1 Introduction and Objective

This is the last official tutorial of CS2010 this semester...

Tuesday, 10 November 2015 is Deepavali public holiday and the only Monday TA, Nathan, is also not available on Monday, 09 November 2015 as he attends ACM ICPC Jakarta 2015.

Therefore, Steven has decided to convert the 3 Monday sessions of Monday, 09 November 2015 as 3-hours of public consultation at COM1-0207 where students from T9/10/11 or any other Tuesday tutorial groups can drop by to ask questions related to CS2010. You can ask for solutions of past exam papers, provided the one who ask has shown Steven non-trivial attempt at the question.

In this last tutorial 10, we will do four important things: WQ2 debrief (max 15 minutes), discuss two harder Dynamic Programming (DP) questions, discuss Subtask A of PS6 (short one), and end CS2010 tutorial sessions with a class photo (Jonathan’s tutorial groups will do this on Week11 instead).

DP can be challenging to master but it is an important algorithm design strategy as it can solve certain problems much more efficiently than using Complete Search. Do not hesitate to ask the teaching staffs if you encounter difficulties with this topic. You will learn more details about DP in the next core module (for CS students): CS3230 and in an optional module CS3233.
2 Tutorial 10 Questions

Written Quiz 2 Debrief

Detailed solutions will only be told to your tutor on Saturday, 31 October 2015 after Written Quiz 2.

Summary of grading feedback:

1. Section A, mostly OK except question 1 and/or 2 which are related to Section C.2. Read the modal answer carefully.

2. Section B

(a) B.1. Only dfs-avoid-odd(7) has many students mysteriously answered 7 AND 5... If we have used strict marking scheme like in Online Quiz, this is a very fatal mistake...

(b) B.2. Given such question, try not to brute force (draw all), think first...

(c) B.3. If the source vertex reaches a negative weight cycle and then spread out to other vertices, those other vertices will have $D[v] = -\infty$ too.

(d) B.4. Without the early announcement that vertex 0 has to be connected with the rest, the answer is trivial, just don’t connect vertex 0 with anything.

(e) B.5. Same, without the early announcement that the graph has to be connected, leaving this question unanswered will also get full marks...

3. Section C

(a) C.1. Do not forget what have been discussed back in Written Quiz 1. You cannot work on a graph that has 100000 vertices and $100000^2$ edges directly. You have to utilize your feeling that any vertex $i$ should be likely connected to any other vertex $j$ in near complete graph. Then think of a case where it is not so, given that you can only delete up to $k \leq 7$ edges. Remember that two vertices can be connected via a direct edge or via a path.

(b) C.2.1. Do not lose marks from easier subtasks like this. Do not forget to delete the only $K = 1$ deleted vertex first before running BFS/DFS from any unvisited vertices (not necessarily neighbors of the deleted vertex, as the deleted vertex may have no neighbor :O).

(c) C.2.2. Due to so many wrong answers, a naive $K$ calls of C.2.1. solution that results in $O(K \times (V + E))$ time complexity already gets half (8) partial marks. The given Figure 7 happens to be a tree, but the question is asked on general graph. The $K$ vertices are deleted one by one and the numbers of connected components have to be reported immediately after each deletion, not at the end. Using Tarjan’s algorithm for finding articulation points (cut vertices) and/or bridges are not correct if only done once as it will be able to identify the cut vertices/bridges that may increase number of CCs after removal, but it cannot count how many. At the end, only 5/174 takers of WQ2 solved this last question fully and they deserved the 10 points gap.
Two Harder DP Question

Q1. A subset of vertices $S \subset V$ is an Independent Set (IS) of graph $G = (V, E)$ if there is no edge between any pair of them. For instance, in Figure 1, vertices $\{1, 5\}$ form an IS, but vertices $\{1, 4, 5\}$ do not. The largest IS in Figure 1 has size 3, e.g. $\{2, 3, 6\}$, $\{1, 4, 6\}$, or $\{2, 4, 6\}$ (this list is not exhaustive). Give a DP solution for finding the size of the largest IS of $G$ when $G$ is a tree. Hint: Attach one extra Boolean parameter taken/not taken to each vertex and proceed from there. Review Lecture 11 about adding extra parameter to convert a non DAG into a DAG.

![Figure 1: A Tree](image)

Here we have a Boolean choice to pick or not pick a vertex. Also once a vertex is picked, we cannot pick its children (otherwise the set will not be independent). We can formulate the recurrence as follows:

$I(v, 0)$ is the largest IS for the tree rooted at vertex $v$ when $v$ is not taken, while $I(v, 1)$ is the case when vertex $v$ is taken.

If $v$ is not taken, we can choose to take or not take its children and we will take the max among the two options, thus: $I(v, 0) = \sum \text{Max}(I(c, 0), I(c, 1))$, for all children $c$ of $v$.

If $v$ is taken, we definitely cannot choose its children without violating the IS requirement, thus: $I(v, 1) = 1 + \sum I(c, 0)$, for all children $c$ of $v$. (notice +1 because we take vertex $v$).

Base cases:
$I(v, 0) = 0$ for all $v$, where $v$ are leaves in $G$.
$I(v, 1) = 1$ for all $v$, where $v$ are leaves in $G$.

The pseudo code is now simple if we have these recurrences, try to come up with it by yourself. We need $2 \times V$ memo table because we duplicate each vertex of the tree inside this computation DAG. For those vertices with are picked, they have total number of outgoing edges to be $(V - 1)$, while for those vertices which are not picked they have total outgoing edges to be $2 \times (V - 1)$, since you can pick or not pick their children vertices. Total edges is then $3 \times (V - 1)$. Thus total number of operations needed is $O(2 \times (V - 1) + 3 \times (V - 1)) = O(5V - 5) = O(V)$.

NOTE: The underlying problem is not really ‘longest path on DAG’ as one can take multiple branches. It may not have a specific name, so let’s just call it a DP algorithm for finding the size of the Maximum Independent Set on a Tree.
Q2. Please download CS2010 Final Exam paper, 20141-15-S1-final.pdf, and solve a problem titled: Applications (25 marks) - Robot Turtles. Only one of the three sub-questions is DP, but I encourage you to read all, identify which one is DP, and try to solve all sub-questions.

Application 1: Minimum Number of ‘Move Forward’ Card(s) Used (7 marks)

1. What do the vertices and the edges of your Graph represent? (2 marks)
   Vertices: \((row, col)\),
   Edges: Connect two cells \(a\) and \(b\) if they are reachable with one of the 4 possible movements.

2. What is the upper bound of the number of vertices and edges in your Graph? (1 mark)
   There are up to \(8 \times 8 = 64\) vertices in this Graph.
   Each vertex can potentially be connected to 4 other vertices. Thus, up to \(4 \times 64 = 256\) edges.
   A more precise computation: \((6 \times 6 \times 4) + (4 \times 2) + (6 \times 4 \times 3) = 224\) edges.
   That is, on empty \(8 \times 8\) board, the middle \(6 \times 6\) cells have up to 4 edges, the 4 corner cells only have 2 edges, and the remaining \(6 \times 3\) cells on the side only have 3 edges.

3. What is the Graph problem that you want to solve? (2 marks)
   We want to find the shortest path from the source cell (cell that contains the robot turtle \(\text{'R'}\)) to the destination cell (cell that contains the jewel \(\text{'J'}\)) in this undirected unweighted graph.

4. What is the best Graph algorithm to solve this problem and its time complexity? (2 marks)
   The best algorithm is \(O(V + E)\) BFS.
   Using Original/Modified Dijkstra’s/Bellman Ford’s will only get 1 mark.

Application 2: Minimum Number of Card(s) Used (8 marks)

1. What do the vertices and the edges of your Graph represent? (3 marks)
   Vertices: \((row, col, dir)\),
   Edges: Connect two cells \(a\) and \(b\) if they are reachable with one of the 4 possible movements and maintain the direction.

2. What is the upper bound of the number of vertices and edges in your Graph? (1 mark)
   There are up to \(8 \times 8 \times 4 = 256\) vertices in this Graph.
   Each vertex can potentially be connected to 3 other vertices (there are only 3 possible actions).
   Thus, there can be up to \(3 \times 256 = 768\) edges. Tighter bound exists but not necessary. Answers of either 768 or 1024 edges are accepted. Answers significantly lower or significantly higher than this range are not.

3. What is the Graph problem that you want to solve? (2 marks)
   We want to find the shortest path from the source cell (cell that contains the robot turtle \(\text{'R'}\) and faces East) to the destination cell (any cell that contains the jewel \(\text{'J'}\), and the robot can be in any orientation when it reaches this cell) in this directed unweighted graph.
4. What is the best Graph algorithm to solve this problem and it’s time complexity? (2 marks)

The best algorithm is still $O(V + E)$ BFS. Using Original/Modified Dijkstra’s/Bellman Ford’s will only get 1 mark.

Application 3: How Many Ways to Complete the Puzzle? (10 marks)

Steven’s old answer, in C++ (too lazy to convert this to Java) is shown below. The eventual marking scheme used is: 0 (blank :O), 1 (very far from acceptable), 4 (basic ideas like the usage of r,c,dir + F/L/R, 6 parameters, the base case is hitting jewel - count as one event, but miss many details, 7 for in between, 9 for minor mistake (usually about using the forward card but it leads the robot outside grid or bumps into obstacle), and 10 (perfect).)

```cpp
// Robot Turtles
// DP, counting paths in DAG

#include <bits/stdc++.h>
using namespace std;
#define MOD 1000003 // you don't have this in the actual exam
                    // this is to prevent the usage of BigInteger
typedef pair<int, int> ii;

typedef pair<pair<int, int>, pair<int, int> > iipii;

typedef multiset<ii> sii;

int dr[] = { 0, 1, 0,-1}; // E, S, W, N
int dc[] = { 1, 0,-1, 0};
int i, j, F, L, R, sr, sc, tr, tc, memo[8][8][4][18][8][8];
// 8*8*4*18*8*8 = 294912, doable :D
char board[10][10];

ii MoveForward(int row, int col, int dir) {
    int nrow = row+dr[dir], ncol = col+dc[dir];
    if ((nrow < 0 || nrow >= 8 || ncol < 0 || ncol >= 8) || // outside grid
        (board[nrow][ncol] == '#')) // hits an obstacle cell
        return ii(row, col); // no change
    else
        return ii(nrow, ncol); // move
}

int dp(int row, int col, int dir, int F_left, int L_left, int R_left) { // O(N*N*4*F*L*R)
    // how many ways to reach jewel if my robot turtle is at (row, col), facing dir,
    // and I have (F_left, L_left, R_Left) cards left
    if (row == tr && col == tc) // arrive at jewel cell, in any orientation
        return 1;
    else
        return dp(row, col, dir, F_left, L_left, R_left) +
            dp(row+dr[dir], col+dc[dir], dir, F_left-1, L_left-1, R_left-1);
}
```

5
if (memo[row][col][dir][F_left][L_left][R_left] != -1)
    return memo[row][col][dir][F_left][L_left][R_left];
int ans = 0;
if (F_left > 0) { // option 1: issue a Move Forward card if possible
    ii rc = MoveForward(row, col, dir);
    ans += dp(rc.first, rc.second, dir, F_left-1, L_left, R_left) % MOD;
}
if (L_left > 0) // option 2: issue a Turn Left card if possible
    ans += dp(row, col, (dir+3)%4, F_left, L_left-1, R_left) % MOD;
if (R_left > 0) // option 3: issue a Turn Right card if possible
    ans += dp(row, col, (dir+1)%4, F_left, L_left, R_left-1) % MOD;
return memo[row][col][dir][F_left][L_left][R_left] = ans % MOD;

int main() {
    while (scanf("%d %d %d ", &F, &L, &R) != EOF) {
        for (i = 0; i < 8; i++) {
            scanf("%s ", &board[i]);
            for (j = 0; j < 8; j++)
                if (board[i][j] == 'R') { // record the (sr, sc) coordinate of the turtle
                    sr = i; sc = j; }
            else if (board[i][j] == 'J') {
                tr = i; tc = j; }
        }
    }
    memset(memo, -1, sizeof memo);
    printf("%lld\n", dp(sr, sc, 0, F, L, R));
    return 0;
}

Btw, post exam, I realized that I have not actually told the students that in the actual puzzle, the
maximum number of F, L, and R are 18, 8, and 8, respectively. This does not really matter though
as students are expected to express the time complexity of their solution in terms of F, L, and R.
Moreover, one student pointed out that parameter dir can actually be dropped and we can derive
dir from two of the other five parameters... This way, the solution can be 4 times faster. Do you
know how? This technique is beyond CS2010 though although not that hard to think of...

Problem Set 6

Now the tutor will discuss Problem Set 6, Subtask A only (EDIT: A only, B during Lecture 12). The
focus for today is to give everyone a head start for this extremely (EDIT: after PS5, this is no longer
that challenging) challenging last PS of this module. Your Lab TA will add a bit more on Thursday
during Lab Demo 10 and then all the best...
For Subtask A, everything in a small supermarket have to be grabbed/bought. $1 \leq K = N \leq 15$. This is the typical range of Held-Karp’s DP-TSP as outlined in Lecture 11. Backtracking is too slow. Students just need to understand the Lecture 11 material and applied it here. Lab TA will demonstrate the detailed implementation of Held-Karp’s DP-TSP algorithm with bitmask during Lab Demo 10 this week.

For Subtask B, attend Lecture 12.

For Subtask C, use your brain :).

For Subtask D, leave it alone, it is not for CS2010 level... unless you are taking CS2010R :O...

Class Photo

As this is the last official tutorial class of CS2010, tutors are recommended to take class photos and post the photos in our CS2010 Facebook Group (Jonathan’s tutorial groups will do this on Week11 instead).

All the best for your final exam of this module and of your other modules.