis with overall observations and future work is highlighted.

Keywords: IPL, Iterative Backtracking, Schedule

Table of Contents

Certificate i
Acknowledgement ii
Abstract iii
Table of Contents iv
List of Figures vi
List of Tables viii

Chapter 1 Introduction 1
1.1 IPL Overview 1
1.2 Tournament Basics 1
1.3 Time Complexity 2
1.4 Backtracking Approach 3

Chapter 2 Literature Review 4
2.1 IPL Tournament Description 4
2.2 Iteration Method 4
2.3 Recursion Approach 5
2.4 Recursion Problem Solving 5
2.5 Thinking Recursively 6
2.6 Recursive Backtracking in General 6
2.7 Recursion vs. Iterative 6
2.8 Rearrangement Approach 7
2.9 Various Algorithms of Tournament 9

Chapter 3 Problem Statement and Objectives 13
3.1 Problem Statement 13
3.2 Objectives and Subtasks 13

Chapter 4 Propose Algorithm for IPL Tournament Schedule 14
4.1 Explanation of Algorithm 14
4.2 Illustration using Example 15

4.3 Demerit of Algorithm 22
4.4 Algorithm of IPL Tournament Schedule 26

Chapter 5 Experimental Results 30
5.1 Result for 9 teams 30
5.2 Results and Conclusion 50

Chapter 6 Conclusion and Future Scope 51
6.1 Conclusion 51
6.2 Thesis Contribution. 51
6.4 Future Scope 51

References 53
List of Publications 56

List of Figures

Figure 1.1 A 5-tournament Graph 1
Figure 2.1 Diagram of repeating the iterations 7
Figure 2.2 Graph of y=f(x) 8
Figure 2.3 Graph of y=f(x), y=x and y=g(x) 9
Figure 5.1 IPL Tournament Schedule Number 1 30
Figure 5.2 IPL Tournament Schedule Number 2 31
Figure 5.3 IPL Tournament Schedule Number 3 31
Figure 5.4 IPL Tournament Schedule Number 4 32
Figure 5.5 IPL Tournament Schedule Number 5 32
Figure 5.6 IPL Tournament Schedule Number 6 33
Figure 5.7 IPL Tournament Schedule Number 7 33
Figure 5.8 IPL Tournament Schedule Number 8 34
Figure 5.9 IPL Tournament Schedule Number 9 34
Figure 5.10 IPL Tournament Schedule Number 10 35
Figure 5.11 IPL Tournament Schedule Number 11 35
Figure 5.12 IPL Tournament Schedule Number 12 36
Figure 5.13 IPL Tournament Schedule Number 13 36
Figure 5.14 IPL Tournament Schedule Number 14 37
Figure 5.15 IPL Tournament Schedule Number 15 37
Figure 5.16 IPL Tournament Schedule Number 16 38
Figure 5.17 IPL Tournament Schedule Number 17 38
Figure 5.18 IPL Tournament Schedule Number 18 39
Figure 5.19 IPL Tournament Schedule Number 19 39
Figure 5.20 IPL Tournament Schedule Number 20 40
Figure 5.21 IPL Tournament Schedule Number 21 40
Figure 5.22 IPL Tournament Schedule Number 22 41
Figure 5.23 IPL Tournament Schedule Number 23 41
Figure 5.24 IPL Tournament Schedule Number 24 42
Figure 5.25 IPL Tournament Schedule Number 25 42
Figure 5.26 IPL Tournament Schedule Number 26 43
Figure 5.27 IPL Tournament Schedule Number 27 43

Figure 5.28 IPL Tournament Schedule Number 28 44
Figure 5.29 IPL Tournament Schedule Number 29 44
Figure 5.30 IPL Tournament Schedule Number 30 45
Figure 5.31 IPL Tournament Schedule Number 31 45
Figure 5.32 IPL Tournament Schedule Number 32 46
Figure 5.33 IPL Tournament Schedule Number 33 46
Figure 5.34 IPL Tournament Schedule Number 34 47
Figure 5.35 IPL Tournament Schedule Number 35 47
Figure 5.36 IPL Tournament Schedule Number 36 48
Figure 5.37 IPL Tournament Schedule Number 37 48
Figure 5.38 IPL Tournament Schedule Number 38 49
Figure 5.39 IPL Tournament Schedule Number 39 49

List of Tables

Table 1.1 Common asymptotic functions 2
Table 2.1 Values of g(x) on some x 9
Table 4.1 Simple schedule of IPL for n=9 15
Table 4.2 IPL Tournament Schedule after insertion of pair (1, 2) and (3, 4) 17
Table 4.3 Simple combination of pair after deleting pairs (1, 2) and (3, 4) 18
Table 4.4 Simple combination of left pairs after simple rearrangement 19
Table 4.5 IPL Tournament Schedule after rearrangement of all pair except pairs (8, 6) and (9, 8) 20
Table 4.6 IPL Tournament Schedule after inserting remains pair (8, 6) on position 22^{nd} 21
Table 4.7 One IPL Tournament Schedule of combination of pairs after inserting last remains pair (9, 6) on position 25^{th} 22
Table 4.8 Simple combination of pairs with starting pair (1, 3) 23
Table 4.9 IPL Tournament Schedule after rearrangement of pairs except some pairs 24
Table 4.10 Simple combination of left pairs after simple rearrangement 25

Chapter 1
Introduction

This chapter introduces tournament, its properties and various developing technologies like iterative backtracking and recursive backtracking.

1.1 IPL Overview

IPL (Indian Premier League) tournament is the league of the cricket of India. The 2013 season of the Indian Premier League was called as IPL 6 or IPL 2013, and these seasons of the Indian Premier League, established by the Board of Control for Cricket in India (BCCI) in 2007. The Premier League is the shortest type of cricket. A T20 match only has 20 overs per inning. Indian and international players take part in IPL and contributing there what is the world's "richest cricket tournament". IPL tournament is basically a cricket tournament and in this tournament all teams play exact two matches with all others teams one on the home town stadium and second on opponent home town. Till now there are six IPL tournaments have been done. But we have no any such schedule in which at least two matches gap between previous and next matches for each and every team.

1.2 Tournament Basics

A tournament T is a complete asymmetric digraph [1]. Where every two distinct u, v $\in$ V (T), either (u, v) $\in$ A (T) or (v, u) $\in$ A (T), but not both, and (u, u) $\notin$ A (T) for all u $\in$ V (T). A tournament with v vertices is called a v-tournament. Equivalently, some define an n- tournament as an orientation of the complete graph K_n.

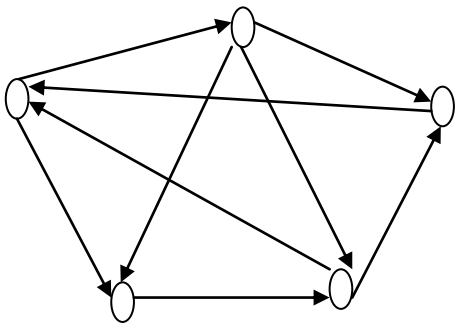


Figure 1.1: A 5-tournament graph

The above Figure 1.1 show A 5-tournament. In studying tournament structure, we often want to look at sub tournaments and the dual of a tournament. Given a tournament T, and $S \subseteq V(T)$, a sub tournament of T is the tournament T[S], with $V(T[S]) = S$, and $(u, v) \in A(T[S])$ if and only if $u, v \in S$ and $(u, v) \in A(T)$. That is, T[S] is the sub digraph induced on S. Also, for a vertex $v \in V(T)$ or a set of vertices S, we define the sub tournaments T -v and T -S to be the tournaments T [V (T)-{v}] and T [V (T)-S] respectively. The dual of T is the tournament T^r, with $V(T^r) = V(T)$ and $(u, v) \in A(T^r)$ if and only if $(v, u) \in A(T)$. We define the dual of a digraph in the same way, and use the same notation.

1.3 Time Complexity

Step count is to compare time complexity of two programs which help to compute same function and also to predict the growth in run time as instance characteristics changes. Determining exact step count is difficult and not necessary also because the values are not exact quantities. We need only comparative statements like c_1n For example, consider two programs with complexities c_1n small values of n, complexity depend upon values of c_1, c_2 and c_3. But there will also be an n beyond which complexity of c_3n is better than that of c_1n break-even point. If this point is zero then c_3n will always faster (or at least as fast as or faster than all others). Common asymptotic functions are given below.

Table1.1: Common asymptotic functions

Sr. No.	Function	Name
1	1	Constant
2	log n	Logarithmic
3	N	Linear
4	n log n	n log n
5	n^2	Quadratic
6	n^3	Cubic
7	2^n	Exponential
8	n!	Factorial

1.4 Backtracking Approach

A backtracking algorithm tries to build a solution to a computational problem incrementally. Whenever the algorithm needs to decide between two alternatives to the next component of the solution, it simply tries both options recursively [2], [3], [4].

It is a systematic search strategy of the state-space of combinatorial problems. It is mainly used to solve problems which ask for finding elements of a set which satisfy some restrictions. Many problems which can be solved by backtracking have the following general form: "Find S subset of A_1 x A_2 x … x A_n (A_k – finite sets) such that each element s= (s_1, s_2, …, s_n) satisfy some restrictions".

Let's consider a more complicated problem, called SUBSETSUM [4]: Given a set Y of positive integers and target integer P, is there a subset of elements in Y that add up to P? Notice that there can be more than one such subset. For example, if Y = {8, 6, 7, 5, 3, 10, 9} and P = 15, the answer is TRUE, thanks to the subsets {8, 7} or {7, 5, 3} or {6, 9} or {5, 10}. On the other hand, if Y = {11, 6, 5, 1, 7, 13, 12} and P = 15, the answer is FALSE. There are two trivial cases. If the target value T is zero, then we can immediately return TRUE [4], [5], because the elements of the empty set add up to zero. On the other hand, if P < 0, or if P = 0 but the set Y is empty, then we can immediately return FALSE.

Chapter 2

Literature Review

This chapter discusses the various tournament introduced and importance of these tournaments with some applications.

2.1 IPL tournament Description

IPL (Indian Premier League) is the league of the Indian cricket. And many others countries are also follows these type of tournament such as National Cricket League Twenty 20 (often abbreviated as NCL T20), was a Twenty20 cricket competition initiated by the Bangladesh Cricket Board (BCB). It includes 6 teams named by the divisions of Bangladesh. It has now been replaced by BPL. And in Sri Lanka its name is Sri Lanka Premier League (SLPL) is the premier Twenty20 cricket competition in Sri Lanka, held by Sri Lanka Cricket. Its first season was held in 2012 and it replaced the Inter-Provincial Twenty20.

The 2013 season of the Indian Premier League, abbreviated as IPL 6 or IPL 2013, is the sixth season of the Indian Premier League, established by the Board of Control for Cricket in India in 2007. The Premier League is generally considered to be the world's showcase for Twenty20 (T20) cricket, which is the shortest type of cricket. A T20 match only has 20 overs per inning. Top Indian and international players take part in IPL, contributing to what is the world's "richest cricket tournament". IPL tournament is basically a cricket tournament and in this tournament all teams play exact two matches with all others teams one on the home town stadium and second on opponent home town. Till now there are six IPL tournaments have been done. But we have no any such schedule in which at least two matches gap between previous and next matches for each and every team.

2.2 Iteration Method

• When we encounter a problem that requires repetition, we often use iteration [4], [6], i.e., some type of loop.

• Sample problem: printing the series of integers from c1 to c2, where c1 <= c2.

Example: print_Series(4, 10) should print the following:

4, 5, 6, 7, 8, 9, 10

Here's an iterative solution to this problem:

Algorithm of Print_Series

Print_Series(c1, c2)

Where c1 and c2 are two integer numbers

1. Set i:=c1
2. Repeat while i < c2
   a. Write i
   b. Set i:=i+1
3. Write c2
4. Return

In this algorithm we use loop that indicates the iterative method. And that iterative method future use in backtracking as iterative backtracking algorithm.

2.3 Recursion Approach

- An alternative approach to problems that require repetition is to solve them using recursion [1], [6].
- A recursive method is a method that calls itself.
- Applying this approach to the Print_Series problem gives:

- Algorithm of Print_Series using recursive backtracking.

Print_Series (c1, c2)

Where c1 and c2 are two integer numbers

1. If c1 = c2 then
   Write c2
2. Otherwise
   a. Write c1
   b. Call Print_Series(c1+1, c2)
3. Return

2.4 Recursive Problem Solving

- When we use recursion, we solve a problem by reducing it to a simpler problem of the same kind [5], [7], [8].

- We keep doing this until we reach a problem that is simple enough to be solved directly.

This simplest problem is known as the base case.

2.5 Thinking Recursively

• When solving a problem using recursion, ask own self these questions:

1. How can we break this problem down into one or more smaller subproblems?

• make recursive method calls to solve the subproblems [8], [9]

2. What are the base cases?

• i.e., which subproblems are small enough to solve directly?

3. Do I need to combine the solutions to the subproblems?

If so, how should we do so?

2.6 Recursive Backtracking in General

• Useful for constraint satisfaction problems that involve assigning values to variables according to a set of constraints.

• N-Queens:

    • variables = Queen's position in each row.

    • constraints = no two queens in same row, column, diagonal.

• Map coloring

    • variables = each state's color.

    • constraints = no two bordering states with the same color [5], [10].

• Many others: tournament scheduling, factory scheduling, room scheduling, etc.

• Backtracking reduces the # of possible value assignments that we consider, because it never considers invalid assignments [7], [10], [11].

• Using recursion allows us to easily handle an arbitrary number of variables.

• stores the state of each variable in a separate stack frame

2.7 Recursion vs. Iteration

- Recursive methods can often be easily converted to a non-recursive method that uses iteration.
- This is especially true for methods in which there is only one recursive call and it comes at the end (tail) of the method.

- Once you're comfortable with using recursion, you'll find that some algorithms are easier to implement using recursion.
- We'll also see that some data structures lend themselves to recursive algorithms [4], [7].
- Recursion is a bit more costly because of the overhead involved in invoking a method.
- Rule of thumb, if it's easier to formulate a solution recursively, use recursion, unless the cost of doing so is too high otherwise, use iteration.

2.8 Rearrangement Approach

This is an alternative method that rearranges the original question into two equations, a straight line and a curve, and then finds where these meet [12], [13], [14]. It works as follows:

1. Rearrange the equation to be solved into the form x = g(x). [g(x) is a different function].
2. The solution to this equation is to find where y = x meets y = g(x). This is where the coordinates on g(x) are the same.
3. In this situation, we can guess a value x_0 and hope that $g(x_0)$ will be a better guess. This can be iterated repeatedly.

Below is a diagram which shows that by starting at a value of x_0 and repeating the iterations (i.e. finding x_1, x_2,...), you get closer and closer to the point where the two lines cross.

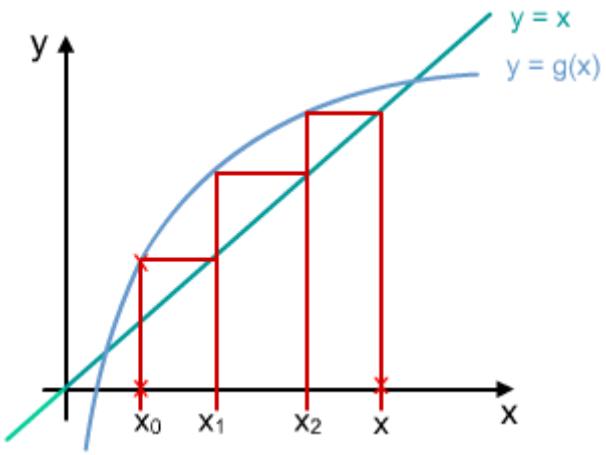


Figure 2.1: Diagram of repeating the iterations.

Performing this iteration produces two possible results:

1. It diverges (i.e. it gets further and further away from the start). This means the rearrangement has not worked [5], [14], [15], [16].
2. It converges (i.e. it homes in) to the root, and solves the equation.

Rearrangement Method

Let $f(x) = e^x - 3x^3 + 2x^2 - 1.25x - 1.1$

The graph of $y = f(x)$ is shown below:

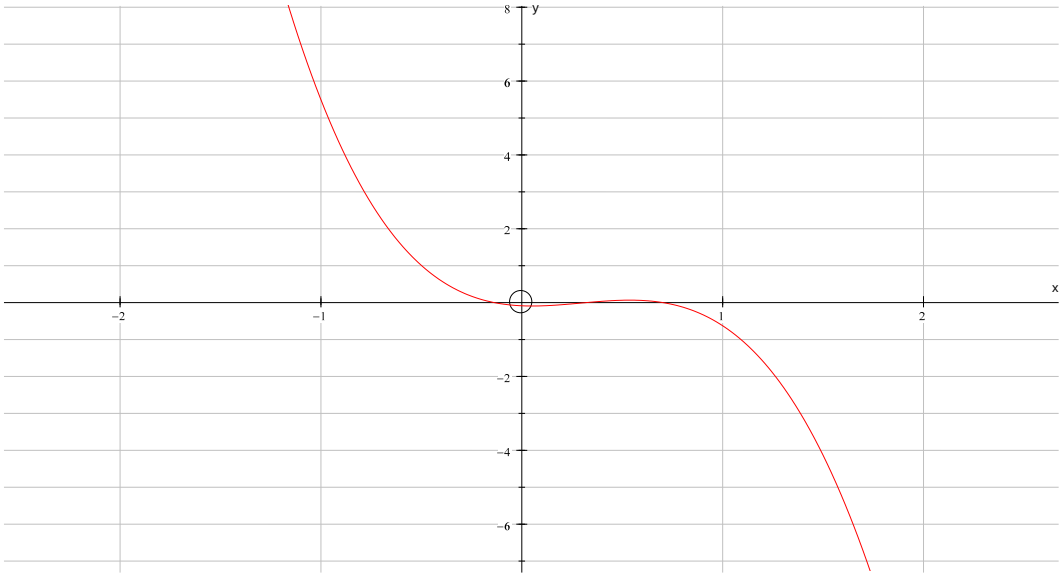


Figure 2.2: Graph of y=f(x)

Rearranging $f(x) = 0$ for x to give iteration may give a single point after converging. Let the right-hand side of this equation be $g(x)$, given that the left-hand side is x.

We are then able to find the roots of the equation $f(x) = 0$ if we are able to find the corresponding roots of the equation $x = g(x)$.

Rearranging $e^x - 3x^3 + 2x^2 - 1.25x - 1.1 = 0$, we get $x = e^x - 3x^3 + 2x^2 - 0.25x - 1.1$

Let $g(x) = e^x - 3x^3 + 2x^2 - 0.25x - 1.1$

Firstly, we draw a graph of $y = f(x)$, $y = x$ and $y = g(x)$.

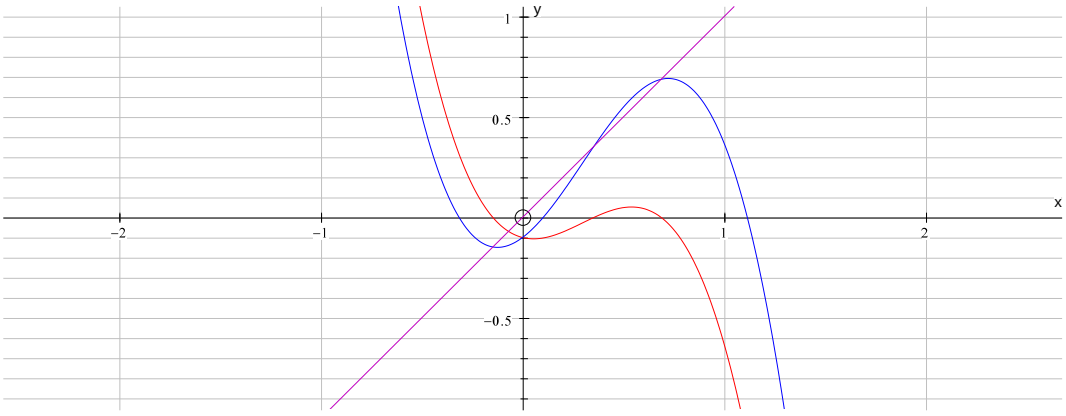


Figure 2.3: Graph of y=f(x), y=x and y=g(x)

Table 2.1: Values of g(x) on some x

X	**g(x)**	**X**	**g(x)**
0.50000000000	0.54872127070	0.68726557965	0.68726622930
0.54872127070	0.60039607388	0.68726622930	0.68726638283
0.60039607388	0.64440838290	0.68726638283	0.68726641912
0.64440838290	0.67148672177	0.68726641912	0.68726642769
0.67148672177	0.68275320038	0.68726642769	0.68726642972
0.68275320038	0.68613509470	0.68726642972	0.68726643020
0.68613509470	0.68699498967	0.68726643020	0.68726643031
0.68699498967	0.68720204775	0.68726643031	0.68726643034
0.68720204775	0.68725120207	0.68726643031	0.68726643034
0.68725120207	0.68726283082	0.68726643034	0.68726643034
0.68726283082	0.68726557965		

The iteration used to find the roots of the equation $f(x)=0$ is summarised by $x_{n+1}=g(x_n)$, or, in actual terms, $x_{n+1}=e^{x_n}-3x_n^3+2x_n^2-0.25x_n-1.1$.

2.9 Various Algorithms of Tournament

A Hamilton path tournament design involving n teams and n/2 stadiums, is a round robin schedule on n − 1 days in which each team plays in each stadium at most twice [11],[12], [13], and the set of games played in each stadium induce a Hamilton path

on n teams. Previously, Hamilton path tournament designs were shown to exist for all even n not divisible by 4, 6, or 10. Here, we give an inductive procedure for the construction of Hamilton path tournament designs for $n = 2p \geq 8$ teams [11].

Studies on methods of scheduling sports tournaments are quite numerous [13], [17], [18], [19]. In particular, the problem of creating a round robin schedule involving n teams and n/2 stadiums (n: even) is a classical problem in the field of combinatorial design.
A balanced tournament design for n teams is a round robin schedule on $n - 1$ days in which
– n/2 games, one in each stadium are played on each day,
– each team uses each stadium at most twice.

Recently, a schedule closely related to balanced tournament designs has been considered by several authors [19], [20], [21], [22]. As in balanced tournament designs, we have n teams and n/2 stadiums, but in this case the goal is to create a schedule on n days in which each team pair plays at least once, each team uses each stadium exactly twice, and no two teams meet twice in the same stadium. For convenience, we will call such schedules stadium-balanced. Stadium-balanced schedules have been constructed for all even $n \geq 8$ (powers of 2 in [14], all other even n in [19]). Note that each team has exactly one opponent which it meets twice.

The ubiquity of combinatorial optimization problems in our society is illustrated by the novel application areas for optimization technology, which range from supply chain management to sports tournament scheduling. Over the last two decades, constraint programming has emerged as a fundamental methodology to solve a variety of combinatorial problems, and rich constraint programming languages have been developed for expressing and combining constraints and specifying search procedures at a high level of abstraction. Local search approaches to combinatorial optimization are able to isolate optimal or near-optimal solutions within reasonable time constraints [21]. This book introduces a method for solving combinatorial optimization problems that combines constraint programming and local search, using constraints to describe and control local search, and a programming language, COMET, that supports both modelling and search abstractions in the spirit of constraint programming. After an

overview of local search including neighbour hoods, heuristics, and metaheuristics, the book presents the architecture and modelling and search components of constraint-based local search and describes how constraint-based local search is supported in COMET [21], [22], [23]. The book describes a variety of applications, arranged by meta-heuristics. It presents scheduling applications, along with the background necessary to understand these challenging problems. The book also includes a number of satisfiability problems, illustrating the ability of constraint-based local search approaches to cope with both satisfiability and optimization problems in a uniform fashion. [20], [24], [25].

There is a method for the highly constrained problem of finding a seasonal schedule for the best Danish soccer league. The league differs from most sports leagues, since it plays a triple round robin tournament which leads to an uneven distribution of home and away games. The solution method presented here uses a logic-based Benders decomposition in which the master problem finds home-away pattern sets while the sub problem finds timetables [26], [27], [28], [29]. Furthermore, column generation techniques are used to enhance the speed of the master problem. The computational results show that the solution method is capable of solving the problem within reasonable time and the Danish Football Association has used it for scheduling the 2006/2007 season [30], [31].

Nemhauser and Trick [32] presented the problem of finding a timetable for the 1997/1998 Atlantic Coast Conference (ACC) in basketball. Their solution, found with a combination of integer programming and exhaustive enumeration, was accepted by the ACC.

Finite-domain constraint programming is another programming technique that can be used for solving combinatorial search problems such as sports tournament scheduling. This paper presents a solution of round-robin tournament planning based on finite-domain constraint programming [32]. The approach yields a dramatic performance improvement, which makes an integrated interactive software solution feasible [23], [34], [35].

Tournament design is of crucial importance in competitive sports. The primary goal of effective tournament design is to provide incentives for the participants to maximize their performance both during the tournament and in the time period

leading up to the tournament. In spectator sports, a secondary goal of tournament design is to also promote interesting match ups that generate fan interest. Seeded tournaments, in general, promote both goals. Teams or individuals with strong performances leading up to a tournament receive higher seeds which increase their chances of progressing further in the tournament. Furthermore, seeding ensures that the strongest teams or players are most likely to meet in the final rounds of the tournament when fan interest is at its peak. Under some distributions of team or player skill, however, a seeding system can introduce anomalies that could affect incentives. Our analysis of the NCAA men's basketball tournament uncovers such an anomaly [33], [34]. The seeding system in this tournament gives teams with better success in the regular season more favourable first round match ups, but the tournament is not reseeded as the games progress. Therefore, while higher seeds progress to the 2nd round of the tournament at uniformly higher rates than lower seeds, this relationship breaks down in later rounds [34],[35]. We find that 10^{th} and 11th seeds average more wins and typically progress farther in the tournament than 8^{th} and 9th seeds. This finding violates the intended incentive structure of seeded tournaments.

Chapter 3

Problem Statement and Objectives

In previous chapter we have to discus about the various tournaments scheduling using different methods. But there is not any schedule for IPL (Indian Premier League) tournament with the constraints.

3.1 Problem Statement

In IPL there are many constraints such that every team play with others twice, one on owns home town and another on opponent home town. And our problem is define one most important constraint, i.e. sufficient gap between previous and next matches for all teams. So we are introducing an algorithm which satisfy all previous constraints and also provide the at least two matches gap between previous and next matches for all teams and this is our constraint on IPL (Indian Premier League) tournament scheduling. There are 6 previous schedules of IPL but no one satisfy this constraint, so we have design an schedule who follows all previous and our constraint.

3.2 Objectives and Subtasks

Our main objective is providing sufficient gap of matches to each and every team to perform better in tournament. And providing equal chance to all team for relaxes. As we know IPL (Indian Premier League) tournament if very fast tournament which is also called IPL20T, it means i.e. 40 overs game, 20 overs each. So, we have design an algorithm which provides many schedules with constraint of at least two matches gap between previous and next matches for each team.

Chapter 4

Proposed Algorithm for IPL Tournament Scheduling

This chapter discuss the algorithm and its explanation with some examples. And also define the implementation of the algorithm.

4.1 Explanation of Algorithm

First, we have making a simple combination of pairs in which we have n teams, n stadium and each team played two matches with all other teams i.e. n*(n-1) matches played in tournament and also n*(n-1) pairs for one combination of this tournament. Now, we will start to rearrange the simple pairs of combination according to constraint. And our constraint is at least two matches different between previous and next match for each and every team. If team t1 played 5^{th} match then in 6^{th} and 7^{th} matches t1 will not allowed to play and same procedure for all teams. We start from first pair and then choose the second according to given constraint and same for all others. And delete the selected pairs from the simple pairs of combination and if there is any pair left in the simple pairs of combination which is/are not chosen from the simple pairs of combination then we will use backtracking algorithm to put the left pairs of the simple combination. In backtracking algorithm we have to choose pairs from pairs of simple combination one by one and start to put it into IPL combination with same constraint from bottom to top checking. And if any pair is not feet in IPL combination then we will rearrange the pairs of simple combination, we will use one left shift operation (first pair replace with second, second pair replace with third and so on for all others, and at last place first pair will come to complete the n*(n-1) pairs), and this pairs of combination is called pairs of simple combination then we will restart our algorithm to make the IPL combination again and this whole process will applied for until we will not got the n*(n-1) pairs in IPL combination or until we will reaches n*(n-1) one left shift operations in pairs of simple combination.

Now this procedure applies for all the combinations of pairs and then we select the best Combination of pairs generated by Hamilton path tournament.

4.2 Illustration Using Example

Table 4.1 indicate the pairs of simple combination. In which we have n*(n-1) pairs and each team play to all other exact two matches. This is a simple combination of pairs for the 9 teams, it means we have considered n = 9.

Table 4.1: Simple schedule of IPL for n=9

Serial No. of matches	**Pairs**	**Serial No. of matches**	**Pairs**	**Serial No. of matches**	**Pairs**	**Serial No. of matches**	**Pairs**
1.	1,2	21.	3,6	41.	6,1	61.	8,5
2.	1,3	22.	3,7	42.	6,2	62.	8,6
3.	1,4	23.	3,8	43.	6,3	63.	8,7
4.	1,5	24.	3,9	44.	6,4	64.	8,9
5.	1,6	25.	4,1	45.	6,5	65.	9,1
6.	1,7	26.	4,2	46.	6,7	66.	9,2
7.	1,8	27.	4,3	47.	6,8	67.	9,3
8.	1,9	28.	4,5	48.	6,9	68.	9,4
9.	2,1	29.	4,6	49.	7,1	69.	9,5
10.	2,3	30.	4,7	50.	7,2	70.	9,6
11.	2,4	31.	4,8	51.	7,3	71.	9,7
12.	2,5	32.	4,9	52.	7,4	72.	9,8
13.	2,6	33.	5,1	53.	7,5		
.14.	2,7	34	5,2	54.	7,6		
15.	2,8	35.	5,3	55.	7,8		
16.	2,9	36.	5,4	56.	7,9		
17.	3,1	37.	5,6	57.	8,1		
18.	3,2	38.	5,7	58.	8,2		
19.	3,4	39	5,8	59.	8,3		
20.	3,5	40.	5,9	60.	8,4		

Every match is played in a stadium and first element of the pair indicate the stadium, such as suppose first pair is (1, 2), it means team t1 and team t2 play with each other on the t1 home town stadium. First element of each pair is not only indicating the team number but also indicate about the stadium. Through which we can satisfy the

constraints of IPL (Indian Premier League) tournament, i.e. each team play with all other exact two matches, one match on owns home town stadium and another on opponent home town stadium.

Table 4.1 show the simple combination of the pairs which is not satisfying our constraint, which is gap constraint. So, we will rearrange the above schedule according to our constraint of gap i.e. at least two matches gap between previous and next matches played by each and every team.

First of all, we start to select one by one pairs and put on another table with our restriction. Such as, select pair (1, 2) and put in to another table and remove from the table4.1. Then select another i.e. (1, 3), but team t1 has played in last match of table4.2, so we can't put it into table4.2, so we will consider next pair from the table4.1. As we are seeing pair (1, 3), (1, 4)... (2, 9), (3, 1) and (3, 2) is not satisfy our constraint, so we will select next pair i.e. (3, 4) and this pair satisfy the constraint show in table4.2.

Now, we will continue to choose such pairs, and if we will reach on the last then again start from the starting off the table until we are unable to select any one from the remaining pairs.
There may be opportunity that there is no any pairs left in the simple combination of pairs schedule, that means we have a successful schedule of the IPL according to constraints. And there may be possibility that there are some pairs left in the simple combination of pairs schedule, i.e. show in table4.4.

Table 4.2: IPL Tournament Schedule after insertion of pair (1, 2) and (3, 4)

Serial No. of matches	**Pairs**	**Serial No. of matches**	**Pairs**	**Serial No. of matches**	**Pairs**	**Serial No. of matches**	**Pairs**
1.	1,2	21.		41.		61.	
2.	3,4	22.		42.		62.	
3.		23.		43.		63.	
4.		24.		44.		64.	
5.		25.		45.		65.	
6.		26.		46.		66.	
7.		27.		47.		67.	
8.		28.		48.		68.	
9.		29.		49.		69.	
10.		30.		50.		70.	
11.		31.		51.		71.	
12.		32.		52.		72.	
13.		33.		53.			
14.		34		54.			
15.		35.		55.			
16.		36.		56.			
17.		37.		57.			
18.		38.		58.			
19.		39		59.			
20.		40.		60.			

Table 4.3: Simple combination of pair after deleting pairs (1, 2) and (3, 4)

Serial No. of matches	**Pairs**	**Serial No. of matches**	**Pairs**	**Serial No. of matches**	**Pairs**	**Serial No. of matches**	**Pairs**
1.	~~**1,2**~~	21.	3,6	41.	6,1	61.	8,5
2.	1,3	22.	3,7	42.	6,2	62.	8,6
3.	1,4	23.	3,8	43.	6,3	63.	8,7
4.	1,5	24.	3,9	44.	6,4	64.	8,9
5.	1,6	25.	4,1	45.	6,5	65.	9,1
6.	1,7	26.	4,2	46.	6,7	66.	9,2
7.	1,8	27.	4,3	47.	6,8	67.	9,3
8.	1,9	28.	4,5	48.	6,9	68.	9,4
9.	2,1	29.	4,6	49.	7,1	69.	9,5
10.	2,3	30.	4,7	50.	7,2	70.	9,6
11.	2,4	31.	4,8	51.	7,3	71.	9,7
12.	2,5	32.	4,9	52.	7,4	72.	9,8
13.	2,6	33.	5,1	53.	7,5		
14.	2,7	34	5,2	54.	7,6		
15.	2,8	35.	5,3	55.	7,8		
16.	2,9	36.	5,4	56.	7,9		
17.	3,1	37.	5,6	57.	8,1		
18.	3,2	38.	5,7	58.	8,2		
19.	~~**3,4**~~	39	5,8	59.	8,3		
20.	3,5	40.	5,9	60.	8,4		

Table 4.4: Simple combination of left pairs after simple rearrangement

Serial No. of matches	**Pairs**	**Serial No. of matches**	**Pairs**	**Serial No. of matches**	**Pairs**	**Serial No. of matches**	**Pairs**
1.	~~1,2~~	21.	~~3,6~~	41.	~~6,1~~	61.	~~8,5~~
2.	~~1,3~~	22.	~~3,7~~	42.	~~6,2~~	62.	**8,6**
3.	~~1,4~~	23.	~~3,8~~	43.	~~6,3~~	63.	~~8,7~~
4.	~~1,5~~	24.	~~3,9~~	44.	~~6,4~~	64.	~~8,9~~
5.	~~1,6~~	25.	~~4,1~~	45.	~~6,5~~	65.	~~9,1~~
6.	~~1,7~~	26.	~~4,2~~	46.	~~6,7~~	66.	~~9,2~~
7.	~~1,8~~	27.	~~4,3~~	47.	~~6,8~~	67.	~~9,3~~
8.	~~1,9~~	28.	~~4,5~~	48.	~~6,9~~	68.	~~9,4~~
9.	~~2,1~~	29.	~~4,6~~	49.	~~7,1~~	69.	~~9,5~~
10.	~~2,3~~	30.	~~4,7~~	50.	~~7,2~~	70.	**9,6**
11.	~~2,4~~	31.	~~4,8~~	51.	~~7,3~~	71.	~~9,7~~
12.	~~2,5~~	32.	~~4,9~~	52.	~~7,4~~	72.	~~9,8~~
13.	~~2,6~~	33.	~~5,1~~	53.	~~7,5~~		
14.	~~2,7~~	34	~~5,2~~	54.	~~7,6~~		
15.	~~2,8~~	35.	~~5,3~~	55.	~~7,8~~		
16.	~~2,9~~	36.	~~5,4~~	56.	~~7,9~~		
17.	~~3,1~~	37.	~~5,6~~	57.	~~8,1~~		
18.	~~3,2~~	38.	~~5,7~~	58.	~~8,2~~		
19.	~~3,4~~	39	~~5,8~~	59.	~~8,3~~		
20.	~~3,5~~	40.	~~5,9~~	60.	~~8,4~~		

And the table 4.5 indicates the possible schedule of 70 matches, there is 2 pairs are not satisfying our constraints. So for remaining pairs we will use iterative backtracking method to find the appropriate schedule for IPL.

Table 4.5: IPL Tournament Schedule after rearrangement of all pair except pairs (8, 6) and (9, 8)

Serial No. of matches	**Pairs**	**Serial No. of matches**	**Pairs**	**Serial No. of matches**	**Pairs**	**Serial No. of matches**	**Pairs**
1.	1,2	21.	2,4	41.	2,7	61.	8,7
2.	3,4	22.	3,7	42.	3,8	62.	1,9
3.	5,6	23.	5,1	43.	4,9	63.	3,2
4.	7,1	24.	6,2	44.	5,7	64.	4,8
5.	8,2	25.	7,3	45.	6,8	65.	6,5
6.	9,3	26.	8,4	46.	9,4	66.	9,7
7.	1,4	27.	9,5	47.	1,7	67.	3,1
8.	2,5	28.	1,3	48.	2,8	68.	4,2
9.	3,6	29.	2,6	49.	3,9	69.	6,9
10.	4,1	30.	4,5	50.	5,4	70.	7,5
11.	5,2	31.	7,8	51.	6,7	71.	
12.	6,3	32.	9,6	52.	8,9	72.	
13.	7,4	33.	1,5	53.	2,1		
14.	8,1	34.	2,3	54.	4,3		
15.	9,2	35.	4,7	55.	5,9		
16.	3,5	36.	5,8	56.	7,6		
17.	4,6	37.	6,1	57.	1,8		
18.	7,2	38.	7,9	58	2,9		
19.	8,3	39.	8,5	59.	5,3		
20.	9,1	40.	1,6	60.	6,4		

Here we have left two pairs, i.e. (8, 6) and 9, 8) and to insert these pairs in IPL tournament schedule we have used iterative backtracking algorithm on all remain pairs, i.e. after inserting pair (8, 6) we have indicated in table 4.6.Table 4.6 indicate all pairs of tournament except pair (9, 8).

Table 4.6: IPL Tournament Schedule after inserting remains pair (8, 6) on position 22nd

Serial No. of matches	**Pairs**	**Serial No. of matches**	**Pairs**	**Serial No. of matches**	**Pairs**	**Serial No. of matches**	**Pairs**
1.	1,2	21.	2,4	41.	1,6	61.	6,4
2.	3,4	22.	**8,6**	42.	2,7	62.	8,7
3.	5,6	23.	3,7	43.	3,8	63.	1,9
4.	7,1	24.	5,1	44.	4,9	64.	3,2
5.	8,2	25.	6,2	45.	5,7	65.	4,8
6.	9,3	26.	7,3	46.	6,8	66.	6,5
7.	1,4	27.	8,4	47.	9,4	67.	9,7
8.	2,5	28.	9,5	48.	1,7	68.	3,1
9.	3,6	29.	1,3	49.	2,8	69.	4,2
10.	4,1	30.	2,6	50.	3,9	70.	6,9
11.	5,2	31.	4,5	51.	5,4	71.	7,5
12.	6,3	32.	7,8	52.	6,7	72.	
13.	7,4	33.	9,6	53.	8,9		
14.	8,1	34	1,5	54.	2,1		
15.	9,2	35.	2,3	55.	4,3		
16.	3,5	36.	4,7	56.	5,9		
17.	4,6	37.	5,8	57.	7,6		
18.	7,2	38.	6,1	58.	1,8		
19.	8,3	39	7,9	59.	2,9		
20.	9,1	40.	8,5	60.	5,3		

There is only one pair is left in simple tournament schedule to insert in the IPL tournament schedule. So we have applied backtracking on another remain pair, i.e. pair (9, 8). And table 4.7 indicate the one schedule of combination of the pairs of the IPL tournament after inserting the pair (9, 6).

Table 4.7: One IPL Tournament Schedule of combination of pairs after inserting last remains pair (9, 6) on position 25th

Serial No. of matches	**Pairs**	**Serial No. of matches**	**Pairs**	**Serial No. of matches**	**Pairs**	**Serial No. of matches**	**Pairs**
1.	1,2	21.	2,4	41.	8,5	61.	5,3
2.	3,4	22.	8,6	42.	1,6	62.	6,4
3.	5,6	23.	3,7	43.	2,7	63.	8,7
4.	7,1	24.	5,1	44.	3,8	64.	1,9
5.	8,2	25.	**9,8**	45.	4,9	65.	3,2
6.	9,3	26.	6,2	46.	5,7	66.	4,8
7.	1,4	27.	7,3	47.	6,8	67.	6,5
8.	2,5	28.	8,4	48.	9,4	68.	9,7
9.	3,6	29.	9,5	49.	1,7	69.	3,1
10.	4,1	30.	1,3	50.	2,8	70.	4,2
11.	5,2	31.	2,6	51.	3,9	71.	6,9
12.	6,3	32.	4,5	52.	5,4	72.	7,5
13.	7,4	33.	7,8	53.	6,7		
14.	8,1	34	9,6	54.	8,9		
15.	9,2	35.	1,5	55.	2,1		
16.	3,5	36.	2,3	56.	4,3		
17.	4,6	37.	4,7	57.	5,9		
18.	7,2	38.	5,8	58.	7,6		
19.	8,3	39	6,1	59.	1,8		
20.	9,1	40.	7,9	60.	2,9		

4.3 Demerit of Algorithm

Each simple combination of pairs is not necessary to produce IPL tournament schedule. There may be some arrangements those are not producing any schedule. If this type of simple combination is founded then the algorithm drops the schedule and applied backtracking approach on simple combination of pairs for the next simple combination of pairs.

Table 4.8: Simple combination of pairs with starting pair (1, 3).

Serial No. of matches	**Pairs**	**Serial No. of matches**	**Pairs**	**Serial No. of matches**	**Pairs**	**Serial No. of matches**	**Pairs**
1.	1,3	21.	3,7	41.	6,2	61.	8,6
2.	1,4	22.	3,8	42.	6,3	62.	8,7
3.	1,5	23.	3,9	43.	6,4	63.	8,9
4.	1,6	24.	4,1	44.	6,5	64.	9,1
5.	1,7	25.	4,2	45.	6,7	65.	9,2
6.	1,8	26.	4,3	46.	6,8	66.	9,3
7.	1,9	27.	4,5	47.	6,9	67.	9,4
8.	2,1	28.	4,6	48.	7,1	68.	9,5
9.	2,3	29.	4,7	49.	7,2	69.	9,6
10.	2,4	30.	4,8	50.	7,3	70.	9,7
11.	2,5	31.	4,9	51.	7,4	71.	9,8
12.	2,6	32.	5,1	52.	7,5	72.	1,2
13.	2,7	33.	5,2	53.	7,6		
.14.	2,8	34	5,3	54.	7,8		
15.	2,9	35.	5,4	55.	7,9		
16.	3,1	36.	5,6	56.	8,1		
17.	3,2	37.	5,7	57.	8,2		
18.	3,4	38.	5,8	58.	8,3		
19.	3,5	39	5,9	59.	8,4		
20.	3,6	40.	6,1	60.	8,5		

Table 4.8 shows such type of simple pairs generated during the backtracking on the simple combination of pairs. This table indicate the simple combination of pairs after producing the successful IPL schedule with starting pair (1, 2), and after that it shift all pairs up with just one position and first pair shift to last position in combination. It means after shifting the starting pair of the simple combination is (1, 3) and the last pair is (1, 2). This type of shifting is done for all pairs once. Table 4.8 indicate such simple combination of pairs whose IPL tournament scheduling is not possible with given constraints.

Table 4.9 shows all arrangement of table 4.8 without applying backtrack approach. There are some pairs, i.e. pairs (1, 7), (2, 9), (3, 8), (5, 9), (6, 4), (8, 7) and (9, 7) are not inserted showing in table 4.10, because of given constraints. So we applied backtracking on all remaining pairs one by one. And remaining pair (1, 7) is not fulfil given constraints and not got any position in IPL tournament schedule. So, the algorithm drops this simple combination of pairs and move for the next one using the sifting down to up with one shift.

Table 4.9: IPL Tournament Schedule after rearrangement of pairs except some pairs.

Serial No. of matches	**Pairs**	**Serial No. of matches**	**Pairs**	**Serial No. of matches**	**Pairs**	**Serial No. of matches**	**Pairs**
1.	1,3	21.	9,1	41.	2,1	61.	9,8
2.	2,4	22.	2,6	42.	4,3	62.	4,2
3.	5,6	23.	3,5	43.	5,7	63.	5,3
4.	7,1	24.	4,7	44.	6,9	64.	7,9
5.	8,2	25.	6,1	45.	1,8	65.	8,6
6.	9,3	26.	8,5	46.	2,7	66.	
7.	1,4	27.	9,4	47.	3,9	67.	
8.	2,5	28.	1,2	48.	4,6	68.	
9.	3,6	29.	3,7	49.	5,8	69.	
10.	4,1	30.	4,5	50.	7,3	70.	
11.	5,2	31.	6,8	51.	9,6	71.	
12.	6,3	32.	7,2	52.	2,8	72.	
13.	7,4	33.	8,4	53.	3,1		
14.	8,1	34	9,5	54.	4,9		
15.	9,2	35.	1,6	55.	6,5		
16.	1,5	36.	2,3	56.	7,8		
17.	3,4	37.	4,8	57.	1,9		
18.	6,2	38.	5,1	58.	3,2		
19.	7,5	39	6,7	59.	5,4		
20.	8,3	40.	8,9	60.	7,6		

This is the main demerit of this algorithm and to solve that we can modify in this algorithm and then we can solve these type of problem. Actually we can get number of combination may be n * (n-1) IPL tournament schedule. And we are working on it to improve this algorithm to get all the possible combination of the IPL tournament schedule.

Table 4.10: Simple combination of left pairs after simple rearrangement

Serial No. of matches	**Pairs**	**Serial No. of matches**	**Pairs**	**Serial No. of matches**	**Pairs**	**Serial No. of matches**	**Pairs**
1.	1,3	21.	3,7	41.	6,2	61.	8,6
2.	1,4	22.	3,8	42.	6,3	62.	8,7
3.	1,5	23.	3,9	43.	6,4	63.	8,9
4.	1,6	24.	4,1	44.	6,5	64.	9,1
5.	1,7	25.	4,2	45.	6,7	65.	9,2
6.	1,8	26.	4,3	46.	6,8	66.	9,3
7.	1,9	27.	4,5	47.	6,9	67.	9,4
8.	2,1	28.	4,6	48.	7,1	68.	9,5
9.	2,3	29.	4,7	49.	7,2	69.	9,6
10.	2,4	30.	4,8	50.	7,3	70.	9,7
11.	2,5	31.	4,9	51.	7,4	71.	9,8
12.	2,6	32.	5,1	52.	7,5	72.	1,2
13.	2,7	33.	5,2	53.	7,6		
.14.	2,8	34	5,3	54.	7,8		
15.	2,9	35.	5,4	55.	7,9		
16.	3,1	36.	5,6	56.	8,1		
17.	3,2	37.	5,7	57.	8,2		
18.	3,4	38.	5,8	58.	8,3		
19.	3,5	39	5,9	59.	8,4		
20.	3,6	40.	6,1	60.	8,5		

4.4 Algorithm of IPL Tournament Schedule

IPL_tournament_Schedule (N)

N is the number of teams, e[100][2] is two dimensional array to store the simple combination of pairs, d[100][2] is two dimensional array to store the IPL_Scheduling.

1. Set k:=1 , no:=1;
2. Set i:=1;
3. Repeat while i <= 9
   a. Set j:=1
   b. Repeat while j<=9
      i. If i!=j then do
         Set e[k][0]:=i
         Set e[k][1]:=j
         Set k:=k+1
      ii. Set j:=j+1
   c. Set i:=i+1
4. Set count:=1
5. Repeat while count!= N*(N-1)

(A) Set m:=1

6. Do until m!= N*(N-1)+1
   a. Set m:=1
   b. If count = N*(N-1) then go to the step 7
   c. Set o:= N*(N-1), k:=1
   d. Repeat while k <= o
      i. Set c[k][0]:=e[k][0]
      ii. Set c[k][1]:=e[k][1]
      iii. Set d[k][0]:=0
      iv. Set d[k][1]:=0
      v. Set k:=k+1
   e. Set p:=1
   f. Repeat while p!=0
      i. Set p:=0
      ii. Set k:=1

iii. Repeat while k<=o

A. If m=1 or (m=2 and c[k][0]!=d[m-1][0] and c[k][0]!=d[m-1][1] and c[k][1]!=d[m-1][0] and c[k][1]!=d[m-1][1]) or (c[k][0]!=d[m-1][0] and c[k][0]!=d[m-1][1] and c[k][1]!=d[m-1][0] and c[k][1]!=d[m-1][1] and c[k][0]!=d[m-2][0] and c[k][0]!=d[m-2][1] and c[k][1]!=d[m-2][0] and c[k][1]!=d[m-2][1]) then do

Set d[m][0]:=c[k][0]

Set d[m][1]:=c[k][1]

Set m:=m+1

Set p:=1, i=k

B. Repeat while i < o

I. Set c[i][0]:=c[i+1][0]

II. Set c[i][1]:=c[i+1][1]

III. Set i:=i+1

C. Set o:=o-1

D. Set k:=k-1

E. Set k:=k+1

g. If o!=0 then do

[1] Set i:=1

[2] Repeat while i<=o

i. Set n:=m

ii. Set d[m][0]:=d[m-1][0]

iii. Set d[m][1]:=d[m-1][1]

iv. Set d[m-1][0]:=d[m-2][0]

v. Set d[m-1][1]:=d[m-2][1]

vi. Set m:=m-2

vii. Repeat while m>=1

A. If m=1 and c[i][0]!=d[m+1][0] && c[i][0]!=d[m+1][1] && c[i][1]!=d[m+1][0] && c[i][1]!=d[m+1][1] && c[i][0]!=d[m+2][0] && c[i][0]!=d[m+2][1] && c[i][1]!=d[m+2][0] && c[i][1]!=d[m+2][1]) || (m==2 && c[i][0]!=d[m-1][0] && c[i][0]!=d[m-1][1] && c[i][1]!=d[m-1][0] && c[i][1]!=d[m-1][1] && c[i][0]!=d[m+1][0] && c[i][0]!=d[m+1][1] && c[i][1]!=d[m+1][0] &&

c[i][1]!=d[m+1][1] && c[i][0]!=d[m+2][0] && c[i][0]!=d[m+2][1] && c[i][1]!=d[m+2][0] && c[i][1]!=d[m+2][1]) || (c[i][0]!=d[m-1][0] && c[i][0]!=d[m-1][1] && c[i][1]!=d[m-1][0] && c[i][1]!=d[m-1][1] && c[i][0]!=d[m+1][0] && c[i][0]!=d[m+1][1] && c[i][1]!=d[m+1][0] && c[i][1]!=d[m+1][1] && c[i][0]!=d[m+2][0] && c[i][0]!=d[m+2][1] && c[i][1]!=d[m+2][0] && c[i][1]!=d[m+2][1] && c[i][0]!=d[m-2][0] && c[i][0]!=d[m-2][1] && c[i][1]!=d[m-2][0] && c[i][1]!=d[m-2][1]) then

Set d[m][0]:=c[i][0]

Set d[m][1]:=c[i][1]

Set m:=n+1

And go to step viii otherwise do

Set d[m][0]:=d[m-1][0]

Set d[m][1]:=d[m-1][1]

Set m:=m-1

viii. If m=0 then do

Set t1:=e[1][0]

Set t2:=e[1][1]

Set k:=1

Repeat while k< N*(N-1)

A. Set e[k][0]:=e[k+1][0]

B. Set e[k][1]:=e[k+1][1]

Set e[k][0]:=t1

Set e[k][1]:=t2

Set count:=count+1

And go to step h

ix. Set i:=i+1

[3] If i=o+1 then

Write all pairs

Otherwise

[1] Write all pairs

(B) Set t1:=e[1][0]

(C) Set t2:=e[1][1]

(D) Set k:=1

(E) Repeat while k < N*(N-1)

a. Set e[k][0]:=e[k+1][0]

b. Set e[k][1]:=e[k+1][1]

c. Set k:=k+1

(F) Set e[k][0]:=t1

(G) Set e[k][1]:=t2

(H) Set count:=count+1

7. Return

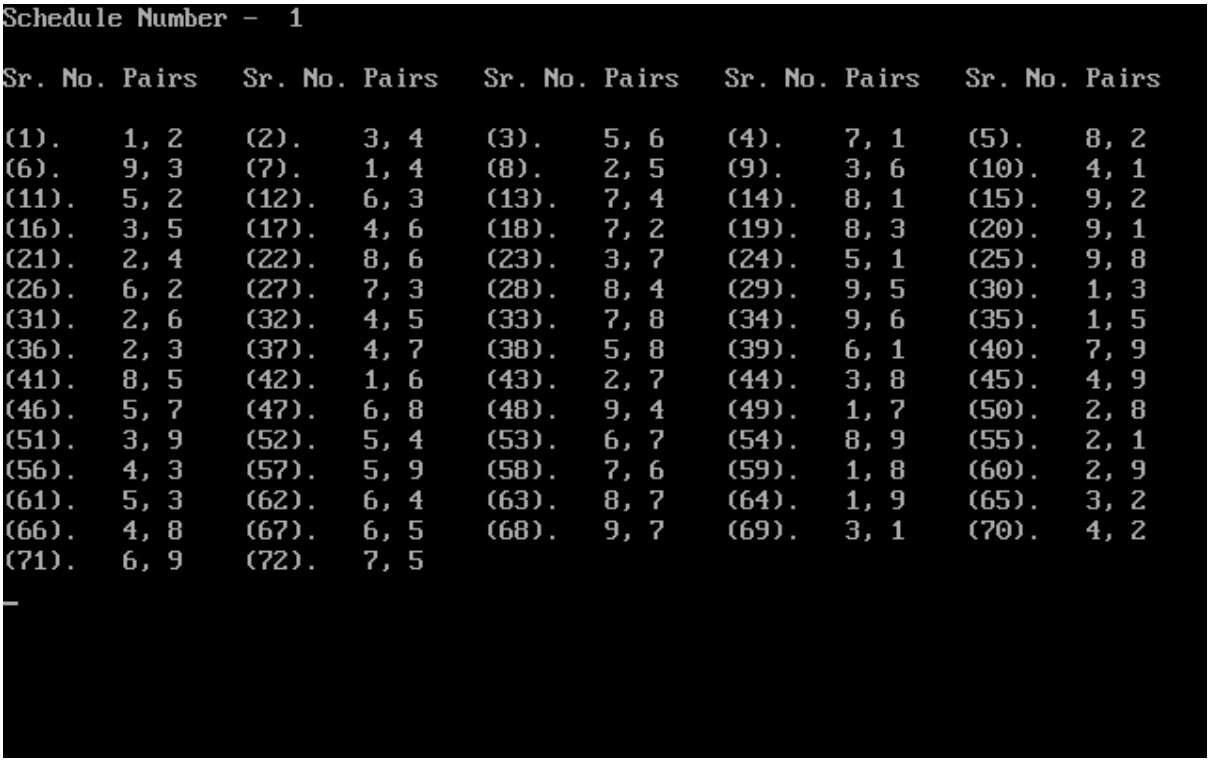
Chapter 5

Experimental Results

This chapter focuses on result for IPL (Indian Premier League) tournament schedules. There are numbers of schedules for IPL tournament.

5.1 Result for 9 teams

Here n is the number of teams participated in IPL tournament and n=9 means total 9 team is participating in the IPL (Indian Premier League) tournament. We have run our algorithm on Turbo C compiler and check out it's result, i.e. showing result is according to our constraints or not. And we also are keeping in view the previous constraints of the IPL tournament.



```
Schedule Number -  1

Sr. No. Pairs   Sr. No. Pairs   Sr. No. Pairs   Sr. No. Pairs   Sr. No. Pairs

(1).    1, 2    (2).    3, 4    (3).    5, 6    (4).    7, 1    (5).    8, 2
(6).    9, 3    (7).    1, 4    (8).    2, 5    (9).    3, 6    (10).   4, 1
(11).   5, 2    (12).   6, 3    (13).   7, 4    (14).   8, 1    (15).   9, 2
(16).   3, 5    (17).   4, 6    (18).   7, 2    (19).   8, 3    (20).   9, 1
(21).   2, 4    (22).   8, 6    (23).   3, 7    (24).   5, 1    (25).   9, 8
(26).   6, 2    (27).   7, 3    (28).   8, 4    (29).   9, 5    (30).   1, 3
(31).   2, 6    (32).   4, 5    (33).   7, 8    (34).   9, 6    (35).   1, 5
(36).   2, 3    (37).   4, 7    (38).   5, 8    (39).   6, 1    (40).   7, 9
(41).   8, 5    (42).   1, 6    (43).   2, 7    (44).   3, 8    (45).   4, 9
(46).   5, 7    (47).   6, 8    (48).   9, 4    (49).   1, 7    (50).   2, 8
(51).   3, 9    (52).   5, 4    (53).   6, 7    (54).   8, 9    (55).   2, 1
(56).   4, 3    (57).   5, 9    (58).   7, 6    (59).   1, 8    (60).   2, 9
(61).   5, 3    (62).   6, 4    (63).   8, 7    (64).   1, 9    (65).   3, 2
(66).   4, 8    (67).   6, 5    (68).   9, 7    (69).   3, 1    (70).   4, 2
(71).   6, 9    (72).   7, 5
_
```

Figure 5.1: IPL Tournament Schedule Number 1

This result show the first schedule of the IPL tournament and it start with pair (1, 2).

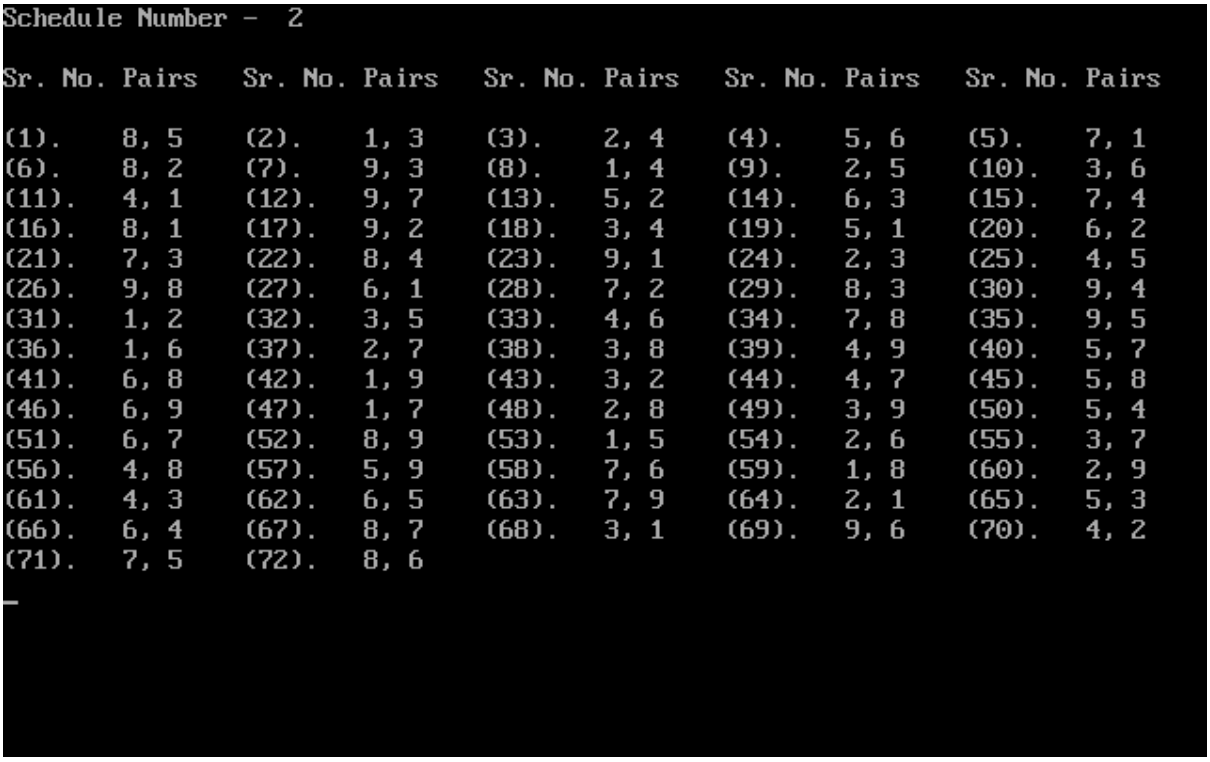


Figure 5.2: IPL Tournament Schedule Number 2

This result show the second schedule of the IPL tournament and it start with pair (8, 5).

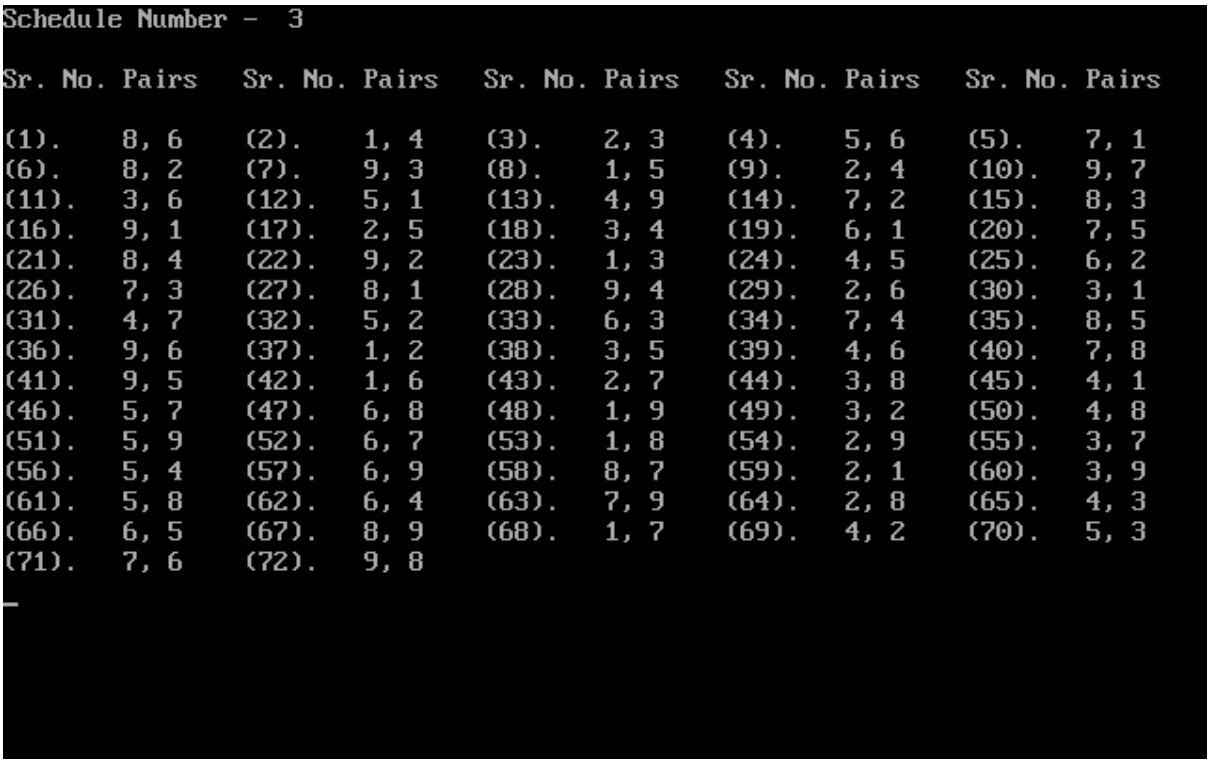


Figure 5.3: IPL Tournament Schedule Number 3

This result show the third schedule of the IPL tournament and it start with pair (8, 6).

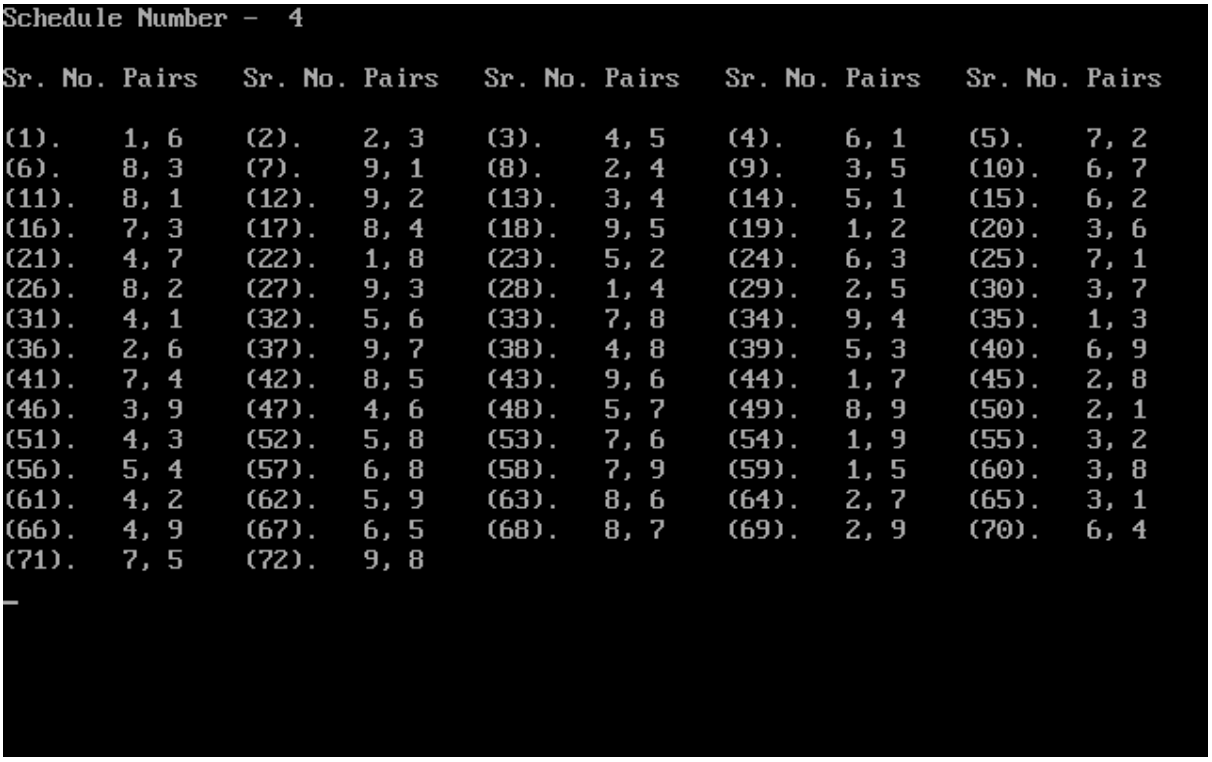


Figure 5.4: IPL Tournament Schedule Number 4

This result show the fourth schedule of the IPL tournament and it start with pair (1, 6).

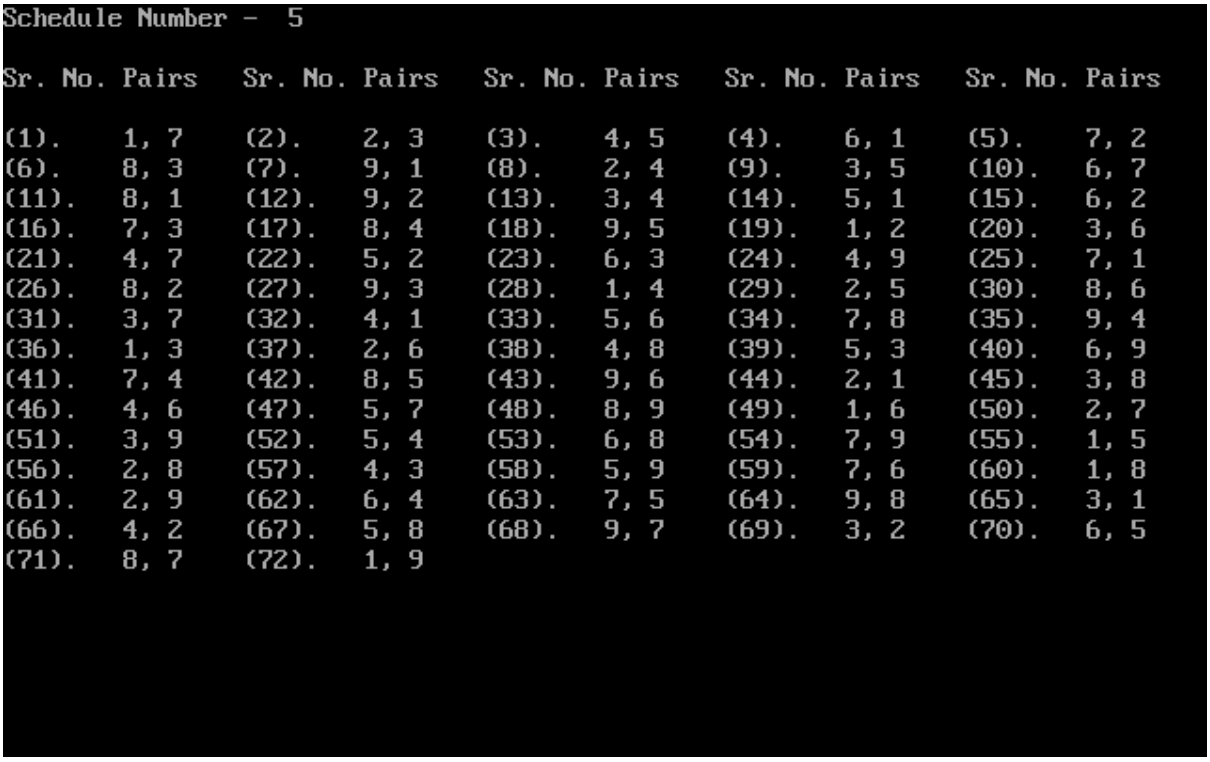


Figure 5.5: IPL Tournament Schedule Number 5

This result show the fifth schedule of the IPL tournament and it start with pair (1, 7).

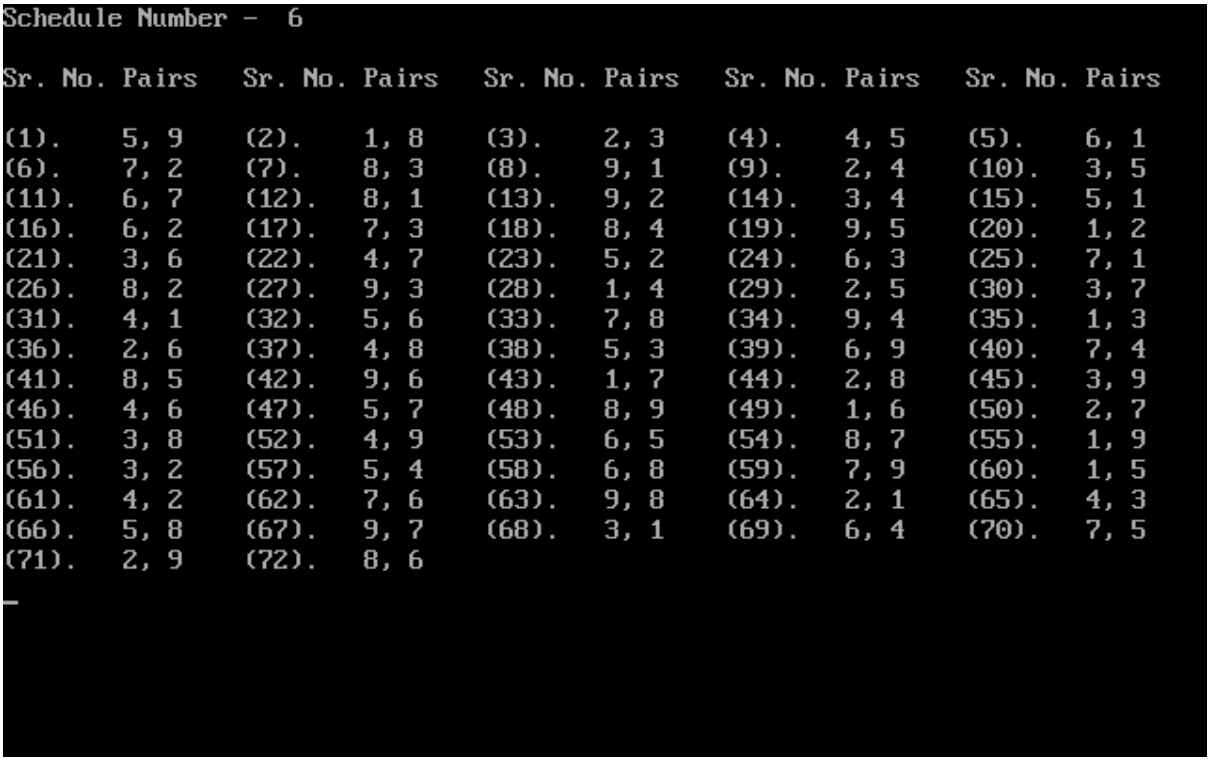


Figure 5.6: IPL Tournament Schedule Number 6

This result show the sixth schedule of the IPL tournament and it start with pair (5, 9).

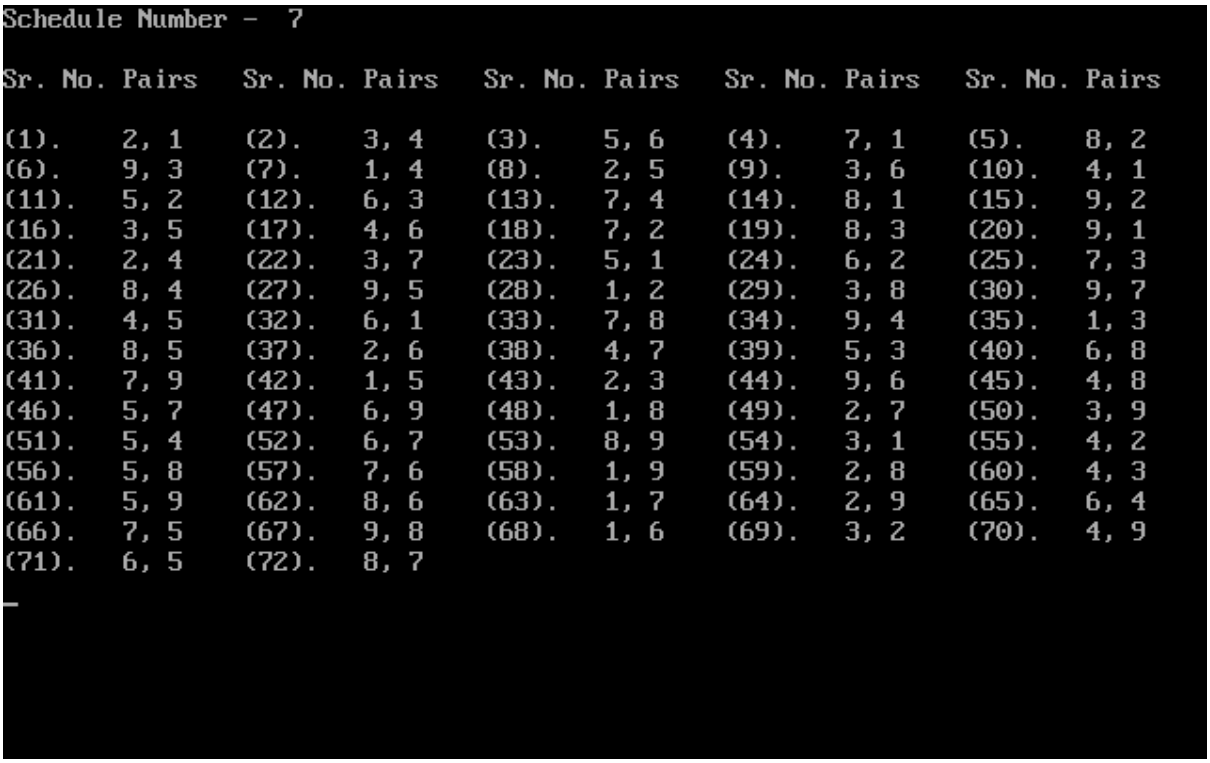


Figure 5.7: IPL Tournament Schedule Number 7

This result show the seventh schedule of the IPL tournament and it start with pair (2, 1).

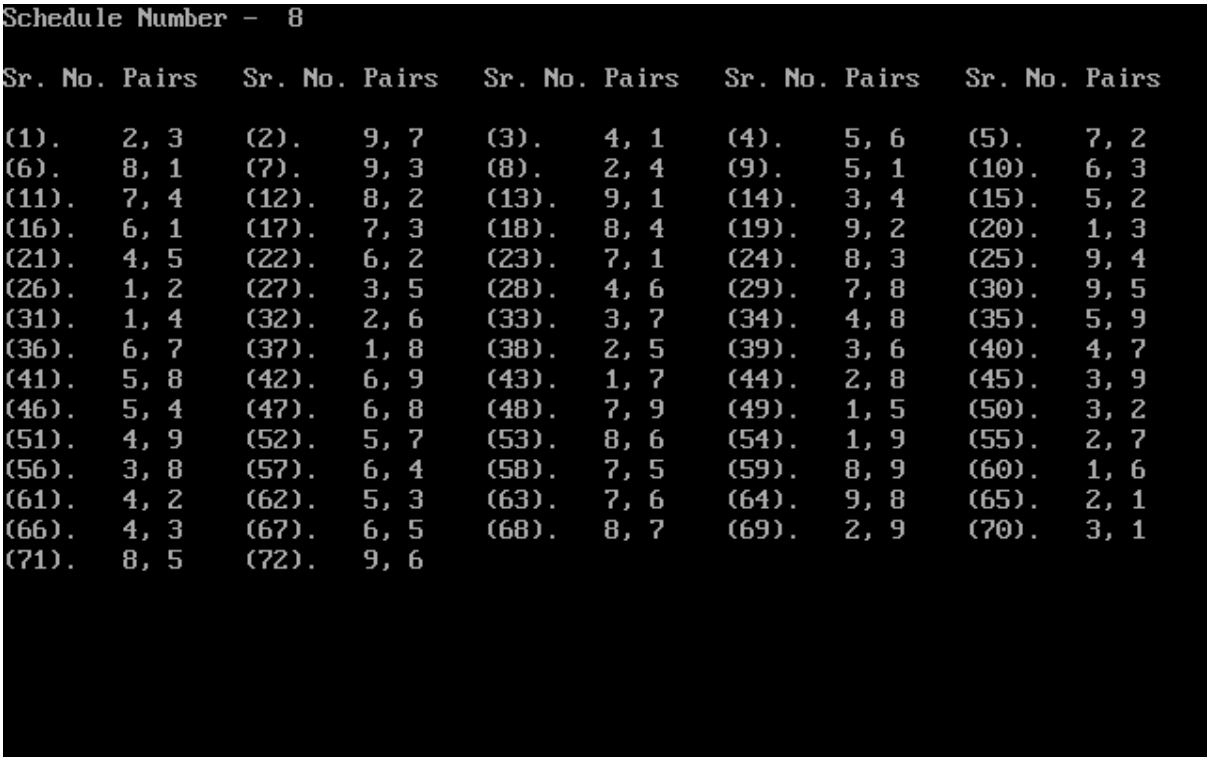


Figure 5.8: IPL Tournament Schedule Number 8

This result show the eighth schedule of the IPL tournament and it start with pair (2, 3).

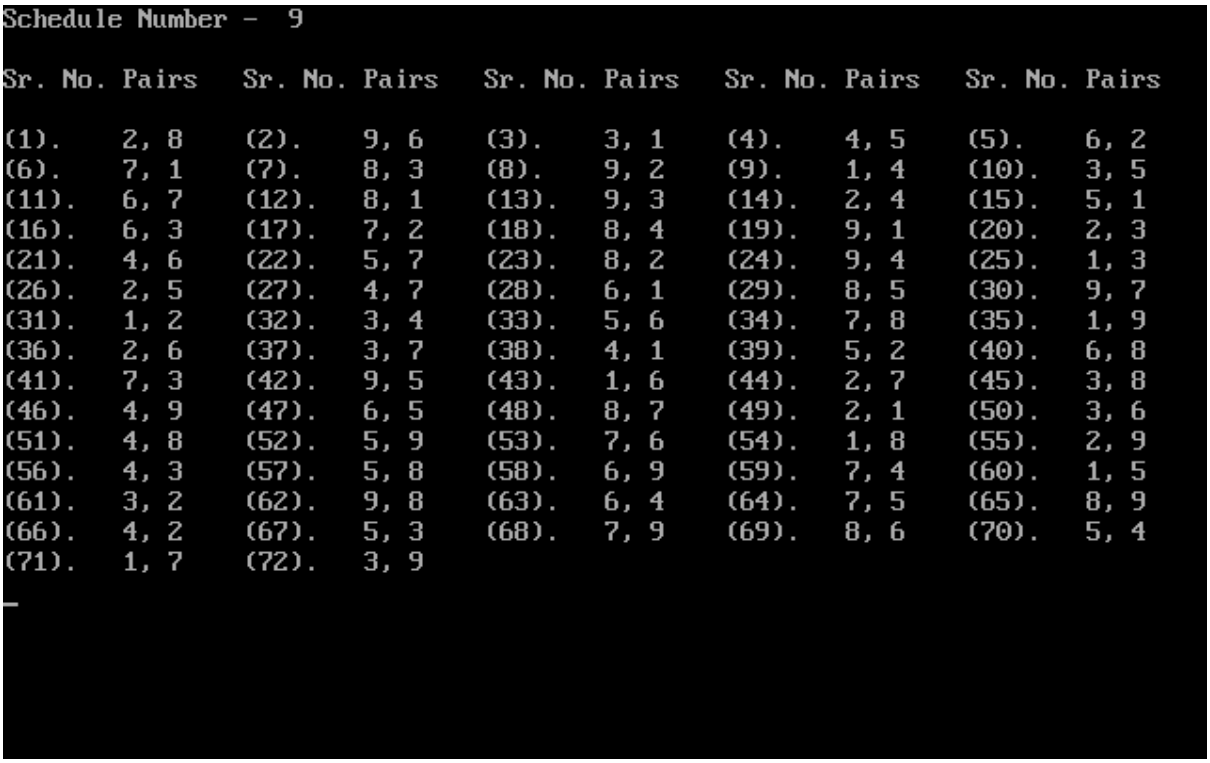


Figure 5.9: IPL Tournament Schedule Number 9

This result show the ninth schedule of the IPL tournament and it start with pair (2, 8).

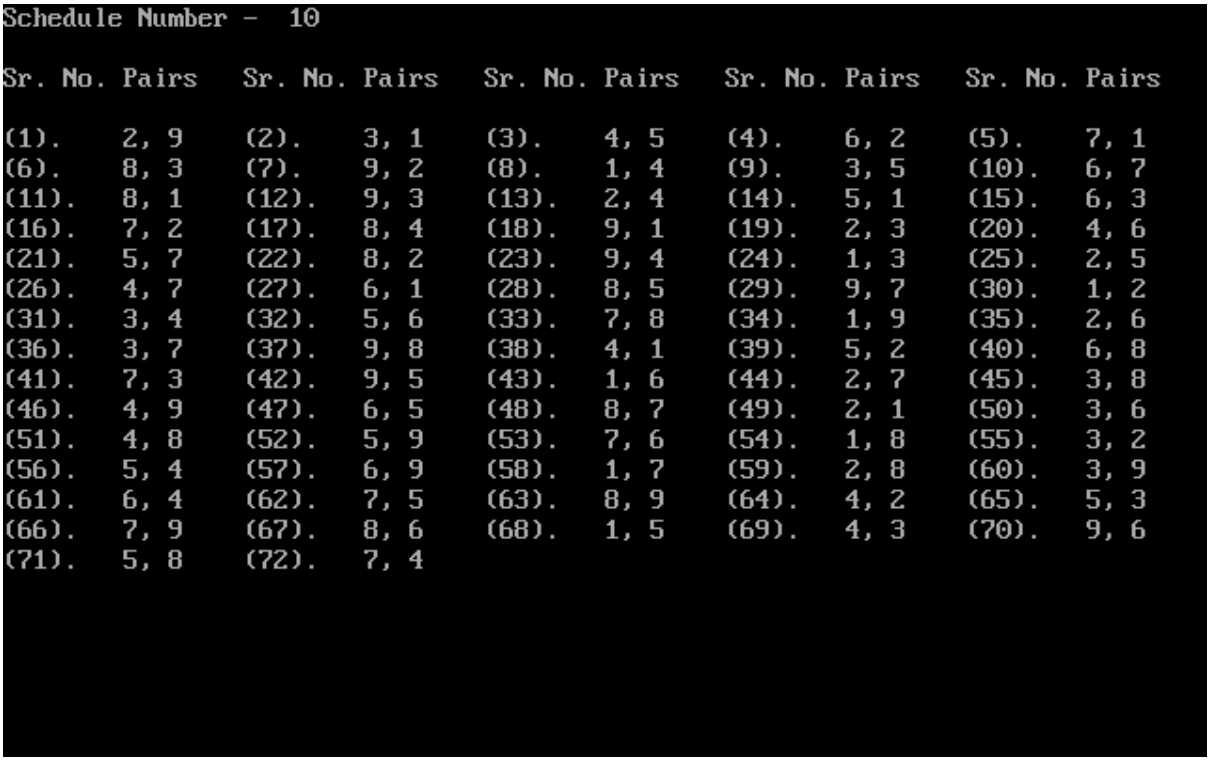


Figure 5.10: IPL Tournament Schedule Number 10

This result show the tenth schedule of the IPL tournament and it start with pair (2, 9).

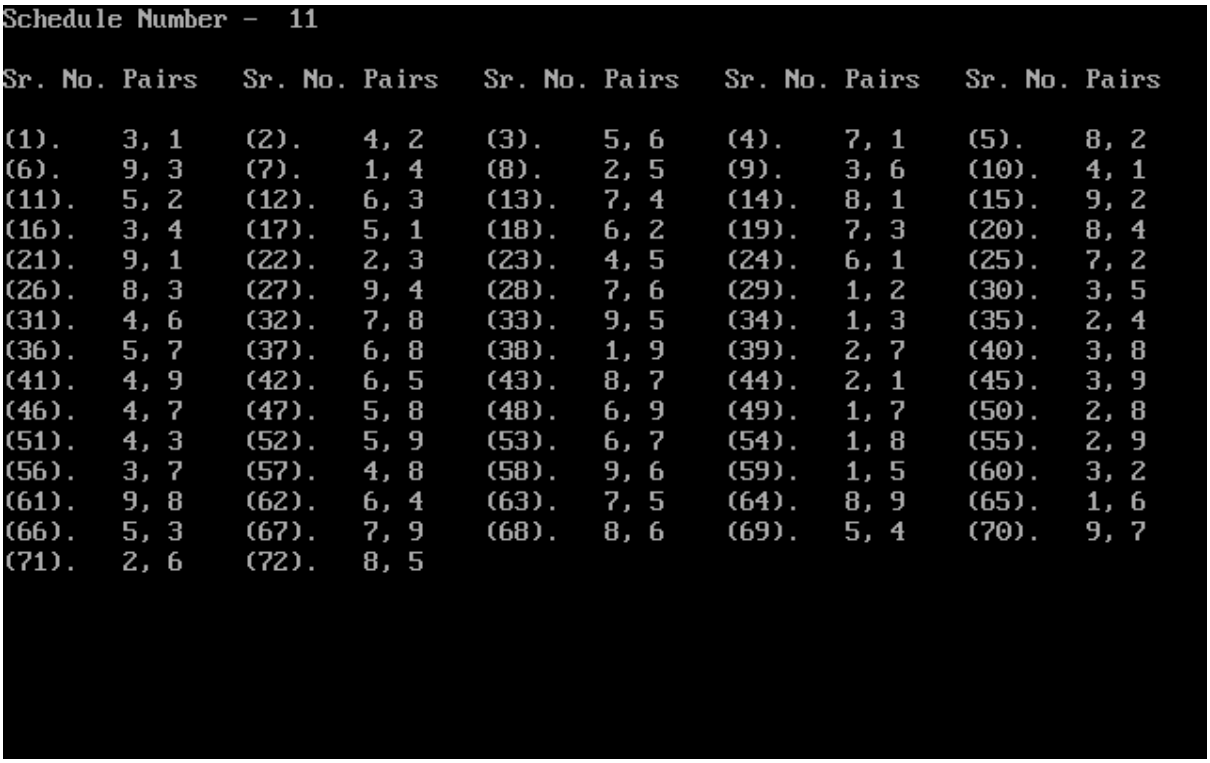
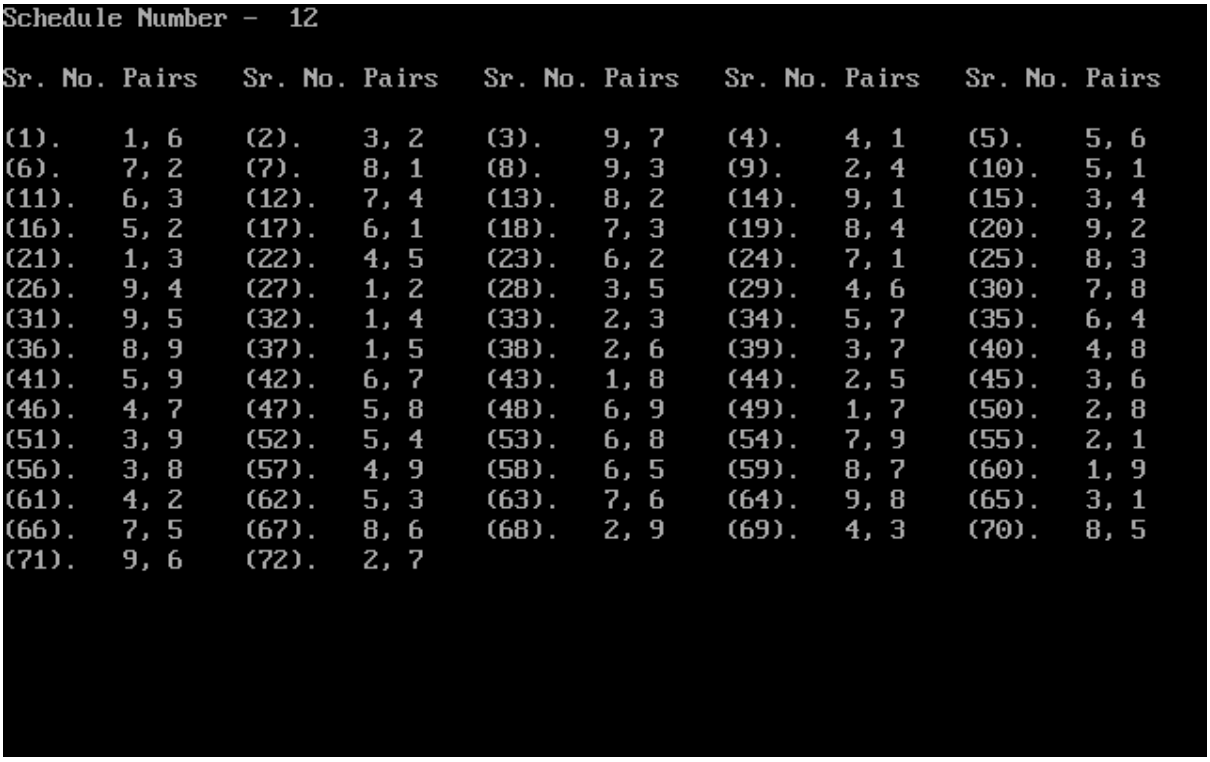


Figure 5.11: IPL Tournament Schedule Number 11

This result show the eleventh schedule of the IPL tournament and it start with pair (3, 1).

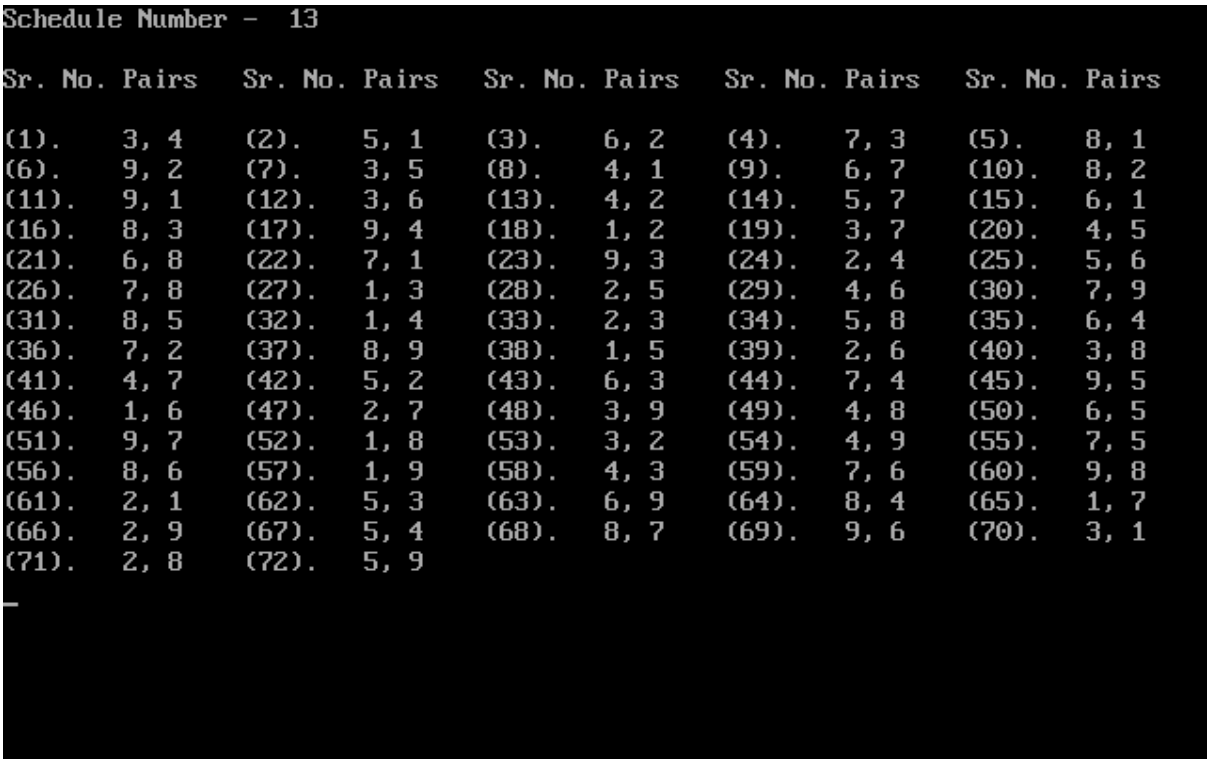


Schedule Number - 12

Sr. No.	Pairs	Sr. No.	Pairs	Sr. No.	Pairs	Sr. No.	Pairs	Sr. No.	Pairs
(1).	1, 6	(2).	3, 2	(3).	9, 7	(4).	4, 1	(5).	5, 6
(6).	7, 2	(7).	8, 1	(8).	9, 3	(9).	2, 4	(10).	5, 1
(11).	6, 3	(12).	7, 4	(13).	8, 2	(14).	9, 1	(15).	3, 4
(16).	5, 2	(17).	6, 1	(18).	7, 3	(19).	8, 4	(20).	9, 2
(21).	1, 3	(22).	4, 5	(23).	6, 2	(24).	7, 1	(25).	8, 3
(26).	9, 4	(27).	1, 2	(28).	3, 5	(29).	4, 6	(30).	7, 8
(31).	9, 5	(32).	1, 4	(33).	2, 3	(34).	5, 7	(35).	6, 4
(36).	8, 9	(37).	1, 5	(38).	2, 6	(39).	3, 7	(40).	4, 8
(41).	5, 9	(42).	6, 7	(43).	1, 8	(44).	2, 5	(45).	3, 6
(46).	4, 7	(47).	5, 8	(48).	6, 9	(49).	1, 7	(50).	2, 8
(51).	3, 9	(52).	5, 4	(53).	6, 8	(54).	7, 9	(55).	2, 1
(56).	3, 8	(57).	4, 9	(58).	6, 5	(59).	8, 7	(60).	1, 9
(61).	4, 2	(62).	5, 3	(63).	7, 6	(64).	9, 8	(65).	3, 1
(66).	7, 5	(67).	8, 6	(68).	2, 9	(69).	4, 3	(70).	8, 5
(71).	9, 6	(72).	2, 7						

Figure 5.12: IPL Tournament Schedule Number 12

This result show the twelfth schedule of the IPL tournament and it start with pair (1, 6).



Schedule Number - 13

Sr. No.	Pairs	Sr. No.	Pairs	Sr. No.	Pairs	Sr. No.	Pairs	Sr. No.	Pairs
(1).	3, 4	(2).	5, 1	(3).	6, 2	(4).	7, 3	(5).	8, 1
(6).	9, 2	(7).	3, 5	(8).	4, 1	(9).	6, 7	(10).	8, 2
(11).	9, 1	(12).	3, 6	(13).	4, 2	(14).	5, 7	(15).	6, 1
(16).	8, 3	(17).	9, 4	(18).	1, 2	(19).	3, 7	(20).	4, 5
(21).	6, 8	(22).	7, 1	(23).	9, 3	(24).	2, 4	(25).	5, 6
(26).	7, 8	(27).	1, 3	(28).	2, 5	(29).	4, 6	(30).	7, 9
(31).	8, 5	(32).	1, 4	(33).	2, 3	(34).	5, 8	(35).	6, 4
(36).	7, 2	(37).	8, 9	(38).	1, 5	(39).	2, 6	(40).	3, 8
(41).	4, 7	(42).	5, 2	(43).	6, 3	(44).	7, 4	(45).	9, 5
(46).	1, 6	(47).	2, 7	(48).	3, 9	(49).	4, 8	(50).	6, 5
(51).	9, 7	(52).	1, 8	(53).	3, 2	(54).	4, 9	(55).	7, 5
(56).	8, 6	(57).	1, 9	(58).	4, 3	(59).	7, 6	(60).	9, 8
(61).	2, 1	(62).	5, 3	(63).	6, 9	(64).	8, 4	(65).	1, 7
(66).	2, 9	(67).	5, 4	(68).	8, 7	(69).	9, 6	(70).	3, 1
(71).	2, 8	(72).	5, 9						

Figure 5.13: IPL Tournament Schedule Number 13

This result show the thirteenth schedule of the IPL tournament and it start with pair (3, 4).

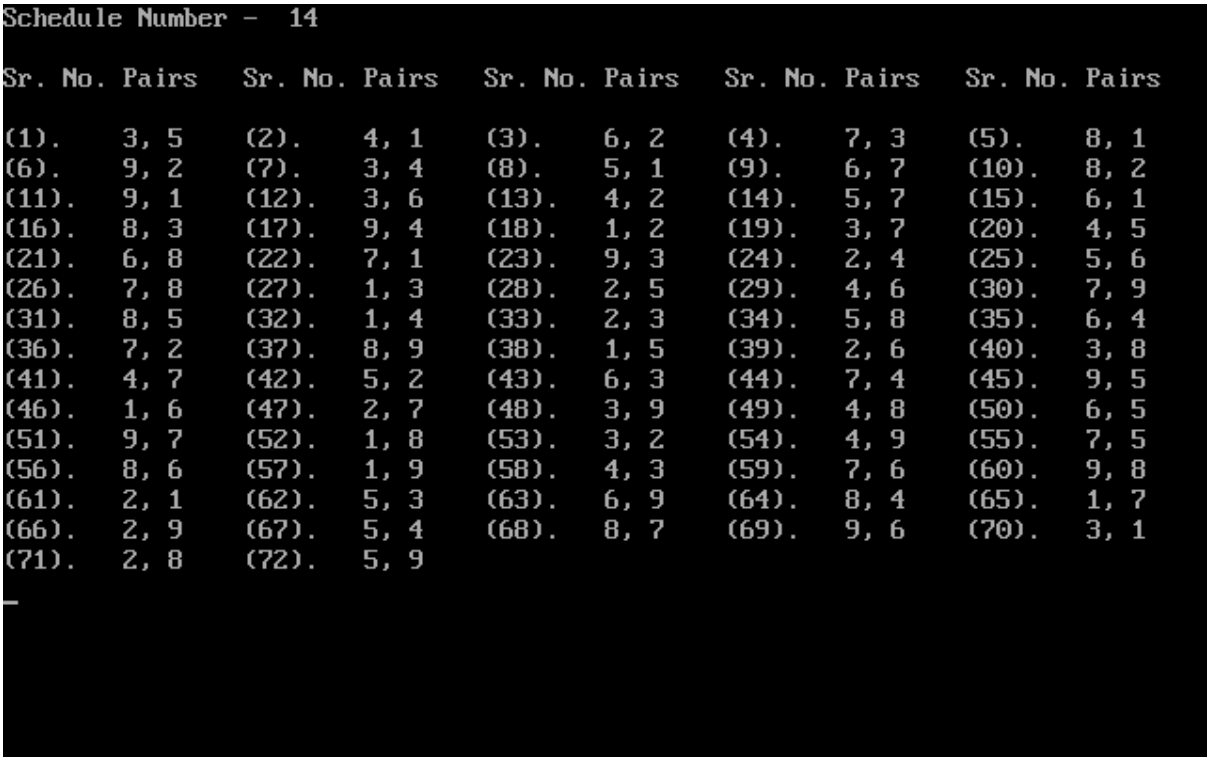


Figure 5.14: IPL Tournament Schedule Number 14

This result show the fourteenth schedule of the IPL tournament and it start with pair (3, 5).

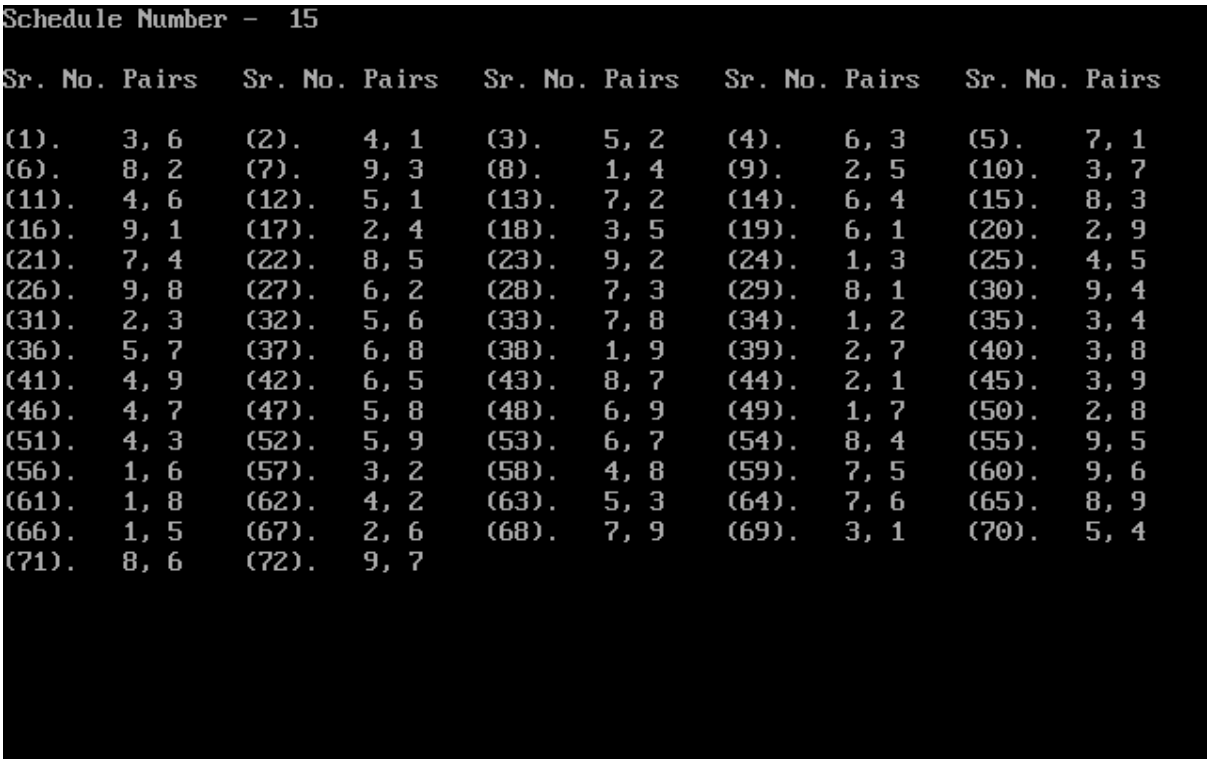


Figure 5.15: IPL Tournament Schedule Number 15

This result show the fifteenth schedule of the IPL tournament and it start with pair (3, 6).

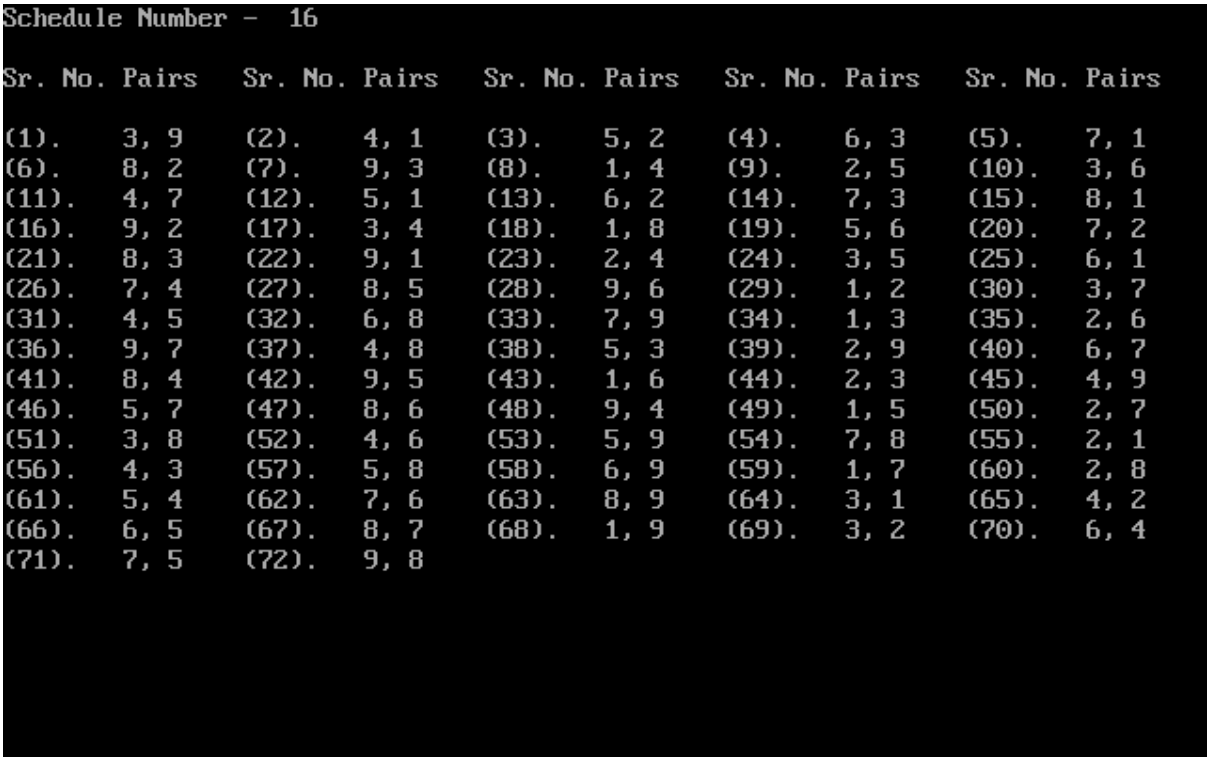


Figure 5.16: IPL Tournament Schedule Number 16

This result show the sixteenth schedule of the IPL tournament and it start with pair (3, 9).

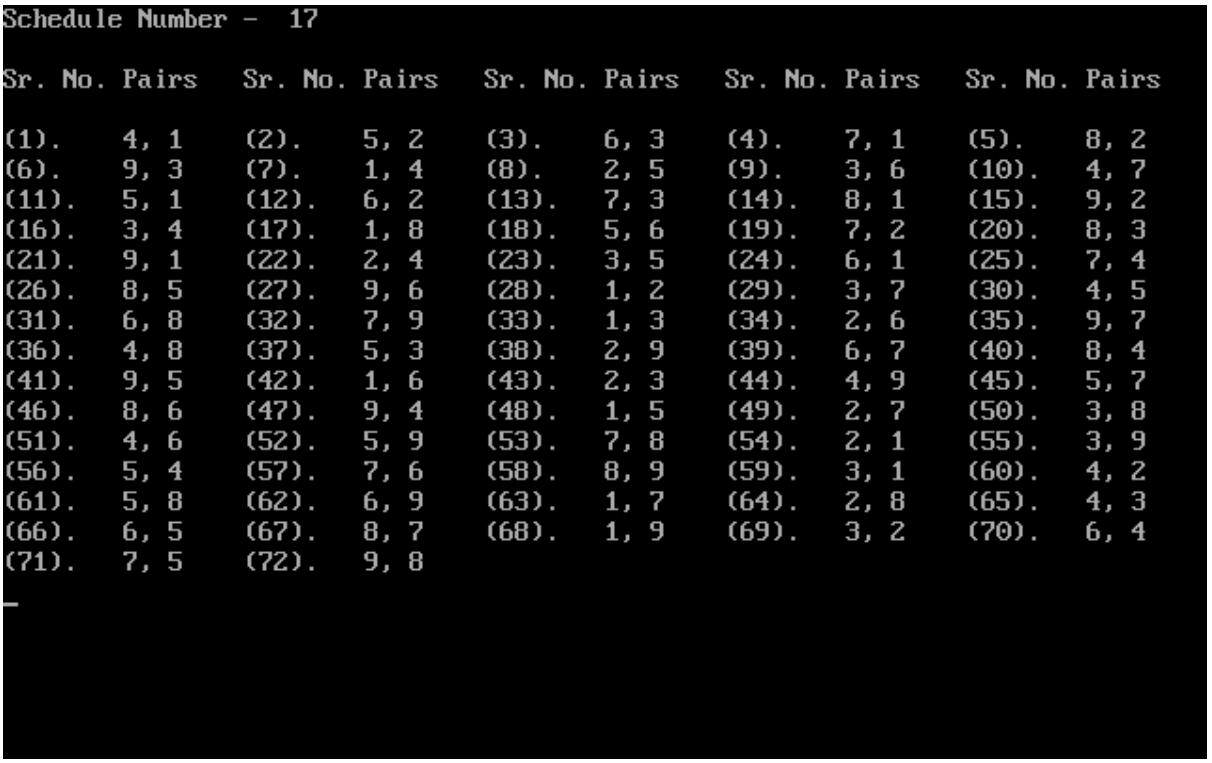


Figure 5.17: IPL Tournament Schedule Number 17

This result show the seventeenth schedule of the IPL tournament and it start with pair (4, 1).

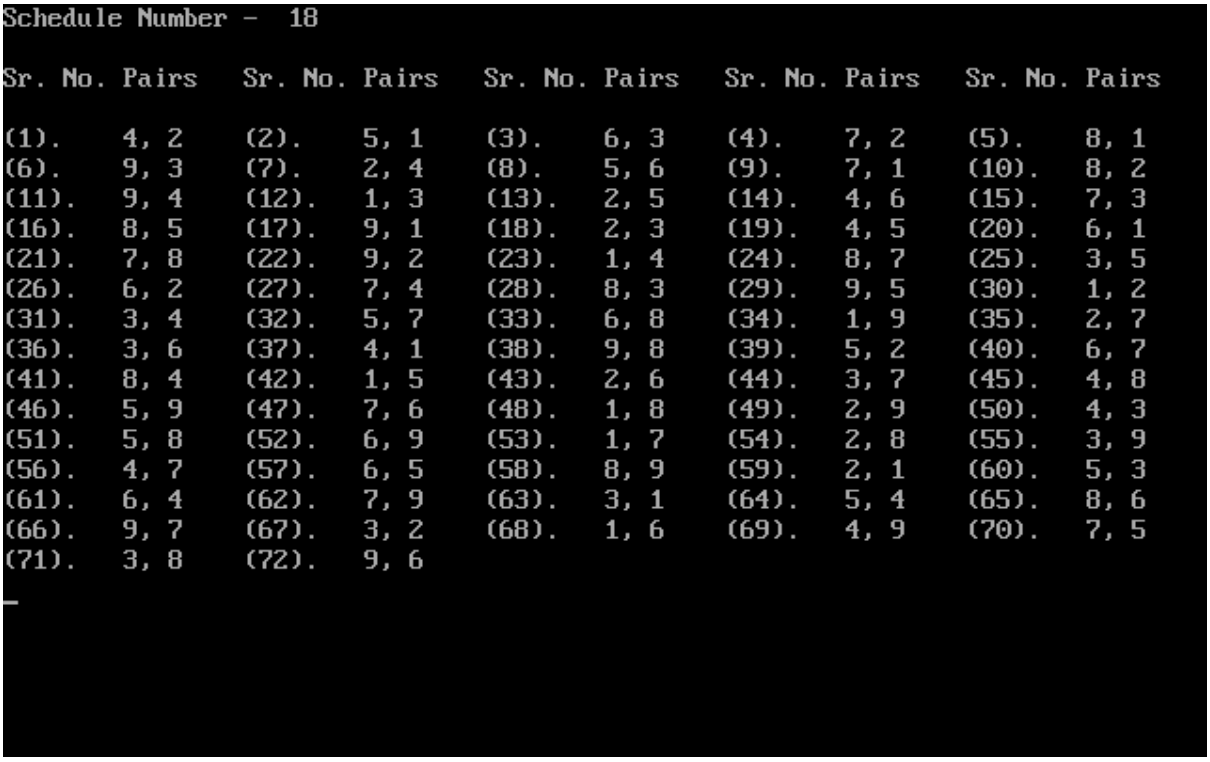


Figure 5.18: IPL Tournament Schedule Number 18

This result show the eighteenth schedule of the IPL tournament and it start with pair (4, 2).

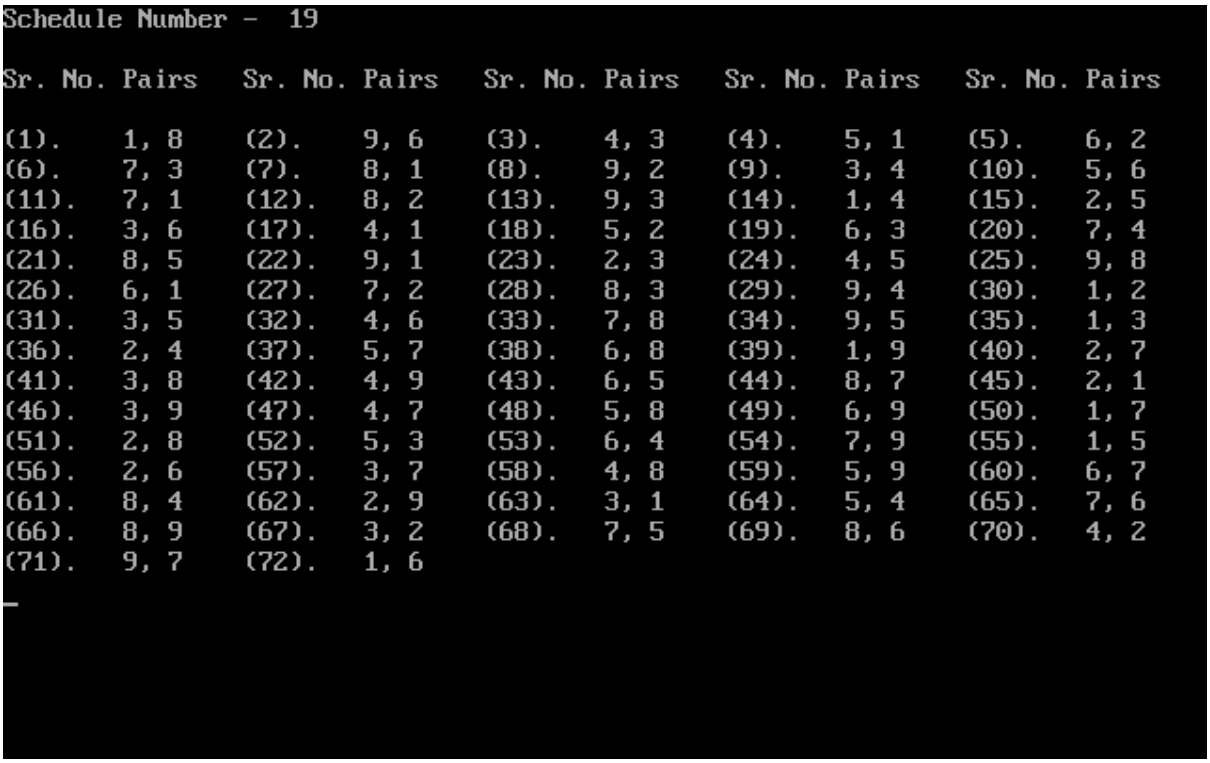


Figure 5.19: IPL Tournament Schedule Number 19

This result show the nineteenth schedule of the IPL tournament and it start with pair (1, 8).

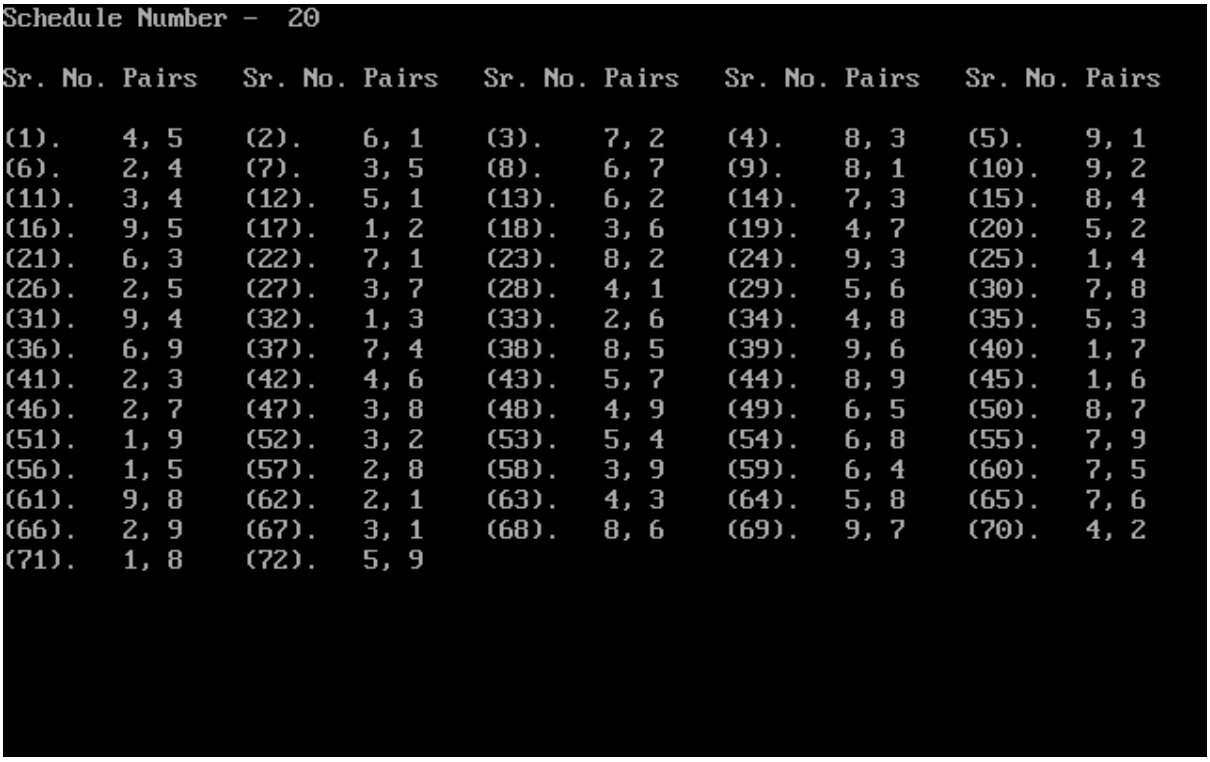


Figure 5.20: IPL Tournament Schedule Number 20

This result show the twentieth schedule of the IPL tournament and it start with pair (4, 5).

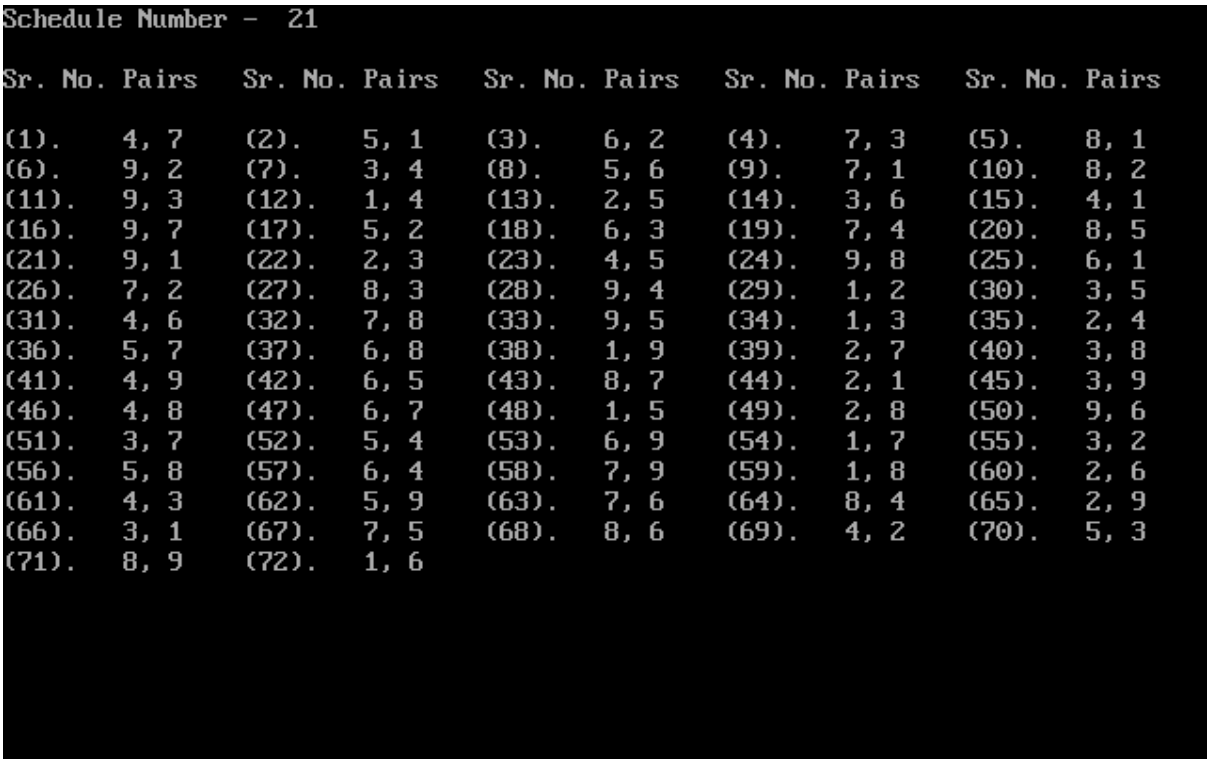


Figure 5.21: IPL Tournament Schedule Number 21

This result show the twenty-first schedule of the IPL tournament and it start with pair (4, 7).

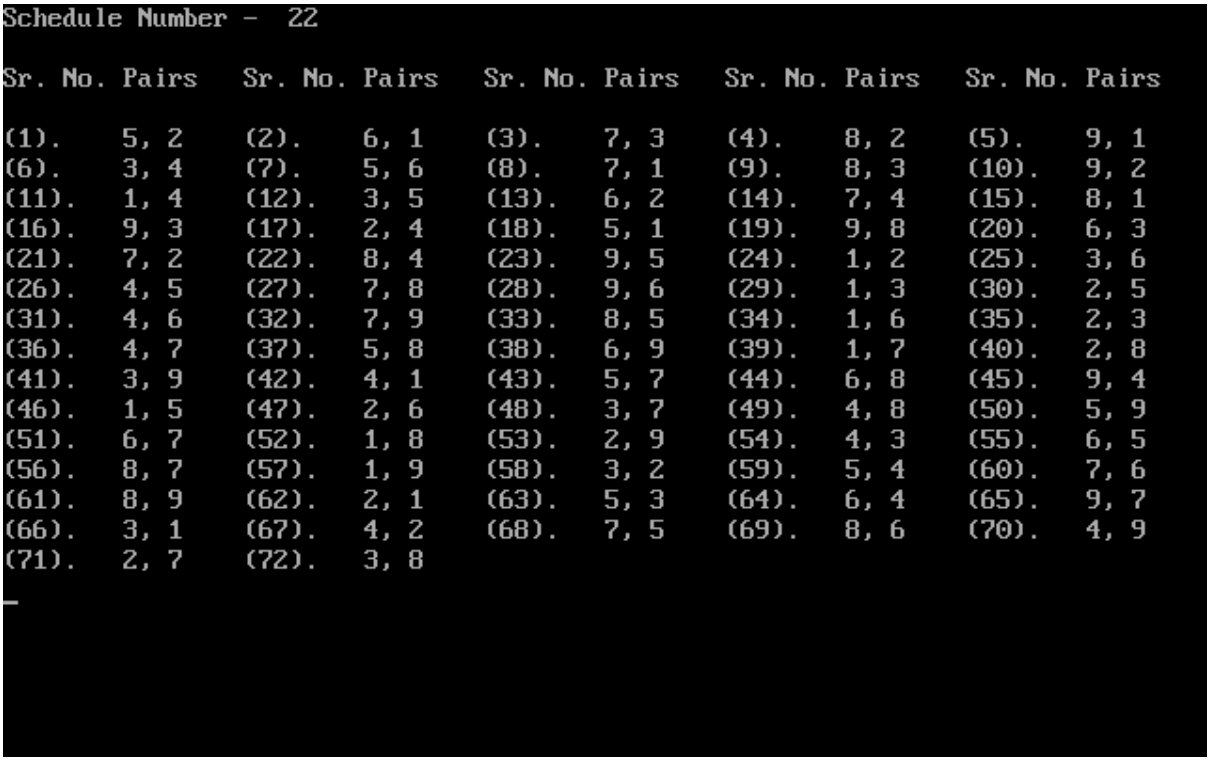


Figure 5.22: IPL Tournament Schedule Number 22

This result show the twenty-second schedule of the IPL tournament and it start with pair (5, 2).

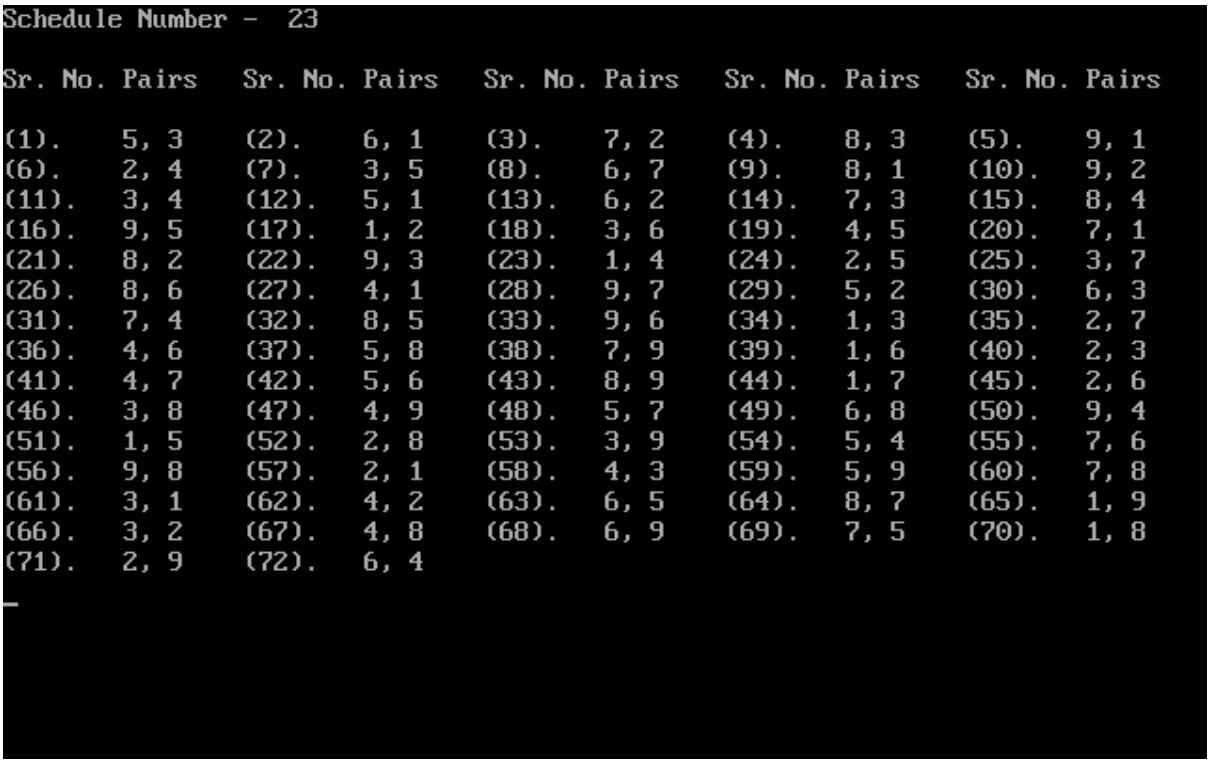
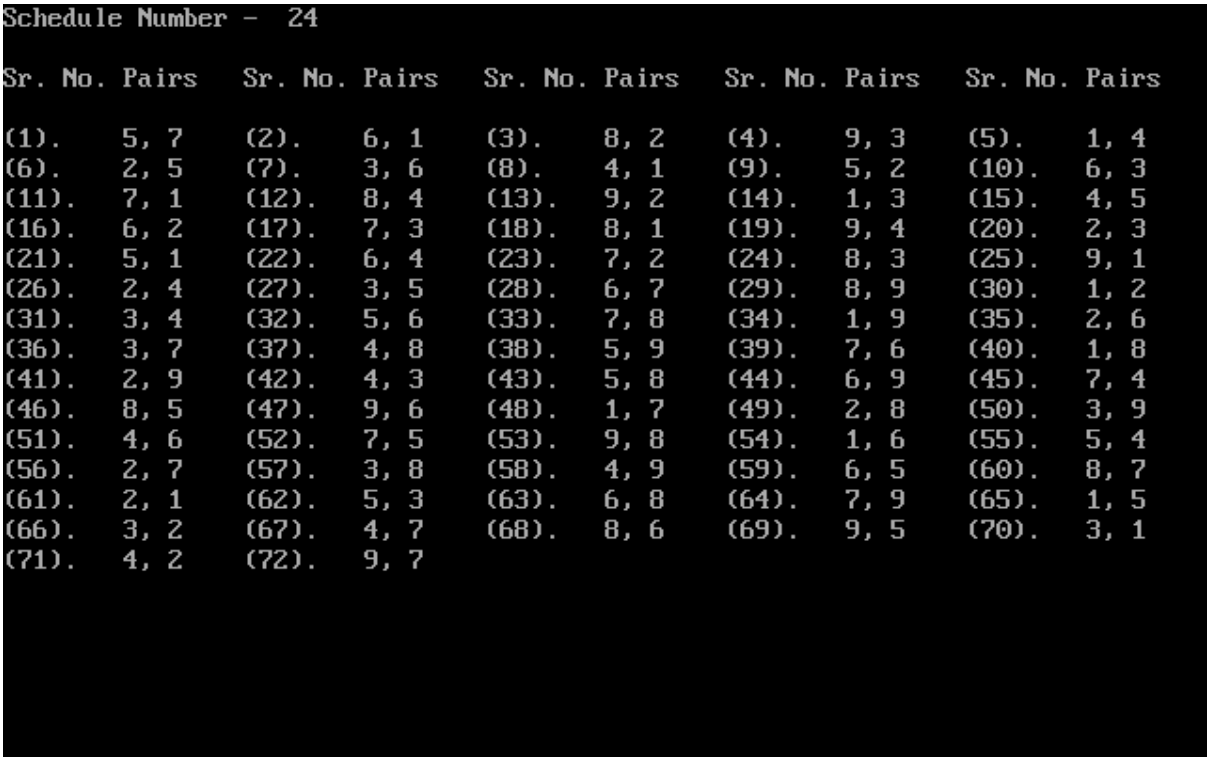


Figure 5.23: IPL Tournament Schedule Number 23

This result show the twenty-third schedule of the IPL tournament and it start with pair (5, 3).

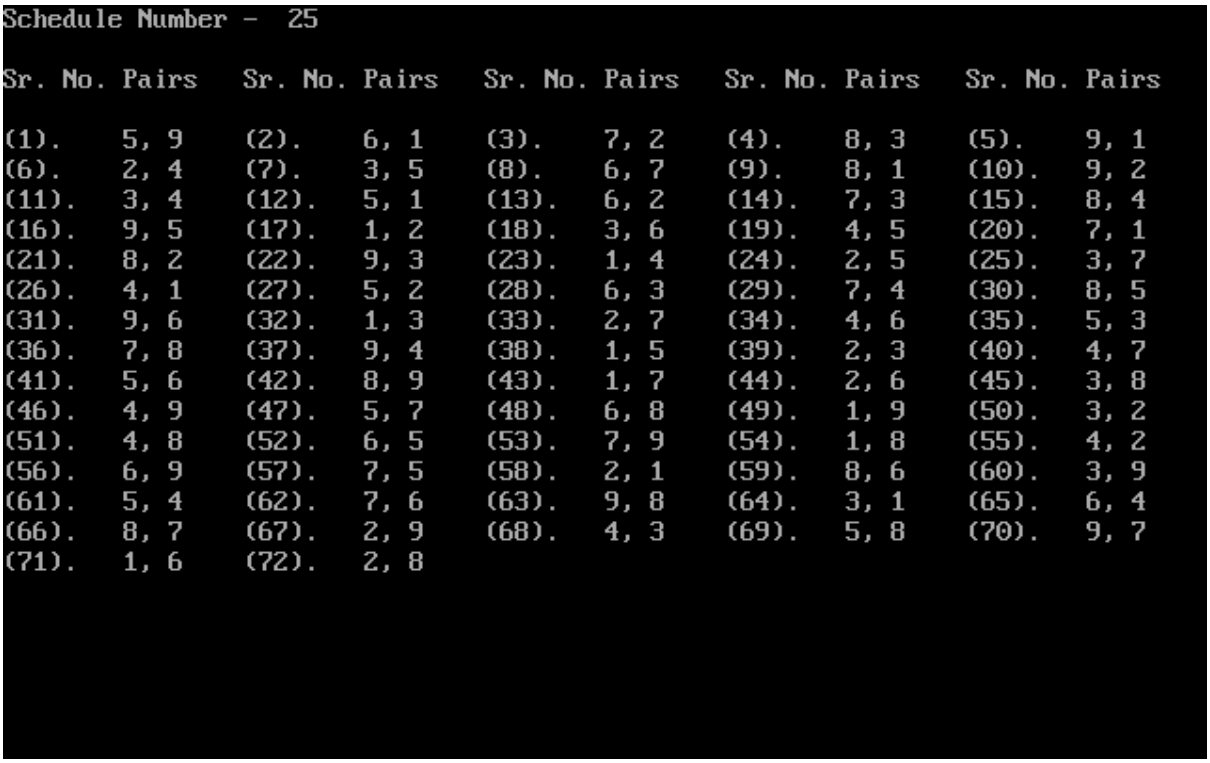


Schedule Number - 24

Sr. No.	Pairs	Sr. No.	Pairs	Sr. No.	Pairs	Sr. No.	Pairs	Sr. No.	Pairs
(1).	5, 7	(2).	6, 1	(3).	8, 2	(4).	9, 3	(5).	1, 4
(6).	2, 5	(7).	3, 6	(8).	4, 1	(9).	5, 2	(10).	6, 3
(11).	7, 1	(12).	8, 4	(13).	9, 2	(14).	1, 3	(15).	4, 5
(16).	6, 2	(17).	7, 3	(18).	8, 1	(19).	9, 4	(20).	2, 3
(21).	5, 1	(22).	6, 4	(23).	7, 2	(24).	8, 3	(25).	9, 1
(26).	2, 4	(27).	3, 5	(28).	6, 7	(29).	8, 9	(30).	1, 2
(31).	3, 4	(32).	5, 6	(33).	7, 8	(34).	1, 9	(35).	2, 6
(36).	3, 7	(37).	4, 8	(38).	5, 9	(39).	7, 6	(40).	1, 8
(41).	2, 9	(42).	4, 3	(43).	5, 8	(44).	6, 9	(45).	7, 4
(46).	8, 5	(47).	9, 6	(48).	1, 7	(49).	2, 8	(50).	3, 9
(51).	4, 6	(52).	7, 5	(53).	9, 8	(54).	1, 6	(55).	5, 4
(56).	2, 7	(57).	3, 8	(58).	4, 9	(59).	6, 5	(60).	8, 7
(61).	2, 1	(62).	5, 3	(63).	6, 8	(64).	7, 9	(65).	1, 5
(66).	3, 2	(67).	4, 7	(68).	8, 6	(69).	9, 5	(70).	3, 1
(71).	4, 2	(72).	9, 7						

Figure 5.24: IPL Tournament Schedule Number 24

This result show the twenty-fourth schedule of the IPL tournament and it start with pair (5, 7).

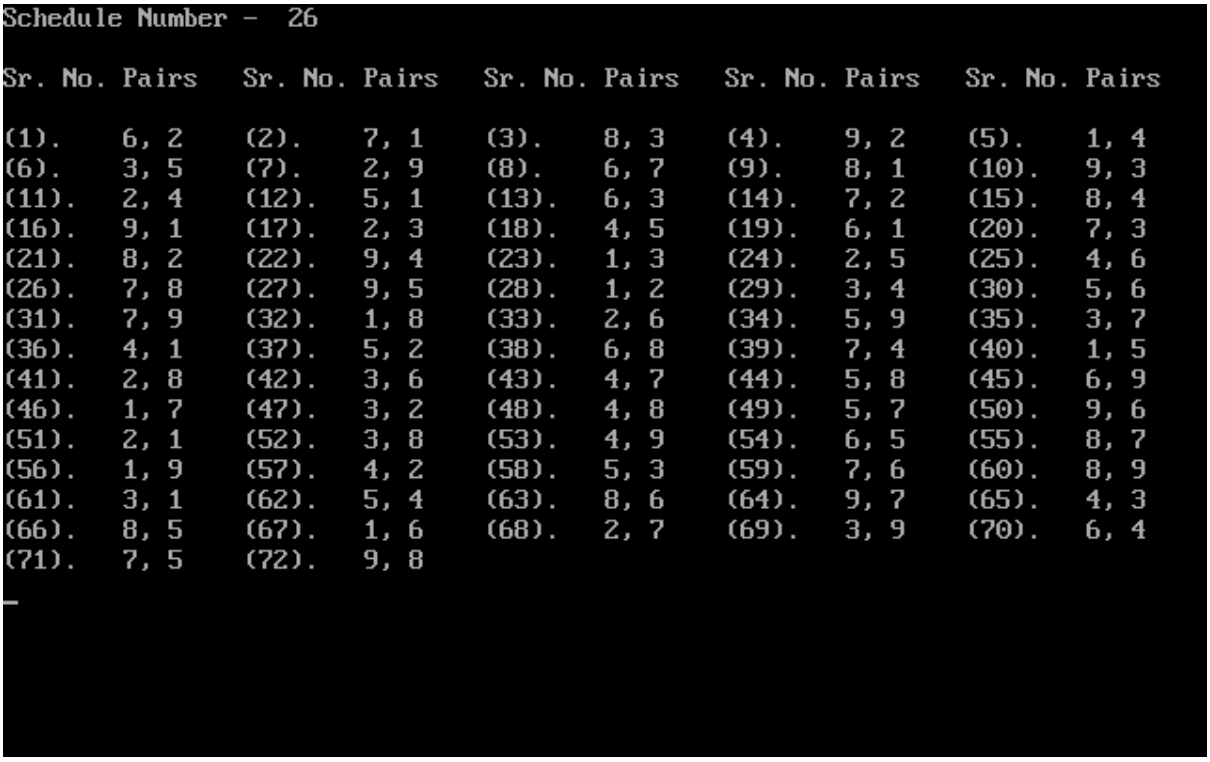


Schedule Number - 25

Sr. No.	Pairs	Sr. No.	Pairs	Sr. No.	Pairs	Sr. No.	Pairs	Sr. No.	Pairs
(1).	5, 9	(2).	6, 1	(3).	7, 2	(4).	8, 3	(5).	9, 1
(6).	2, 4	(7).	3, 5	(8).	6, 7	(9).	8, 1	(10).	9, 2
(11).	3, 4	(12).	5, 1	(13).	6, 2	(14).	7, 3	(15).	8, 4
(16).	9, 5	(17).	1, 2	(18).	3, 6	(19).	4, 5	(20).	7, 1
(21).	8, 2	(22).	9, 3	(23).	1, 4	(24).	2, 5	(25).	3, 7
(26).	4, 1	(27).	5, 2	(28).	6, 3	(29).	7, 4	(30).	8, 5
(31).	9, 6	(32).	1, 3	(33).	2, 7	(34).	4, 6	(35).	5, 3
(36).	7, 8	(37).	9, 4	(38).	1, 5	(39).	2, 3	(40).	4, 7
(41).	5, 6	(42).	8, 9	(43).	1, 7	(44).	2, 6	(45).	3, 8
(46).	4, 9	(47).	5, 7	(48).	6, 8	(49).	1, 9	(50).	3, 2
(51).	4, 8	(52).	6, 5	(53).	7, 9	(54).	1, 8	(55).	4, 2
(56).	6, 9	(57).	7, 5	(58).	2, 1	(59).	8, 6	(60).	3, 9
(61).	5, 4	(62).	7, 6	(63).	9, 8	(64).	3, 1	(65).	6, 4
(66).	8, 7	(67).	2, 9	(68).	4, 3	(69).	5, 8	(70).	9, 7
(71).	1, 6	(72).	2, 8						

Figure 5.25: IPL Tournament Schedule Number 25

This result show the twenty-fifth schedule of the IPL tournament and it start with pair (5, 9).

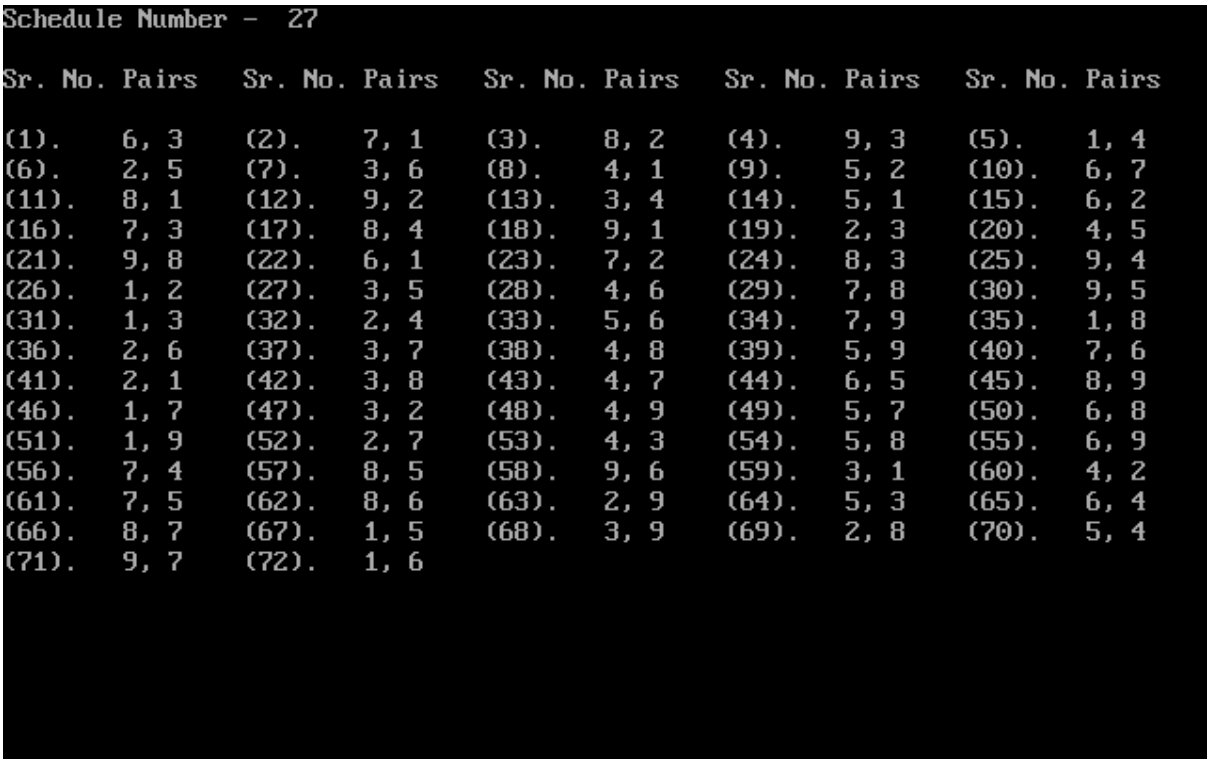


Schedule Number - 26

Sr. No.	Pairs	Sr. No.	Pairs	Sr. No.	Pairs	Sr. No.	Pairs	Sr. No.	Pairs
(1).	6, 2	(2).	7, 1	(3).	8, 3	(4).	9, 2	(5).	1, 4
(6).	3, 5	(7).	2, 9	(8).	6, 7	(9).	8, 1	(10).	9, 3
(11).	2, 4	(12).	5, 1	(13).	6, 3	(14).	7, 2	(15).	8, 4
(16).	9, 1	(17).	2, 3	(18).	4, 5	(19).	6, 1	(20).	7, 3
(21).	8, 2	(22).	9, 4	(23).	1, 3	(24).	2, 5	(25).	4, 6
(26).	7, 8	(27).	9, 5	(28).	1, 2	(29).	3, 4	(30).	5, 6
(31).	7, 9	(32).	1, 8	(33).	2, 6	(34).	5, 9	(35).	3, 7
(36).	4, 1	(37).	5, 2	(38).	6, 8	(39).	7, 4	(40).	1, 5
(41).	2, 8	(42).	3, 6	(43).	4, 7	(44).	5, 8	(45).	6, 9
(46).	1, 7	(47).	3, 2	(48).	4, 8	(49).	5, 7	(50).	9, 6
(51).	2, 1	(52).	3, 8	(53).	4, 9	(54).	6, 5	(55).	8, 7
(56).	1, 9	(57).	4, 2	(58).	5, 3	(59).	7, 6	(60).	8, 9
(61).	3, 1	(62).	5, 4	(63).	8, 6	(64).	9, 7	(65).	4, 3
(66).	8, 5	(67).	1, 6	(68).	2, 7	(69).	3, 9	(70).	6, 4
(71).	7, 5	(72).	9, 8						

Figure 5.26: IPL Tournament Schedule Number 26

This result show the twenty-sixth schedule of the IPL tournament and it start with pair (6, 2).



Schedule Number - 27

Sr. No.	Pairs	Sr. No.	Pairs	Sr. No.	Pairs	Sr. No.	Pairs	Sr. No.	Pairs
(1).	6, 3	(2).	7, 1	(3).	8, 2	(4).	9, 3	(5).	1, 4
(6).	2, 5	(7).	3, 6	(8).	4, 1	(9).	5, 2	(10).	6, 7
(11).	8, 1	(12).	9, 2	(13).	3, 4	(14).	5, 1	(15).	6, 2
(16).	7, 3	(17).	8, 4	(18).	9, 1	(19).	2, 3	(20).	4, 5
(21).	9, 8	(22).	6, 1	(23).	7, 2	(24).	8, 3	(25).	9, 4
(26).	1, 2	(27).	3, 5	(28).	4, 6	(29).	7, 8	(30).	9, 5
(31).	1, 3	(32).	2, 4	(33).	5, 6	(34).	7, 9	(35).	1, 8
(36).	2, 6	(37).	3, 7	(38).	4, 8	(39).	5, 9	(40).	7, 6
(41).	2, 1	(42).	3, 8	(43).	4, 7	(44).	6, 5	(45).	8, 9
(46).	1, 7	(47).	3, 2	(48).	4, 9	(49).	5, 7	(50).	6, 8
(51).	1, 9	(52).	2, 7	(53).	4, 3	(54).	5, 8	(55).	6, 9
(56).	7, 4	(57).	8, 5	(58).	9, 6	(59).	3, 1	(60).	4, 2
(61).	7, 5	(62).	8, 6	(63).	2, 9	(64).	5, 3	(65).	6, 4
(66).	8, 7	(67).	1, 5	(68).	3, 9	(69).	2, 8	(70).	5, 4
(71).	9, 7	(72).	1, 6						

Figure 5.27: IPL Tournament Schedule Number 27

This result show the twenty-seventh schedule of the IPL tournament and it start with pair (6, 3).

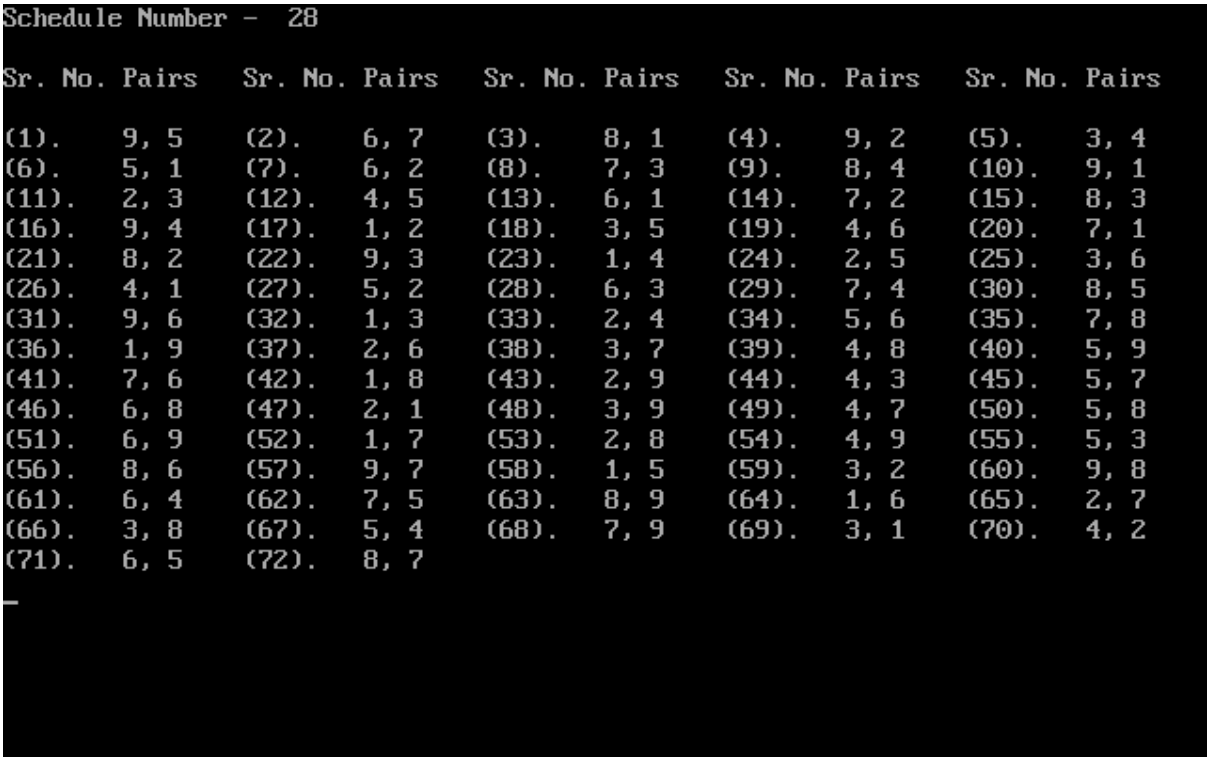


Figure 5.28: IPL Tournament Schedule Number 28

This result show the twenty-eigth schedule of the IPL tournament and it start with pair (9, 5).

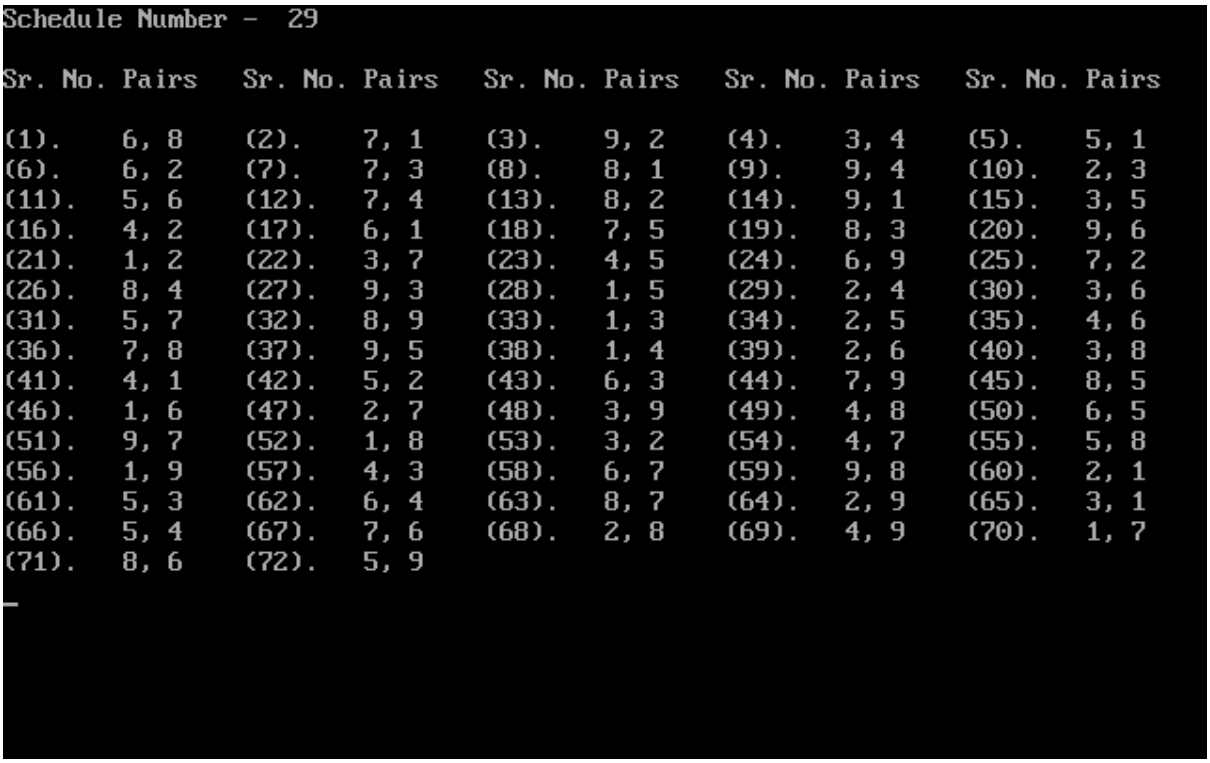


Figure 5.29: IPL Tournament Schedule Number 29

This result show the twenty-ninth schedule of the IPL tournament and it start with pair (6, 8).

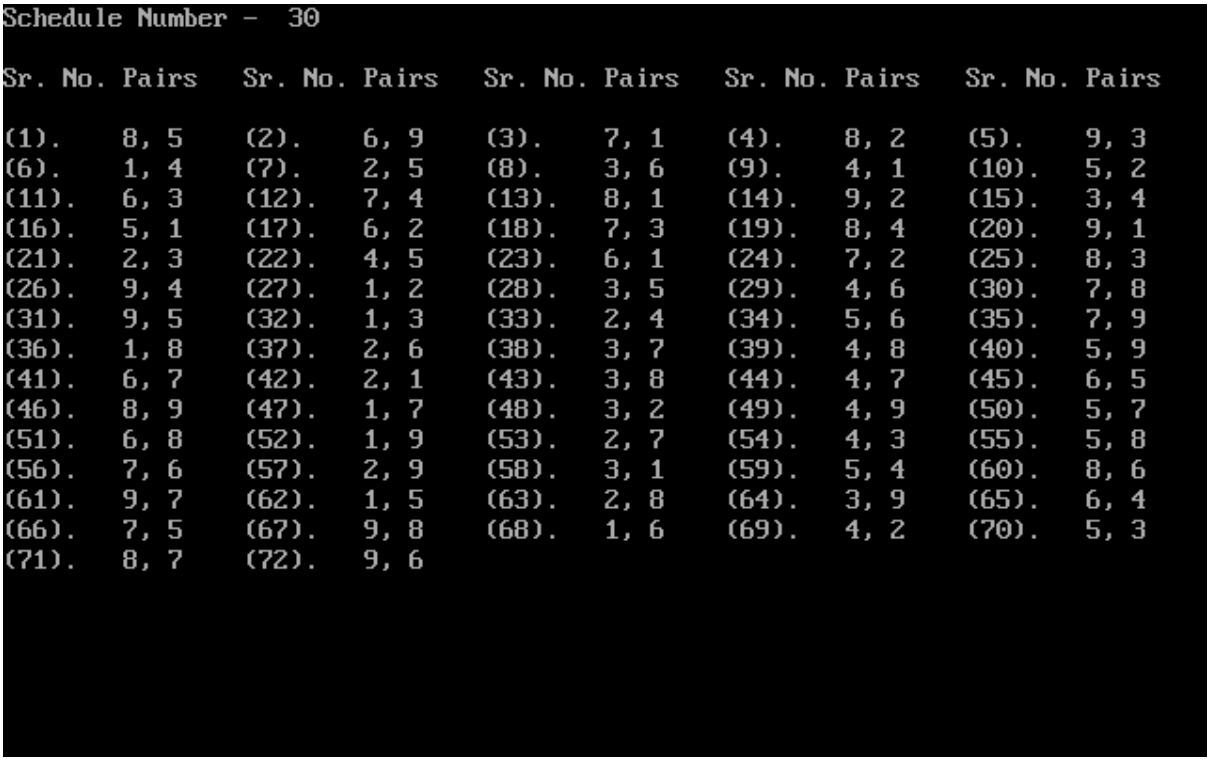


Figure 5.30: IPL Tournament Schedule Number 30

This result show the thirtieth schedule of the IPL tournament and it start with pair (8, 5).

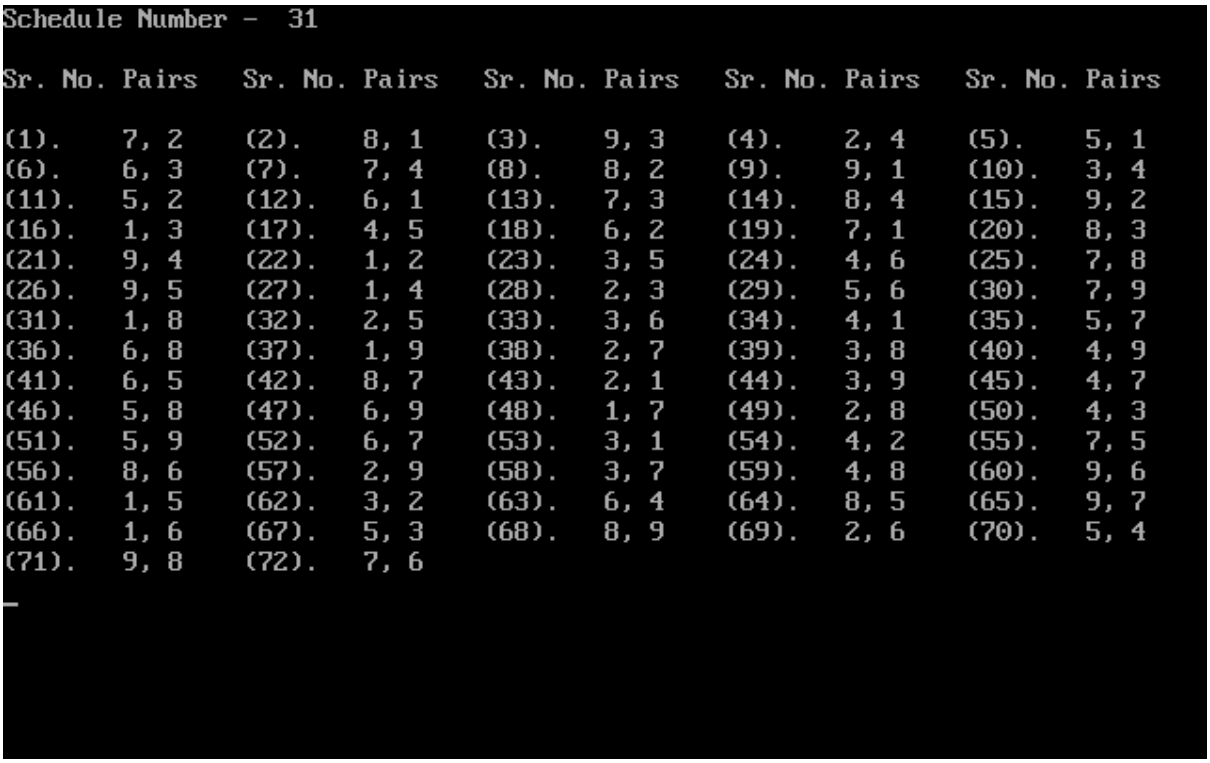
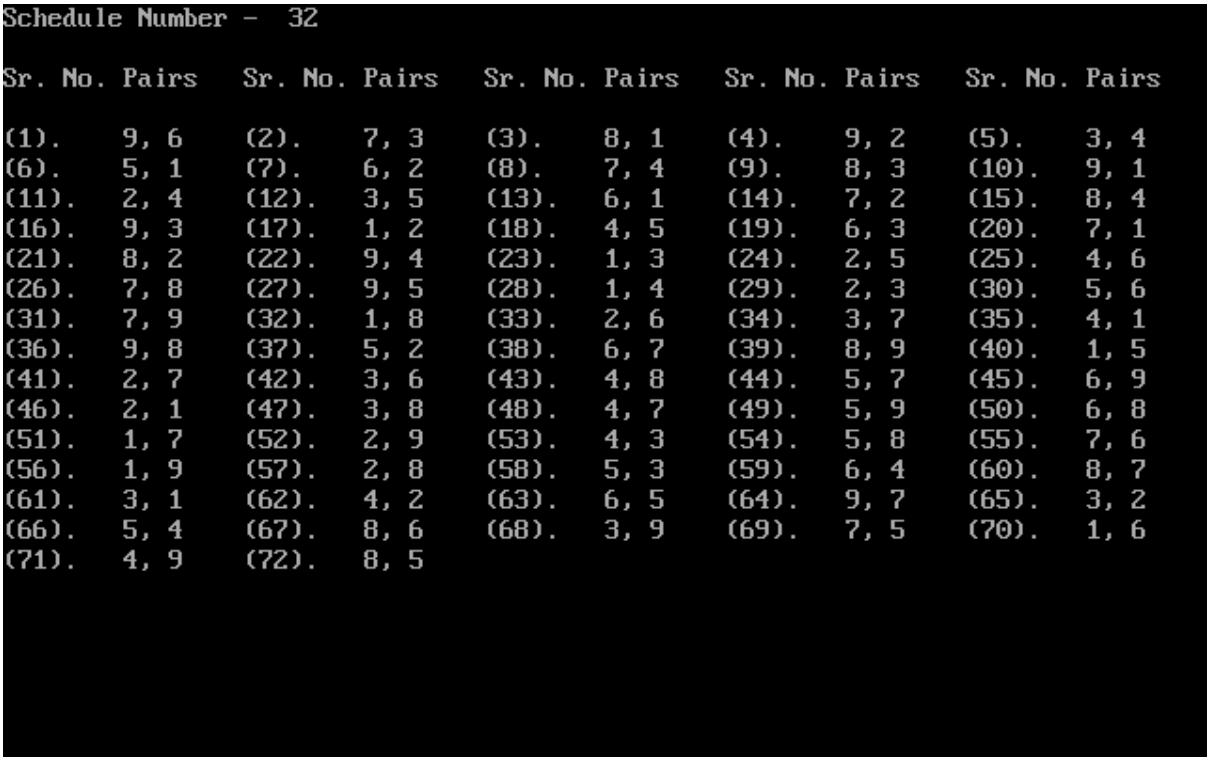


Figure 5.31: IPL Tournament Schedule Number 31

This result show the thirty-first schedule of the IPL tournament and it start with pair (7, 2).

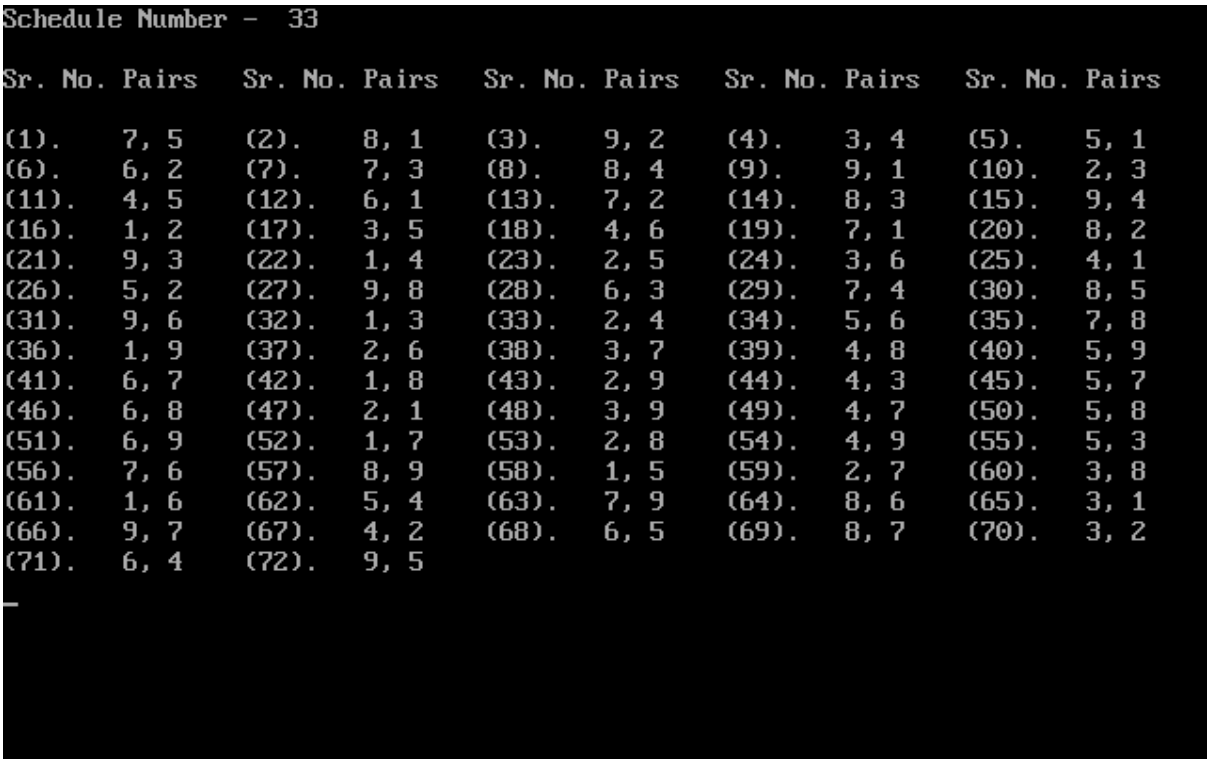


Schedule Number - 32

Sr. No.	Pairs	Sr. No.	Pairs	Sr. No.	Pairs	Sr. No.	Pairs	Sr. No.	Pairs
(1).	9, 6	(2).	7, 3	(3).	8, 1	(4).	9, 2	(5).	3, 4
(6).	5, 1	(7).	6, 2	(8).	7, 4	(9).	8, 3	(10).	9, 1
(11).	2, 4	(12).	3, 5	(13).	6, 1	(14).	7, 2	(15).	8, 4
(16).	9, 3	(17).	1, 2	(18).	4, 5	(19).	6, 3	(20).	7, 1
(21).	8, 2	(22).	9, 4	(23).	1, 3	(24).	2, 5	(25).	4, 6
(26).	7, 8	(27).	9, 5	(28).	1, 4	(29).	2, 3	(30).	5, 6
(31).	7, 9	(32).	1, 8	(33).	2, 6	(34).	3, 7	(35).	4, 1
(36).	9, 8	(37).	5, 2	(38).	6, 7	(39).	8, 9	(40).	1, 5
(41).	2, 7	(42).	3, 6	(43).	4, 8	(44).	5, 7	(45).	6, 9
(46).	2, 1	(47).	3, 8	(48).	4, 7	(49).	5, 9	(50).	6, 8
(51).	1, 7	(52).	2, 9	(53).	4, 3	(54).	5, 8	(55).	7, 6
(56).	1, 9	(57).	2, 8	(58).	5, 3	(59).	6, 4	(60).	8, 7
(61).	3, 1	(62).	4, 2	(63).	6, 5	(64).	9, 7	(65).	3, 2
(66).	5, 4	(67).	8, 6	(68).	3, 9	(69).	7, 5	(70).	1, 6
(71).	4, 9	(72).	8, 5						

Figure 5.32: IPL Tournament Schedule Number 32

This result show the thirty-second schedule of the IPL tournament and it start with pair (9, 6).



Schedule Number - 33

Sr. No.	Pairs	Sr. No.	Pairs	Sr. No.	Pairs	Sr. No.	Pairs	Sr. No.	Pairs
(1).	7, 5	(2).	8, 1	(3).	9, 2	(4).	3, 4	(5).	5, 1
(6).	6, 2	(7).	7, 3	(8).	8, 4	(9).	9, 1	(10).	2, 3
(11).	4, 5	(12).	6, 1	(13).	7, 2	(14).	8, 3	(15).	9, 4
(16).	1, 2	(17).	3, 5	(18).	4, 6	(19).	7, 1	(20).	8, 2
(21).	9, 3	(22).	1, 4	(23).	2, 5	(24).	3, 6	(25).	4, 1
(26).	5, 2	(27).	9, 8	(28).	6, 3	(29).	7, 4	(30).	8, 5
(31).	9, 6	(32).	1, 3	(33).	2, 4	(34).	5, 6	(35).	7, 8
(36).	1, 9	(37).	2, 6	(38).	3, 7	(39).	4, 8	(40).	5, 9
(41).	6, 7	(42).	1, 8	(43).	2, 9	(44).	4, 3	(45).	5, 7
(46).	6, 8	(47).	2, 1	(48).	3, 9	(49).	4, 7	(50).	5, 8
(51).	6, 9	(52).	1, 7	(53).	2, 8	(54).	4, 9	(55).	5, 3
(56).	7, 6	(57).	8, 9	(58).	1, 5	(59).	2, 7	(60).	3, 8
(61).	1, 6	(62).	5, 4	(63).	7, 9	(64).	8, 6	(65).	3, 1
(66).	9, 7	(67).	4, 2	(68).	6, 5	(69).	8, 7	(70).	3, 2
(71).	6, 4	(72).	9, 5						

Figure 5.33: IPL Tournament Schedule Number 33

This result show the thirty-third schedule of the IPL tournament and it start with pair (7, 5).

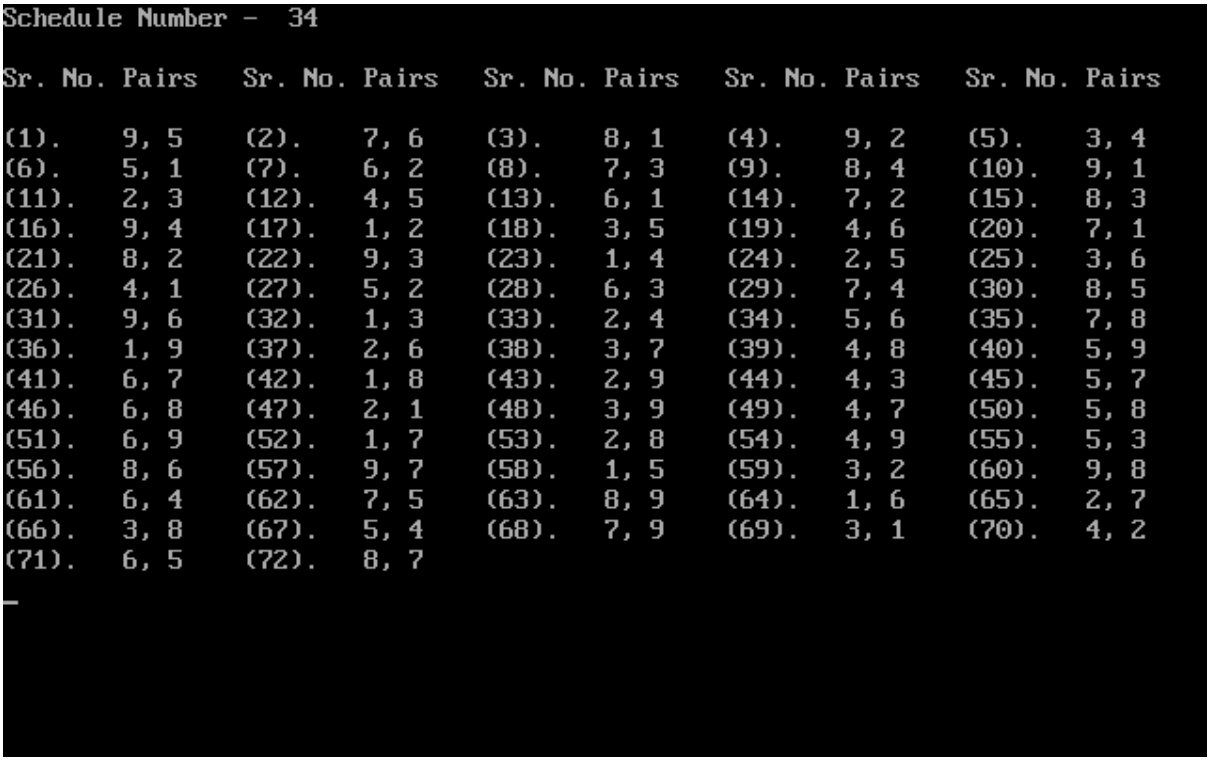


Figure 5.34: IPL Tournament Schedule Number 34

This result show the thirty-forth schedule of the IPL tournament and it start with pair (9, 5).

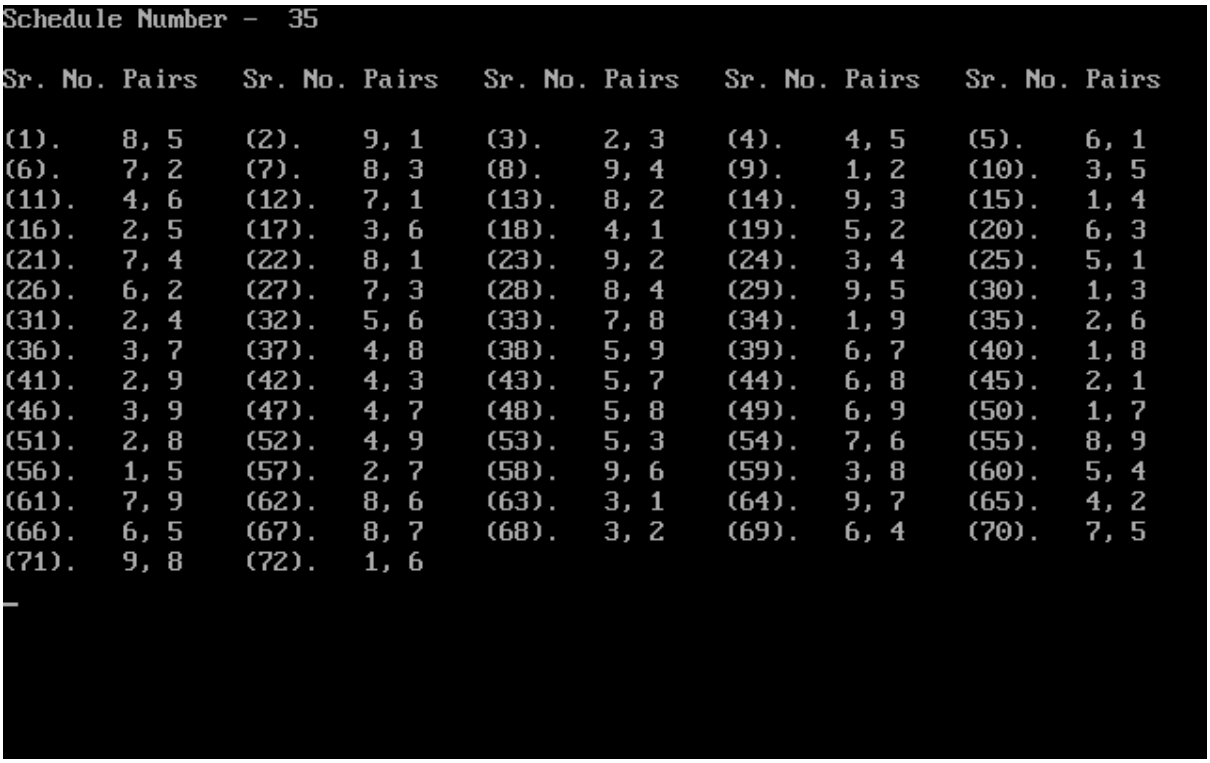


Figure 5.35: IPL Tournament Schedule Number 35

This result show the thirty-fifth schedule of the IPL tournament and it start with pair (8, 5).

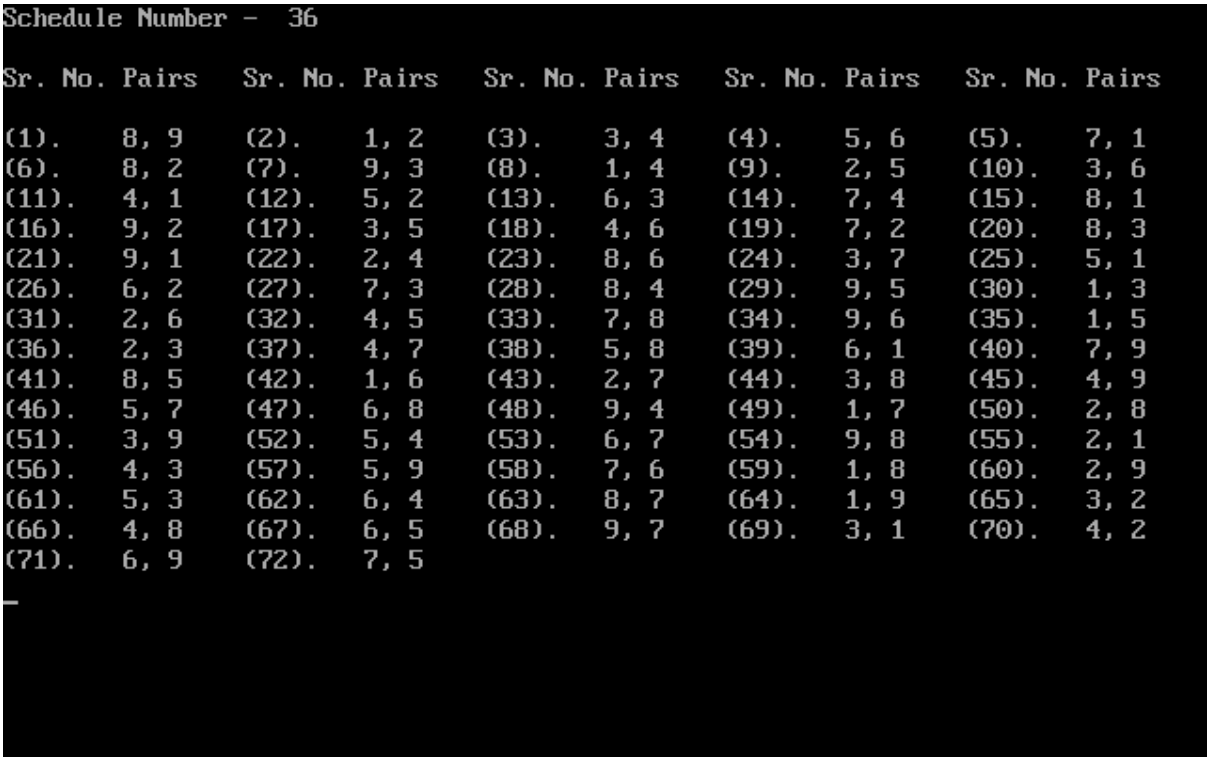


Figure 5.36: IPL Tournament Schedule Number 36

This result show the thirty-sixth schedule of the IPL tournament and it start with pair (8, 9).

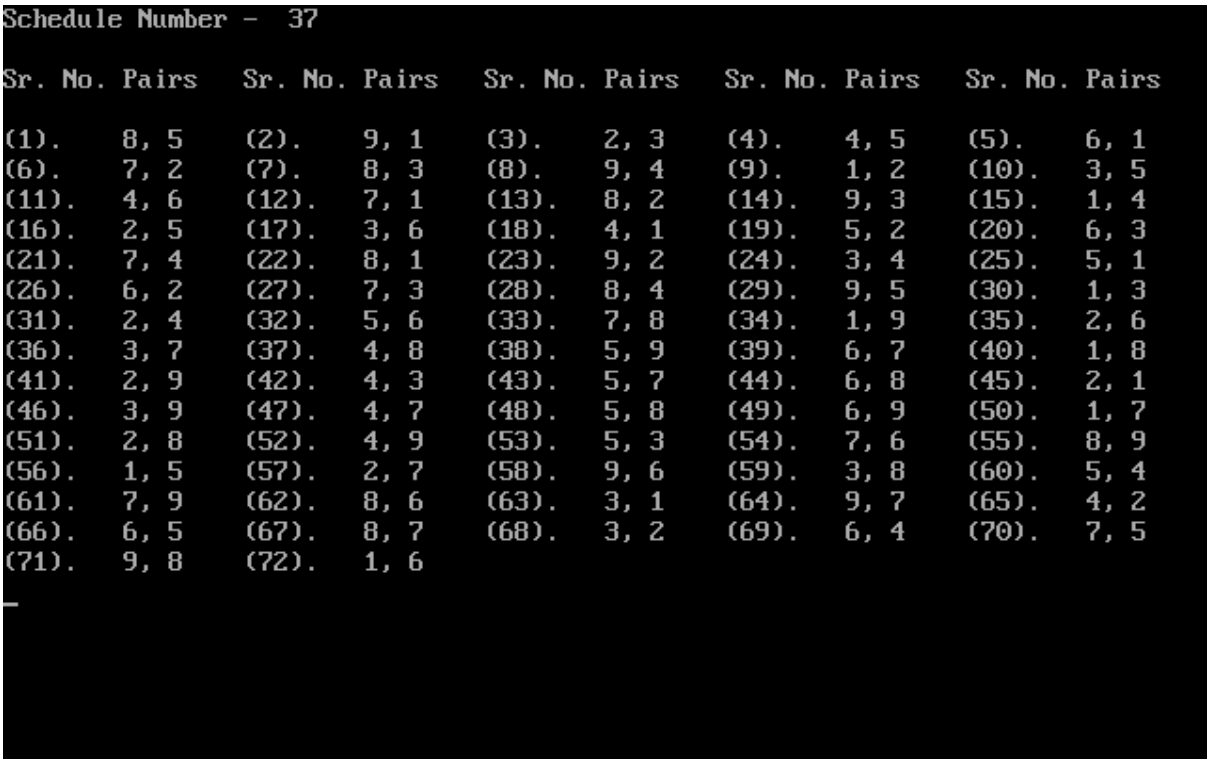


Figure 5.37: IPL Tournament Schedule Number 37

This result show the thirty-seventh schedule of the IPL tournament and it start with pair (8, 5).

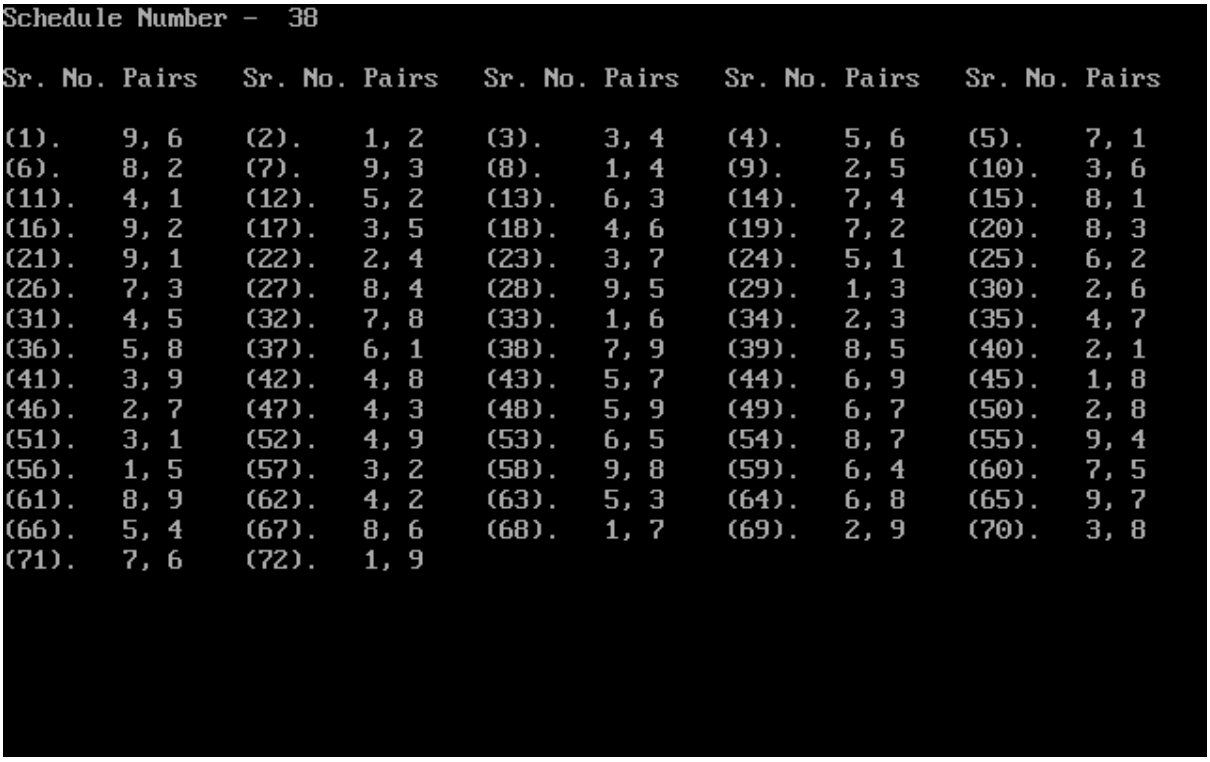


Figure 5.38: IPL Tournament Schedule Number 38

This result show the thirty-eighth schedule of the IPL tournament and it start with pair (9, 6).

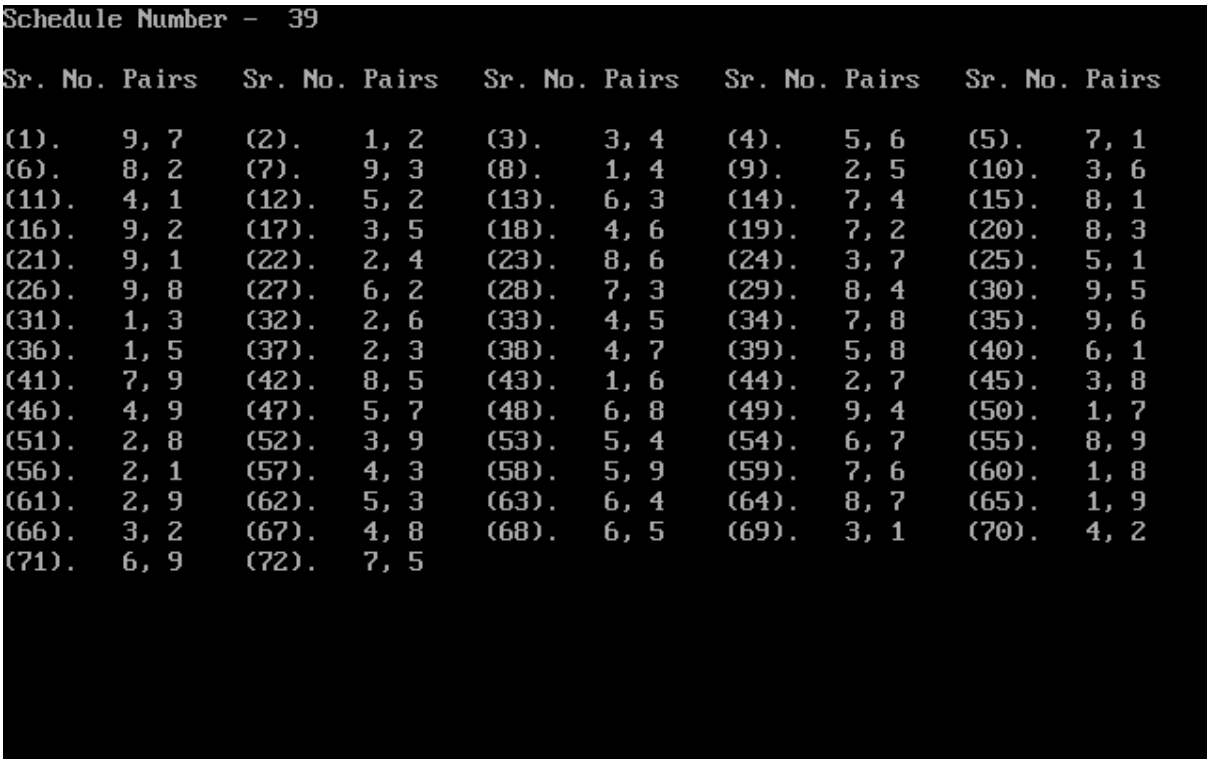


Figure 5.39: IPL Tournament Schedule Number 39

This result show the thirty-ninth schedule of the IPL tournament and it start with pair (9, 7).

5.2 Results and Conclusions

It can be concluded from the results that relaxation for each team and it will increase energy of the team. It is also decrease the pressure of teams and provide the approximate equal probability to each and every team to get the gap for the relaxation. It is also provide the number of tournament schedule for the same tournament with the same constraints. All schedule help to save the partiality means it does allow the partiality for the all team. It also provide the all

Chapter 6

Conclusion and Future Scope

This chapter discusses the conclusions of the work presented in this thesis. This chapter ends with a discussion of the future direction which can be taken further.

6.1 Conclusion

This thesis gives the introduction of IPL (Indian premier League) tournament schedules using iterative backtracking method. It is also fulfil the all previous constraints of the IPL tournament. It will provide more relax to each team of IPL.

The primary goal of this tournament design is to provide incentives for the participants to maximize their performance during the tournament. And the secondary goal of tournament design is to also promote interesting match using gap that generate fan interest. IPL tournaments, in general, promote both goals. Every team will get equal opportunities for relax and gap of them matches. Two matches gap is sufficient for all the team for relax.

In this work we have got the worst case time complexity such as O (!n). This algorithm will work for more than 9 teams i.e. n >= 9.

6.2 Thesis Contribution

a) In this thesis a new IPL (Indian Premier League) tournament schedule has been analyzed and implement.

b) A IPL (Indian Premier League) tournament schedule algorithm provide use number of schedule with same constraints.

c) The designed has been implemented with n teams and explain with 9 teams, i.e. n=9.

d) Experimental results have been gathered.

6.3 Future Scope

a) This algorithm is providing us number of schedules and we can further develop this algorithm to find the best schedule according to some cost or time constraints.

b) Further we can implement it with recursive backtracking approach and then we can compare with this tournament schedule algorithm.
c) In future we can compare this algorithm with recursive backtracking approach with some constraints and check for the better once.
d) This algorithm is proving number of schedules but there may be some more schedules with same constraints.

References

[1] K. Coolsaet, J. Degraer, "A computer assisted proof of the uniqueness of the Perkel graph, Designs Codes Cryptogr," *34 (2–3) (2005) 155–171,* 2005.

[2] Bing JIANG, Yongmin MU, Zhihua Zhang, "Research of Optimization Algorithm for Path-Based Regression Testing Suit: Proceedings of Second International Workshop on Education Technology and Computer Science," *IEEE,* 2010.

[3] Leung H. K. N, White L, "Insights into regression testing: Proceedings of International conference on Software Maintenance," LOS Alamipos: *IEEE Computer Society, 1989, pp. 60-69.*

[4] David G. Sullivan, " Recursion and Recursive Backtracking*," Computer Science E-119Harvard Extension School,* 2012.

[5] David S. Johnson, Michael A. Trick, "Cliques, Coloring, and Satisfiability: Second Dimacs Implementation Challenge," *American Mathematical Soc., 01-Jan-1996,October 11-13, 1993 (Google eBook).*

[6] M. Melcher and K. B. Reid.," Monochromatic sinks in nearly transitive arccolored Tournaments," *Discrete Math., 310(20):2697–2704,* 2010.

[7] H.A. Priestley, M.P. Ward," A Multipurpose Backtracking Algorithm," *Ph.D. thesis, Mathematical Institute 24/29, St. Giles Oxford OX1 3LB,*January 17, 2003.

[8] Ambjörn, Seatrack," Web forecasts and backtracking of oil spills-an efficient tool to find illegal spills using AIS," *US/EU-Baltic International Symposium, 2008 IEEE/OES, pp 1-9, C,* 2008.

[9] Gerhart, S. L. & Yelowitz, L., "Control Structure Abstractions of the Backtracking Programming Technique," *IEEE Trans. Software Eng. SE 2, 4, pp. 285{292,* 1976.

[10] Ward, M.," Derivation of Data Intensive Algorithms by Formal Transformation," *IEEE trans. Software Eng. 22, 9, pp. 665{686. hhttp://www.dur.ac.uk/»dcs0mpw/martin/papers/sw-alg.ps.gzi,* 1996.

[11] Yoshiko T. Ikebe , Akihisa Tamura, " Construction of Hamilton Path Tournament Designs," in *Springer, Graphs and Combinatorics (2011) 27:703–711, DOI 10.1007/s00373-010-0998-6,* 2011.

[12] de Werra, D.,"Some models of graphs for scheduling sports competitions,", *Discrete Appl. Math. 21,47–65,* 1988.

[13] de Werra, D., Ekim, T., Raess, C.," Construction of sports schedules with multiple venues,", *Discrete Appl. Math. 154, 47–58,* 2006.

[14] Ikebe, Y.T., Tamura, A.,v" On the existence of sports schedules with multiple venues,", *Discrete Appl.Math. 156, 1694–1710,* 2008.

[15] Lamken,E.R.,Vanstone, S.A.," Balanced tournament designs and related topics,*" Discrete Math. 77, 159–176,* 1989.

[16] Nemhauser, G., Trick,M.," Scheduling a major college basketball conference,*" Oper. Res. 46, 1–8,* 1998.

[17] Russell, R.A., Urban, T.L.," Aconstraint-programming approach to themultiple-venue sport-scheduling problem," *Comput. Oper. Res. 33, 1895–1906,* 2006.

[18] Urban, T.L., Russell, R.A.," Scheduling sports competitions on multiple venues*," European J. Oper. Res. 148, 302–311*, 2003.

[19] LeRoy B. Beasley, Richard A. Brualdi, and Bryan L. Shader," *Combinatorial orthogonality, Combinatorial and Graph-Theoretical Problems in Linear Algebra," Discrete Math. 77, 159–176*, 1993.

[20] Richard A. Brualdi, Shmuel Friedland, and Victor Klee, eds.," The IMA Volumes in Mathematics and its Applications," *vol. 50, Springer- Verlag, New York, 1993, pp. 207{218*.

[21] Lowell W. Beineke and K. B. Reid, " Tournaments, Selected Topics in Graph Theory," *Academic press,New york, pp. 169{204.,* 1978.

[22] Alfred Brauer and Ivey C. Gentry," On the characteristic roots of tournament matrices," *Bulletin of the American Mathematical Society 74 (1968), 1133{1134,* 1968.

[23] Ezra Brown and K.B. Reid," Doubly regular tournaments are equivalent to skew hadamard matrices*," Journal of Combinatorial Theory, Series A 12 (1972), 332{338.198,* 1972

[24] Richard A. Brualdi, Rachel Manber, and Je_ery A. Ross," On the minimum rank of regular classes of matrices of zeros and ones," *Journal of Combinatorial Theory, Series A 41 (1986), 32{49*, 1986.

[25] D. deCaen, D.A. Gregory, S.J. Kirkland, and N.J. Pullman, " Algebraic multiplicity of the eigen values of a tournament matrix, Linear Algebra and its Applications," *169, 179{193.,* 1992.

[26] Faun C. C. Doherty, J. Richard Lundgren, and Daluss J. Siewert," Biclique covers and partitions of bipartite graphs and digraphs and related ranks of (0; 1)-matrices," *Congressus Numerantium 136 (1999), 73{96,* 1999.

[27] Carolyn Eschenbach, Frank Hall, Rohan Hemasinha, Stephen J. Kirkland, Zhongshan Li, Bryan L. Shader, Je_ery L. Stuart, and James R.Weaver," On almost regular tournament matrices, Linear Algebra and its Applications," *306 (2000), 103{121.199,* 2000.

[28] Bryan Shader, " On tournament matrices, Linear Algebra and its Applications," *162-164 (1992), 335{368., 1992*.

[29] Daluss J. Siewert, " Biclique covers and partitions of bipartite graphs and digraphs and related ranks of f0; 1g-matrices," Ph.D. thesis, *University of Colorado at Denver, Denver, Colorado*, May 2000.

[30] J. H. vanLint, " f0; 1; _g distance problems in combinatorics, Surveys in Combinatorics," 1985 (Ian Anderson, ed.), *London Mathematical Society Lecture Note Series, vol. 103, Cambridge University Press, Cambridge, pp. 113{135.,* 1985.

[31] Bing JIANG, Yongmin MU, Zhihua Zhang, "Research of Optimization Algorithm for Path-BasedRegression Testing Suit: Proceedings of Second *International Workshop on Education Technology and Computer Science," IEEE, DOI 10.1109/ETCS.2010.365*, 2010.

[32] Leung H K N, White L, "Insights into regression testing: Proceedings of International Conference on Software Maintenance," LOS Alamipos: *IEEE Computer Society, pp. 60-69.,* 1989.

[33] Rasmus V. Rasmussen, "Scheduling a triple round robin tournament for the best Danish soccer league," *Department of Operations Research, University of Aarhus, Ny Munkegade, Building 1530, 8000 Aarhus C,* Denmark.

[34] **Martin Henz**, "Scheduling a Major College Basketball: Proceedings of national conference on Computer Science," *School of Computing, National University of Singapore*, 1999.

[35] Robert Baumann, Victor A. Matheson and Cara A. Howe, "Anomalies in Tournament Design," in *Journal of Quantitative Analysis in Sports, Volume 6, Issue 2, Pages –, ISSN (Online) 1559-0410, DOI: 10.2202/1559-0410.1233,* April 2010.

List of Papers

Published

[1] Trinetra Kumar Pathak and Vinod K. Bhalla, " Optimization of Hamilton Path Tournament Schedule (Paper ID: IJARCET-6936)" *International Journal of Advanced Research in Computer Engineering & Technology*, Volume 2 Issue 6 of (ISSN: 2278-1323), http://ijarcet.org/wp-content/uploads/VOLUME-2-ISSUE-6-2091-2094.pdf , June 2013.

Communicated

[2] Trinetra Kumar Pathak and Vinod K. Bhalla, " PL Tournament Schedule using Backtracking Approach, " *International Journal of Foundations of Computer Science*.