*Project Report*

On

**Time table generator, knight’s tour problem**

Submitted by

**Mounika Mukkamalla 160001036**

**Neha Nagendra 160001042**

Computer Science and Engineering

2nd year

Under the Guidance of

**Dr. Kapil Ahuja**



Department of Computer Science and Engineering

Indian Institute of Technology Indore

Spring 2017

Remaining report can be organized as-

1. **Introduction**

Our project is Time table generator and knight’s tour problem analysis. We chose this topic because it drew our attention and was a good way to apply the algorithms that we learnt in the course.

Our project time table generator is a modification of graph coloring algorithm. While solving knight’s tour problem we came across many algorithms which are there in our course such as Warnsdorff’s algorithm, Hamiltonian path, divide and conquer approach. Overall this project gave us a good practical experience of our course.

1. **Algorithm Analysis**
2. **Time table generator**

We schedule the timetable in such a way that there will be no student who has two exams at the same time. This problem is similar to a graph coloring problem which we solved using greedy algorithm. The result of graph coloring will be the set of nodes having a same color. We can have those exams simultaneously which have same color.

1. **Knight’s tour problem**

A knight's tour is a sequence of moves of a knight on a chessboard such that the knight visits every square only once. The knight's tour problem is the mathematical problem of finding a knight's tour. We will analyze this problem applying 2-3 algorithms. It is a special case of Hamiltonian path. (this section may contain your problem definition and analysis of available algorithms for your problem)

**Existence of the path**

For any m × n board with m ≤ n, a closed knight's tour is always possible unless one or more of these three conditions are met:

m and n are both odd (or) m = 1, 2, or 4 (or) m = 3 and n = 4, 6, or 8.

**Brute-force algorithm:**

In this algorithm we check all the possible paths and see which path leads to completion of the tour. This isn’t a polynomial time algorithm. The complexity of this algorithm will be exponential. It is useful when board size is small.

**Divide and conquer algorithms:**

By dividing the board into smaller pieces, constructing tours on each piece, and patching the pieces together, one can construct tours on most rectangular boards in linear time - that is, in a time proportional to the number of squares on the board.

**Warnsdorff's rule:**

Warnsdorff's rule is a heuristic for finding a knight's tour. The knight is moved so that it always proceeds to the square from which the knight will have the fewest onward moves. When calculating the number of onward moves for each candidate square, we do not count moves that revisit any square already visited.

1. **Algorithm design**

**Time table generator**

We made an exam time table generator using graph coloring approach.

* **Starting with input phase** :

We make a graph using adjacency list for colouring purpose. There will be 2 kinds of inputs. First one being naming of the nodes in our graph. The user inputs course names which become nodes of our graph. 2nd one being the edges between the nodes. If any two courses have some students in common they will form an edge in the graph. User inputs course names and no. of students common.

* **The algorithm phase:**

With all the inputs with us we start our graph coloring process.

***pseudo code:***

for every node i :

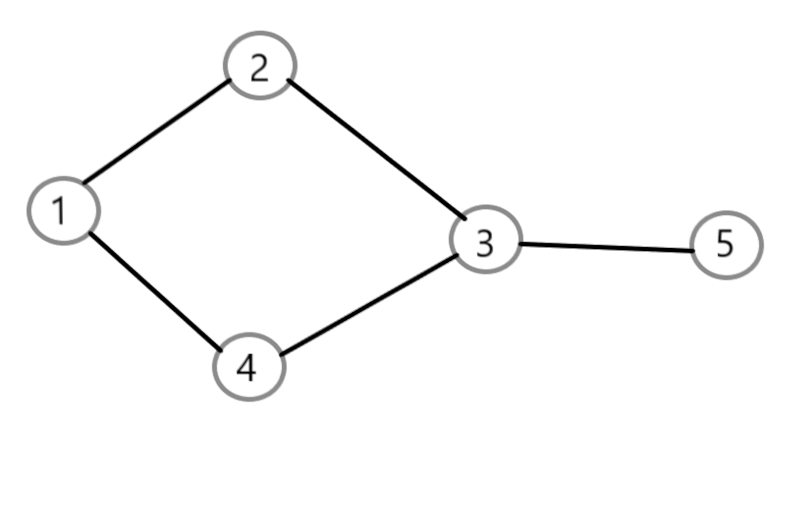
check nodes j=1 to i-1

if neither i and j don’t have an edge nor are linked indirectly ------------ 1

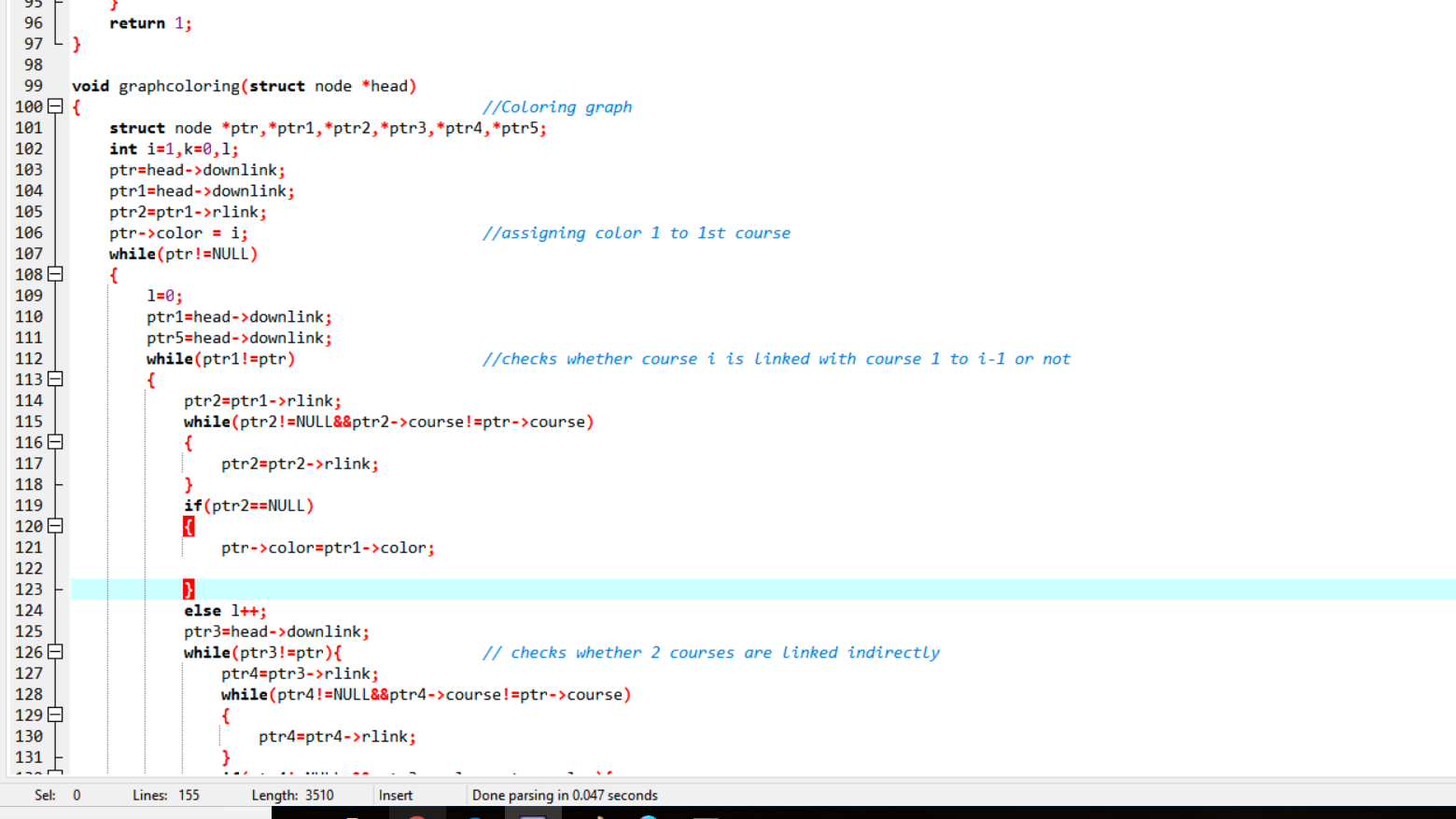
color(i)=color(j)

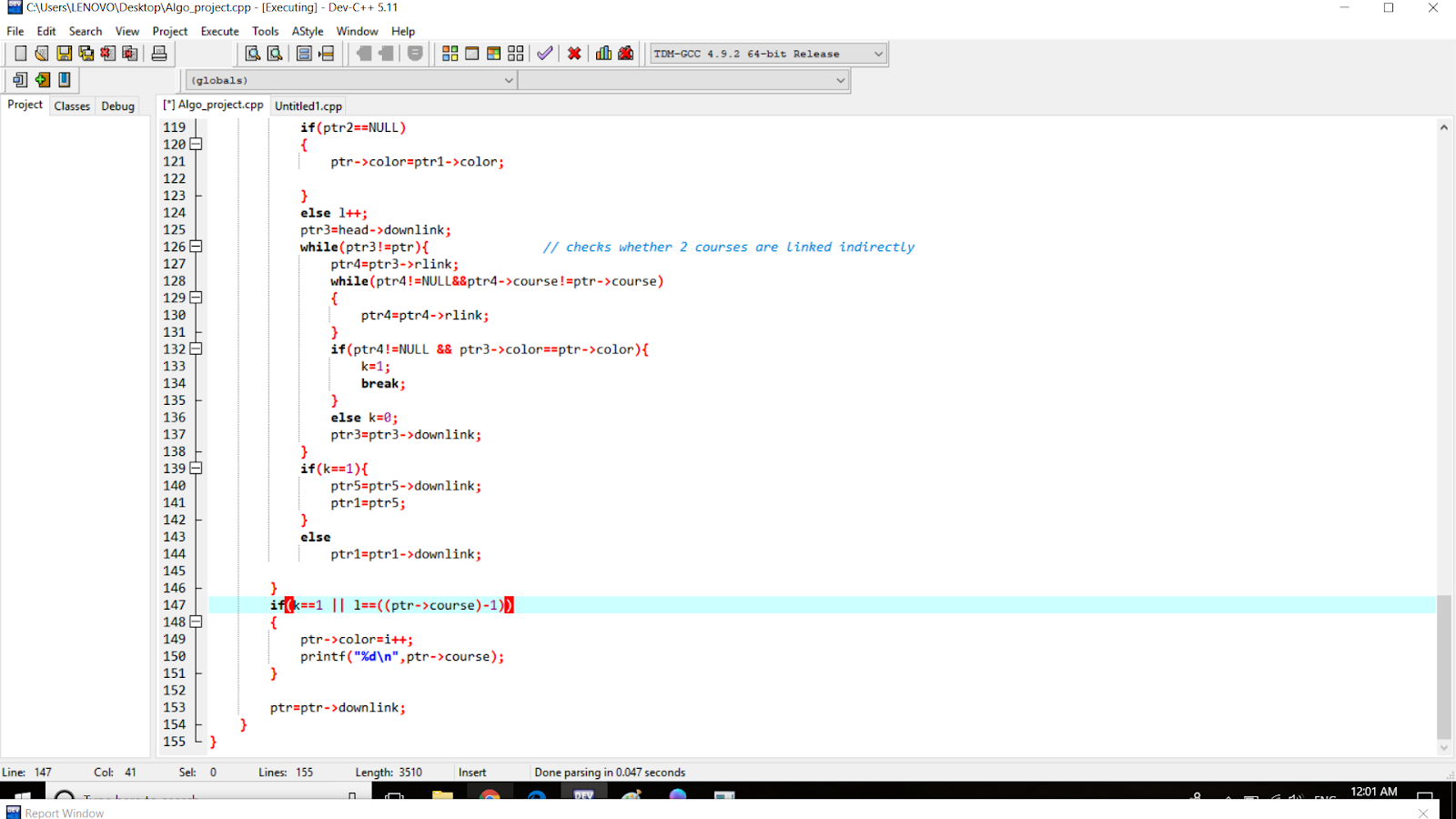
Linked indirectly meaning:

In the graph beside node 5 isn’t directly connected to node 1, and hence node 5 should get colour of node 1. But as we see node 3 is connected to node 5 and our algo in the same way gives colour of 1 to 3. this is indirect linking. 3 and 5 can’t have same colour. So we check such cases also.

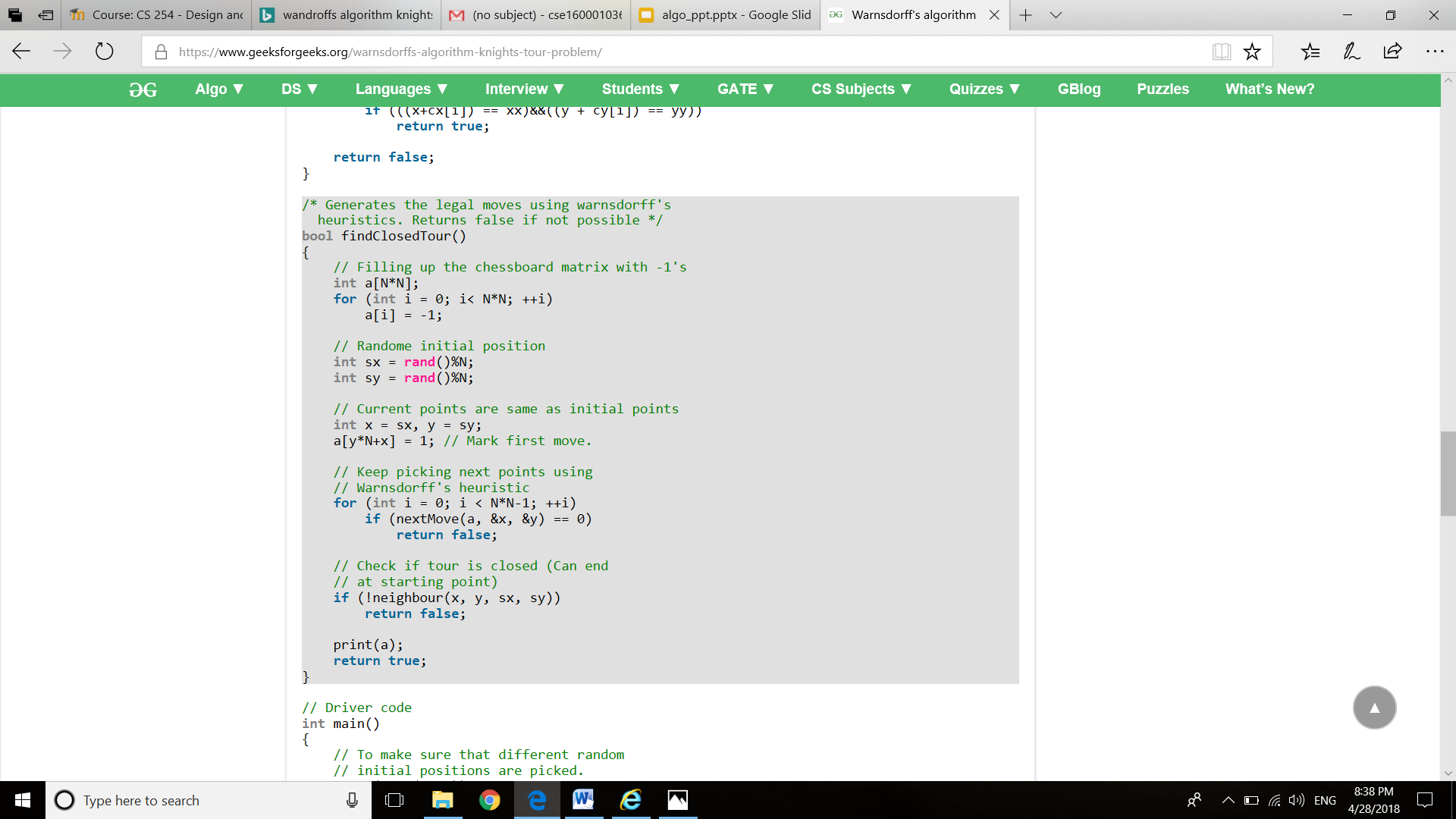


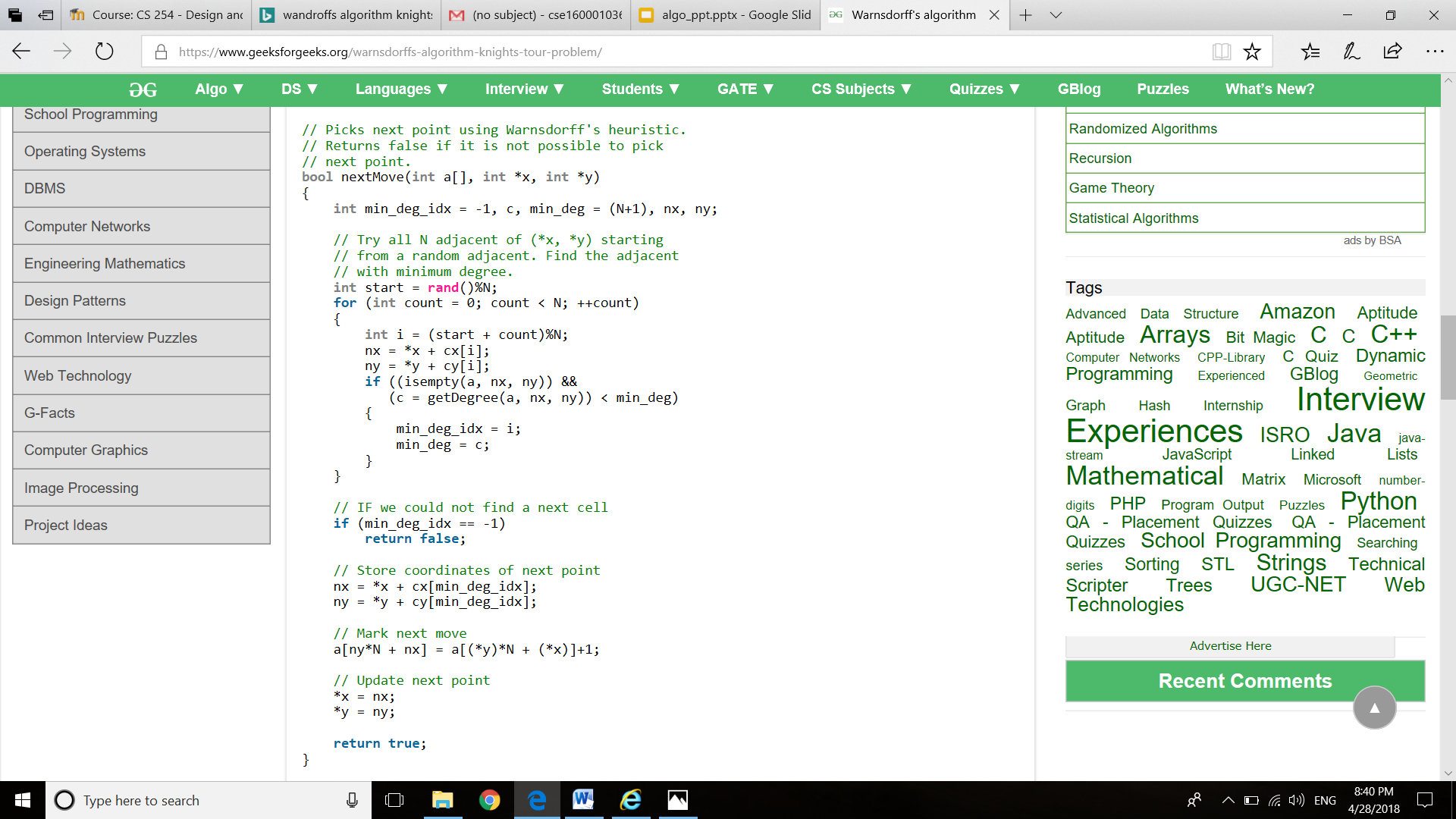
Screenshots of our code:





Knight’s tour problem

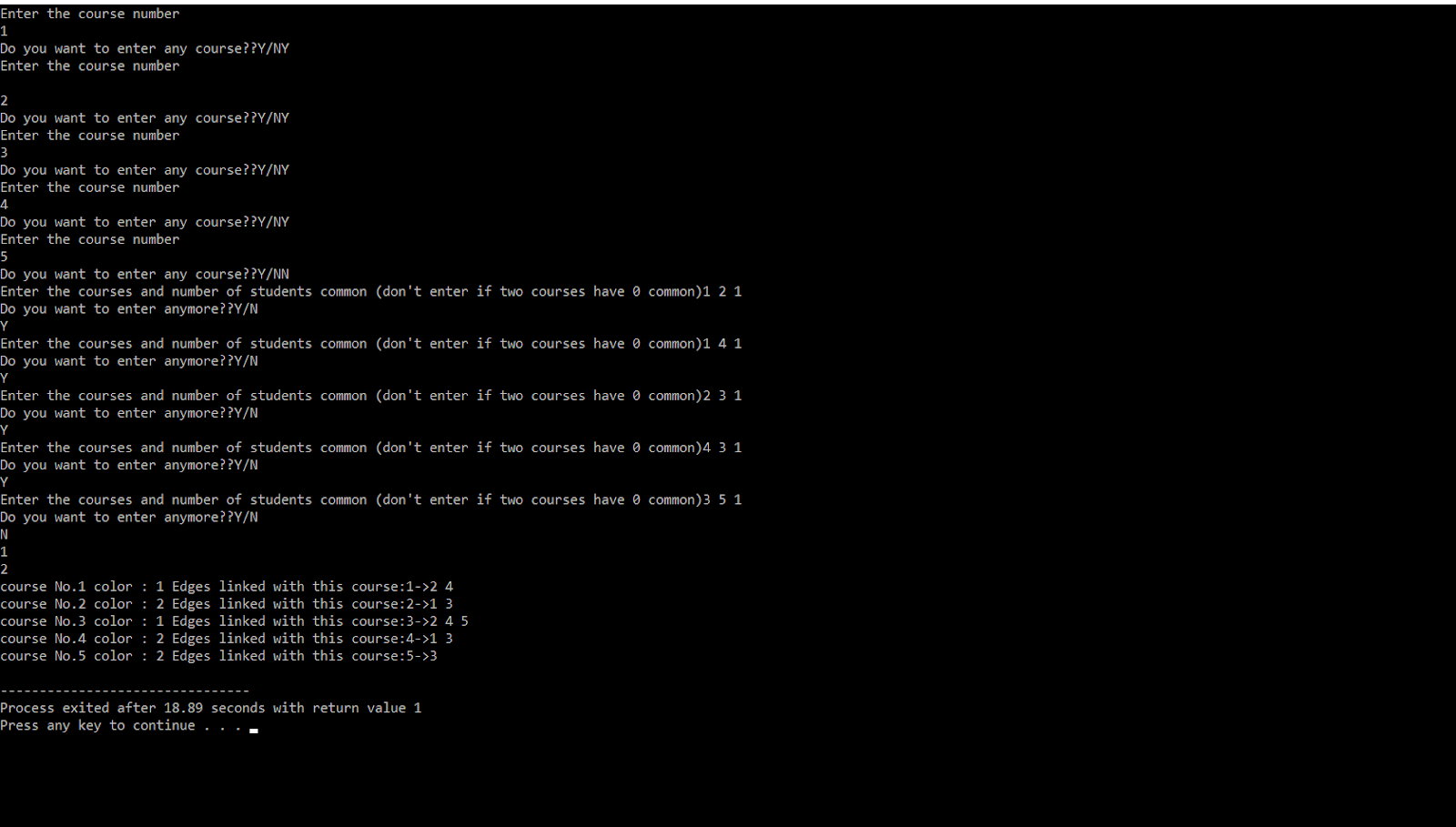




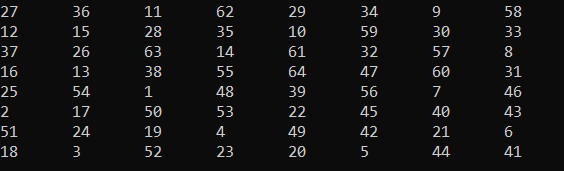
1. **Implementation and Results**

Screenshot of our results:

Time table generator :



Warnsdorff’s Algorithm



1. **Conclusions and Future Work**

Graph coloring algorithm complexity – O(n4)

Knight’ tour :

Brute force – O(2n)

Divide and conquer – O(n)

Warnsdroff’s Algorithm – O(n)

We will try to make the graph coloring algorithm more efficient.

1. **References**

www.geeksforgeeks.com