COS 485 — Homework 5

Samuel Barton Benjamin Montgomery

April 3, 2017

Problem 1

In this problem, we are asked to do out the N-queens problem by hand with N=9. Below is a table containing our solution. For each queen we have assigned a color as follows:

queen 1 — red
queen 2 — blue
queen 3 — green
queen 4 — yellow
queen 5 — cyan
queen 6 — teal
queen 7 — magenta
queen 8 — olive

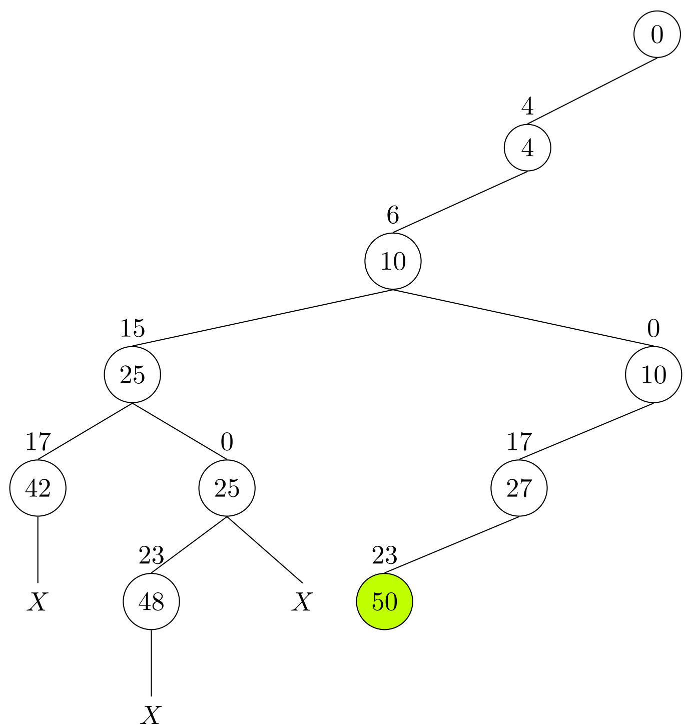
·	·	Q_1	·	·	·	·	·	·
·	x_1	x_1	x_1	·	·	·	·	Q_2
x_1	Q_3	x_1	·	x_1	·	·	x_2	x_2
x_3	x_3	x_1	·	·	x_1	x_2	Q_4	x_2
Q_5	x_3	x_1	x_3	·	x_2	x_1	x_4	x_2
x_5	x_3	x_1	Q_6	x_2	x_4	·	x_1	x_2
x_5	x_3	x_1	x_2	x_4	x_3	Q_7	x_4	x_1
x_5	x_3	x_1	x_4	Q_8	x_6	x_3	x_4	x_2
x_5	x_2	x_1	x_6	x_5	x_8	x_6	x_3	x_2

In the above table we demonstrate the use of the Monte Carlo approach to generate an estimate of the total cost to solve the N-queens problem with N=9. In the below table we give the row sums for all of the options, and then the total sum which represents the overall cost to solve the problem:

Row #	**Choices**	**Cost**
1	9	9
2	6	$9 \cdot 6$
3	4	$9 \cdot 6 \cdot 4$
4	3	$9 \cdot 6 \cdot 4 \cdot 3$
5	2	$9 \cdot 6 \cdot 4 \cdot 3 \cdot 2$
6	2	$9 \cdot 6 \cdot 4 \cdot 3 \cdot 2 \cdot 2$
7	1	$9 \cdot 6 \cdot 4 \cdot 3 \cdot 2 \cdot 2 \cdot 1$
8	1	$9 \cdot 6 \cdot 4 \cdot 3 \cdot 2 \cdot 2 \cdot 1 \cdot 1$
9	0	0
Total		**9999**

Problem 2

In this problem we demonstrate the use of "The Backtracking Algorithm for the Sum of Subsets problem" from Section 5.4 in the text. We utilized the algorithm to find the combination of the elements in the set $\{4,6,15,17,23,26,31\}$ that sum to 50.



As requested, we are only showing the path to the first combination that sums to 50.

Problem 3

In this problem we explain a way the algorithm in Exercise 5.19 could be implemented.

```
Let there be an array where we will store the colors we create

Let there be an adjacency list for this graph

Let there be a count of uncolored nodes

for each node in the adjacency list
    if the count of uncolored nodes is zero
        stop
    if the node has not been colored
        pick a new color
        add that color to the array of possible colors
        color that index with the new color
        decrement the count of uncolored nodes
        for each node in the adjacency list
            if this node is not adjacent to any nodes with the current color
                color this node
                decrement the count of uncolored nodes
```

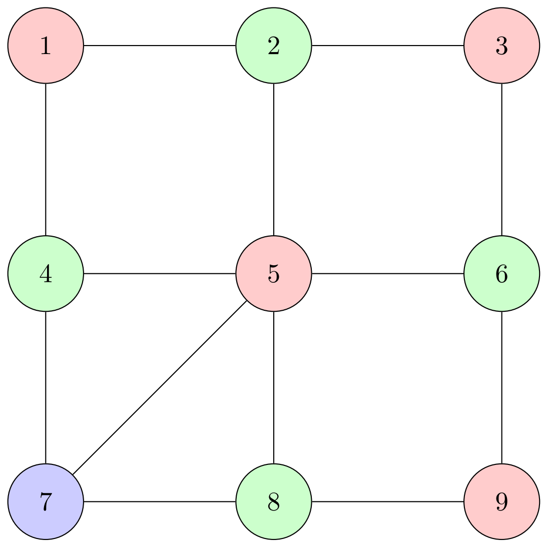
Using the above implementation, the algorithm would take the following amount of time for an arbitrary graph with N nodes.

The outermost loop will iterate N times in the worst case, thus in the worst case we will have N colors. This would happen if we were handed a completely connected graph. Since we go through each node in the graph, and for each node we visit every other node in the graph, we have, based on experience, an O(N^2) algorithm.

Since we wish for a more instructive analysis of this algorithm, we are going to analyze at a higher level of granularity. Thus, we may note that the number of times we go through the outer loop is dependent on the number of colors needed to completely color the graph. The number of colors needed is dependent on the level of connectedness in the graph. Thus, if it takes K colors to completely color the graph, then our algorithm is $\Theta(\mathrm{N} \cdot K)$.

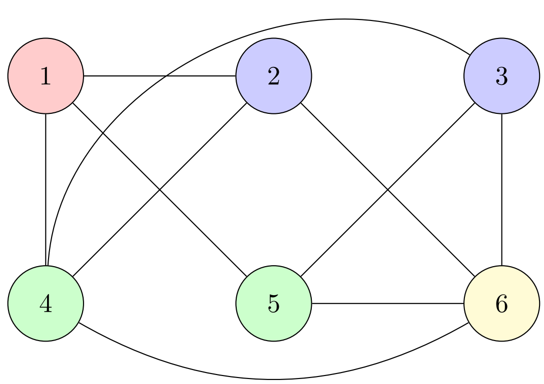
Problem 4

In this problem we are asked to demonstrate the use of our greedy algorithm from Problem 3 to color the graph shown below. Interestingly, the algorithm does indeed produce an optimal coloring for the graph as there is no possible way to get a coloring with only two colors given the extra edge between nodes 5 and 7.

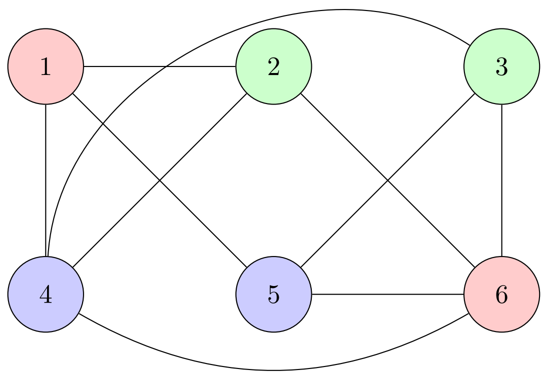


Problem 5

In this problem we demonstrate an arbitrary graph where the greedy algorithm will fail. To illustrate this we will show two colored graphs: one using the greedy algorithm to color the graph, and the other using the "eyesight" algorithm.



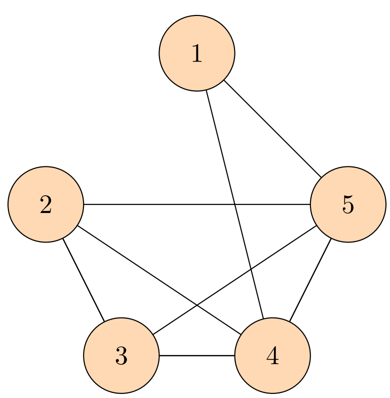
The optimal solution



Problem 6

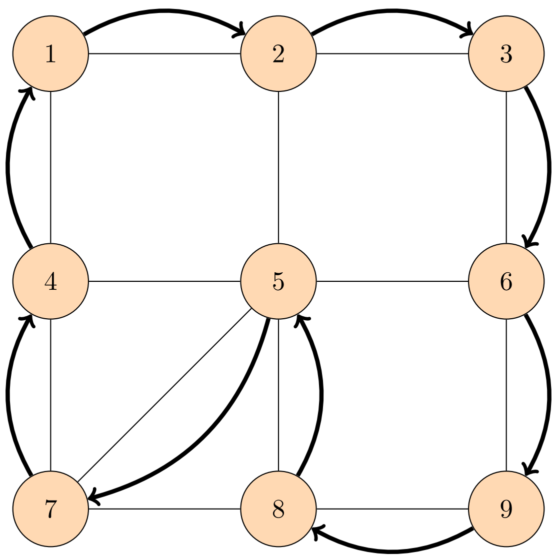
In this problem we are constructing a prototype which can be used to force the 3-Coloring Backtracking Algorithm to take exponential time to discover that it is not 3-Colorable.

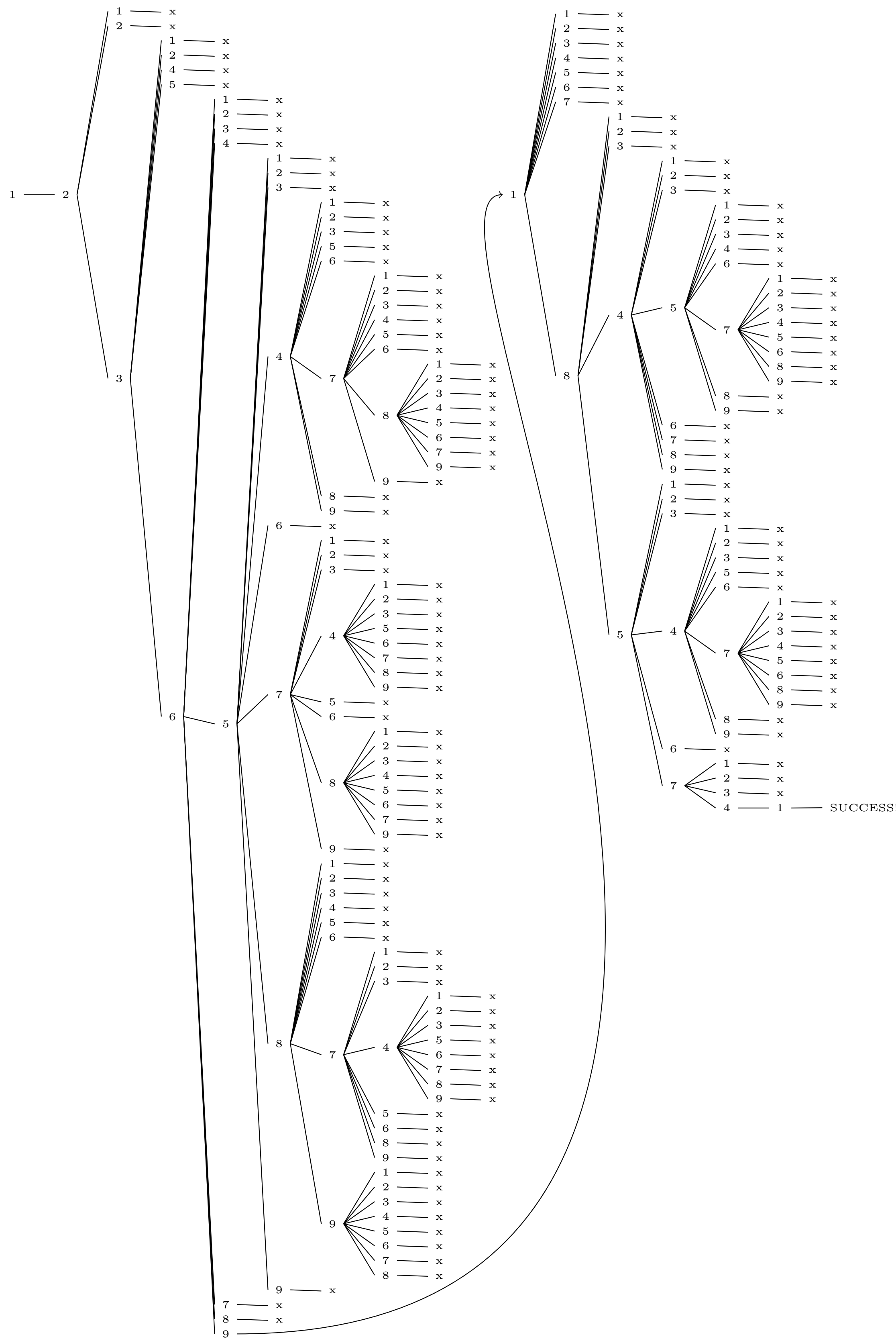
The idea is as follows: Construct a graph of n nodes where V_n and V_{n-1} are connected to every other vertex, and V_{n-2} is connected to V_{n-3}. This graph will take exponential time to solve. Below we provide an example with $n = 5$. This took approximately 125 operations to determine that it was not 3-colorable and so took $T_n = \frac{3^n}{2} \implies T_n \in \Theta\left(3^n\right)$.



Problem 7

In this problem we demonstrate, in great detail, the execution of the Hamiltonian Circuit algorithm on the graph below. On this graph we have denoted the path taken by the circuit. See the next page for the state-space tree for this circuit.




SUCCESS!

Extra Credit Puzzle

Given the scenario described in the homework, namely that we have four people who have to cross a bridge, the bridge can only hold two people, and there is a flashlight which each group of 1 or 2 must carry with them. The group must all be across within 17 minutes. Each person will take a different amount of time to cross the bridge once:

Son	1 min
Father	2 min
Grandfather	5 min
Great-grandfather	10 min

The father and son cross	2 min
The son returns	1 min
The grandfather and great-grandfather cross	10 min
The father returns	2 min
The father and son cross the bridge as the flashlight grows dim	2 min
Total time	17 min