

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

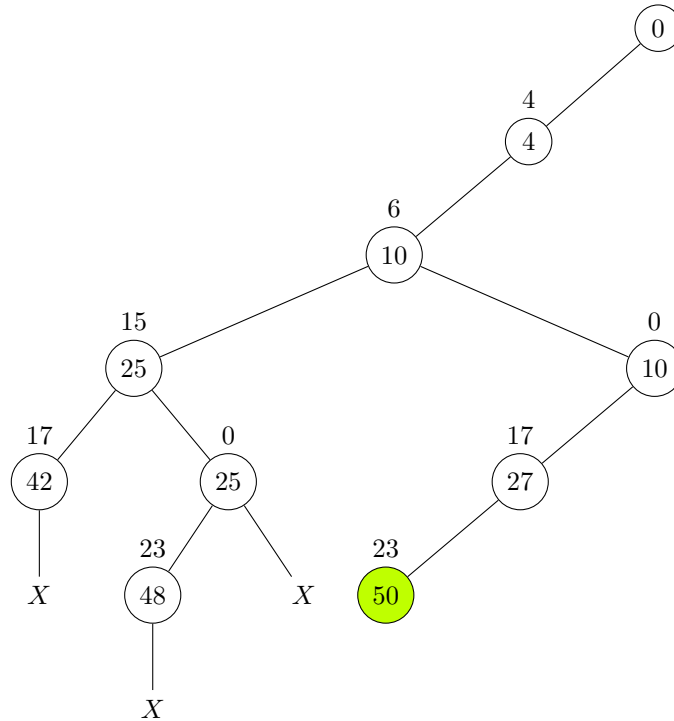
.	.	Q ₁
.	x ₁	x ₁	x ₁	Q ₂
x ₁	Q ₃	x ₁	.	x ₁	.	.	x ₂	x ₂
x ₃	x ₃	x ₁	.	.	x ₁	x ₂	Q ₄	x ₂
Q ₅	x ₃	x ₁	x ₃	.	x ₂	x ₁	x ₄	x ₂
x ₅	x ₃	x ₁	Q ₆	x ₂	x ₄	.	x ₁	x ₂
x ₅	x ₃	x ₁	x ₂	x ₄	x ₃	Q ₇	x ₄	x ₁
x ₅	x ₃	x ₁	x ₄	Q ₈	x ₆	x ₃	x ₄	x ₂
x ₅	x ₂	x ₁	x ₆	x ₅	x ₈	x ₆	x ₃	x ₂

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 · 6
3	4	9 · 6 · 4
4	3	9 · 6 · 4 · 3
5	2	9 · 6 · 4 · 3 · 2
6	2	9 · 6 · 4 · 3 · 2 · 2
7	1	9 · 6 · 4 · 3 · 2 · 2 · 1
8	1	9 · 6 · 4 · 3 · 2 · 2 · 1 · 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 $\Theta(N \cdot K)$.