CSCI803 Assignment

Yao Xiao
SID 2019180015

November 19, 2020

1 Problem 1

```
# Set N
N = 8
ld = [0] * 30
rd = [0] * 30
cl = [0] * 30

def printSolution(board):
    for i in range(N):
        for j in range(N):
            print(board[i][j], end = " ")
        print()

def solveNQUtil(board, col):

    if (col >= N):
        return True

    for i in range(N):

        if ((ld[i - col + N - 1] != 1 and
             rd[i + col] != 1) and cl[i] != 1):

            board[i][col] = 1
            ld[i - col + N - 1] = rd[i + col] = cl[i] = 1

            if (solveNQUtil(board, col + 1)):
                return True

            board[i][col] = 0 # BACKTRACK
            ld[i - col + N - 1] = rd[i + col] = cl[i] = 0

    return False

def solveNQueue():
    board = [[0, 0, 0, 0, 0, 0 ,0, 0],
```

```
                [0, 0, 0, 0, 0, 0 ,0, 0],
                [0, 0, 0, 0, 0, 0 ,0, 0],
                [0, 0, 0, 0, 0, 0 ,0, 0],
                [0, 0, 0, 0, 0, 0 ,0, 0],
                [0, 0, 0, 0, 0, 0 ,0, 0],
                [0, 0, 0, 0, 0, 0 ,0, 0],
                [0, 0, 0, 0, 0, 0 ,0, 0],
                ]
    if (solveNQUtil(board, 0) == False):
        print("Solution does not exist")
        return False
    printSolution(board)
    return True

solveNQueue()
```

```
▲ ~/Desktop/Lab7 py3 task1.py
1 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0
0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 1
0 1 0 0 0 0 0 0
0 0 0 1 0 0 0 0
0 0 0 0 0 1 0 0
0 0 1 0 0 0 0 0
```

2 Problem 2

```
#include <iostream>
#include "stdc++.h"

using namespace std;
#define N 4

struct Node {
    Node *parent;
    int pathCost;
    int cost;
    int machineID;
```

```
12      int taskID;
13      bool assigned[N];
14  };
15
16  struct comp {
17      bool operator()(const Node *lhs,
18                      const Node *rhs) const {
19          return lhs->cost > rhs->cost;
20      }
21  };
22
23  Node *newNode(int x, int y, bool assigned[],
24                Node *parent) {
25      Node *node = new Node;
26
27      for (int j = 0; j < N; j++)
28          node->assigned[j] = assigned[j];
29      node->assigned[y] = true;
30
31      node->parent = parent;
32      node->machineID = x;
33      node->taskID = y;
34
35      return node;
36  }
37
38  int calculateCost(int costMatrix[N][N], int x,
39                    int y, bool assigned[]) {
40      int cost = 0;
41
42      bool available[N] = {true};
43
44      for (int i = x + 1; i < N; i++) {
45          int min = INT_MAX, minIndex = -1;
46
47          for (int j = 0; j < N; j++) {
48              if (!assigned[j] && available[j] &&
49                  costMatrix[i][j] < min) {
50                  minIndex = j;
51
52                  min = costMatrix[i][j];
53              }
54          }
55
56          cost += min;
57
58          available[minIndex] = false;
59      }
60
61      return cost;
62  }
```

```
63
64  void printAssignments(Node *min) {
65      if (min->parent == NULL)
66          return;
67
68      printAssignments(min->parent);
69      cout << "Assign Machine " << char(min->machineID + '1')
70           << " to Task " << min->taskID << endl;
71
72  }
73
74  // Finds minimum cost using Branch and Bound
75  int findMinCost(int costMatrix[N][N]) {
76      priority_queue<Node *, std::vector<Node *>, comp> prq;
77
78      bool assigned[N] = {false};
79      Node *root = newNode(-1, -1, assigned, NULL);
80      root->pathCost = root->cost = 0;
81      root->machineID = -1;
82
83      prq.push(root);
84
85      while (!prq.empty()) {
86          Node *min = prq.top();
87
88          prq.pop();
89
90          int i = min->machineID + 1;
91
92          if (i == N) {
93              printAssignments(min);
94              return min->cost;
95          }
96
97          for (int j = 0; j < N; j++) {
98              if (!min->assigned[j]) {
99                  Node *child = newNode(i, j, min->assigned, min);
100
101                 child->pathCost = min->pathCost + costMatrix[i][j];
102
103                 child->cost = child->pathCost +
104                               calculateCost(costMatrix, i, j, child
                                  ->assigned);
105
106                 prq.push(child);
107             }
108         }
109     }
110 }
111
112 int main() {
```

```
113     int costMatrix[N][N] =
114             {
115                     {13, 4,  7, 6},
116                     {1,  11, 5, 4},
117                     {6,  7,  2, 8},
118                     {1,  3,  5, 9}
119             };
120
121
122     cout << findMinCost(costMatrix)
123          << " is optimal min cost!";
124     return 0;
125 }
```

```
/Users/december/Desktop/task2/cmake-build-debug/untitled
Assign Machine 1 to Task 1
Assign Machine 2 to Task 3
Assign Machine 3 to Task 2
Assign Machine 4 to Task 0
11 is optimal min cost!
Process finished with exit code 0
```