CS2020: Data Structures and Algorithms (Accelerated)

Problems 1–3

For Week12

1 Lecture 19 Material - APSP/Floyd Warshall's

1.1 Verifying Your Understanding of Floyd Warshall's Algorithm

As usual, let's start your DG this week with something light. This time, we will deal with matrices than drawings. Suppose you have a small directed graph of $V = 4$ vertices. This graph is stored in the following Adjacency Matrix:

```
 |0 1 2 3
---------
0|0 5 0 9
1|0 0 1 4
2|7 0 0 1
3|0 0 0 0
```

Together, with your DG, run Floyd Warshall's algorithm step by step from $k = 0$ (only vertex 0 is allowed as intermediate vertex). Do you have to change some values in this matrix? Then $k = 1$ (now vertex 0 and 1 are allowed as intermediate vertices), $k = 2$, and finally $k = 3$. Note that even with this small graph of size $V = 4$, we have to do $4^3 = 64$ operations... How lucky we are that we have computers to help us...

Now, after running Floyd Warshall's algorithm *manually*, answer these observational questions: 1). What is the 'topological order' used by this Dynamic Programming algorithm. 2). When you have computed `AdjMatrix[i][j]` and move on to the next cell `AdjMatrix[i][j+1]` (or to `AdjMatrix[i+1][0]` if `j == 3` – last column), do you ever have to 'backtrack' and re-do previous computation?

1.2 An Assumption for Motivating Problem 1

What if we assume that the given graph is always a tree? Can you come up with a more efficient algorithm than $O(V^3)$ Floyd Warshall's? If you can, present your solution and discuss the benefits (or maybe, some disadvantages) of your solution.

1.3 A Better Solution for Motivating Problem 3?

Do you think the motivating problem 3 shown in Lecture19 – finding the best meeting point between two sources in a non-negative weighted graph – can be solved in a better, more efficient way than using $O(V^3)$ Floyd Warshall's algorithm? If you can, present your solution and discuss the benefits (or maybe, some disadvantages) of your solution.

1.4 Can Floyd Warshall's Algorithm Detect Negative Cycle?

Bellman Ford's algorithm can do this. Original Dijkstra's algorithm cannot do this. Modified Dijkstra's algorithm like the one presented in Lecture17 can do this, albeit slower than the Bellman Ford's version. Now, how about Floyd Warshall's algorithm? Can it be used to detect negative cycle in a weighted graph? If possible, how?

1.5 Solve This Arbitrage Problem with Floyd Warshall's

Arbitrage is the use of discrepancies in currency exchange rates to transform one unit of a currency into more than one unit of the same currency. For example, suppose that 1 US Dollar buys 0.5 British pound, 1 British pound buys 10.0 French francs, and 1 French franc buys 0.21 US dollar. Then, by converting currencies, a clever trader can start with 1 US dollar and buy 0.5 * 10.0 * 0.21 = 1.05 US dollars, making a profit of 5 percent.

Given a foreign exchange matrix, determine with the help of Floyd Warshall's algorithm whether arbitrage is possible or not (`AdjMatrix[i][i] = 0`, as there is no point doing that). Hint: change the default `relaxation` operation to `multiplication` and then this problem becomes similar to the problem discussed just before this one.

```
Index 0 = US dollar, 1 = British pound, 2 = French Franc

Table 1
 |    0     1     2
------------------
0| 0.00  0.50  0.00
1| 0.00  0.00 10.00
2| 0.21  0.00  0.00

Table 2
 |    0     1     2
------------------
0| 0.00  0.50  4.90
1| 1.99  0.00 10.00
2| 0.19  0.09  0.00
```

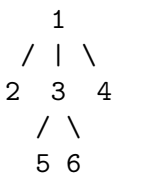
2 Recitation on Week11 - Minimum Vertex Cover on Tree

```
private static int mVC(int v, int flag) { // min Vertex Cover
  int ans = 0;

  if (memo[v][flag] != -1)
    return memo[v][flag];
  else if (leaf[v])
    ans = flag; // 1/0 = taken/not
  else if (flag == 0) { // if v is not taken, we must take its children
    ans = 0;
    for (int j = 0; j < Children.get(v).size(); j++)
      ans += mVC(Children.get(v).get(j), 1);
  }
  else if (flag == 1) { // if v is taken, take the min between
    ans = 1;            // taking or not taking its children
    for (int j = 0; j < Children.get(v).size(); j++)
      ans += Math.min(mVC(Children.get(v).get(j), 1), mVC(Children.get(v).get(j), 0));
  }

  return memo[v][flag] = ans;
}
```

The code above is the working version of the method `mVC(int v, int flag)` as shown in the Recitation on Week11. `leaf[v]` is true if vertex `v` is a leaf, false otherwise; `Children` is an Adjacency List that contains the directed version of the original tree (parent points to its children but children does not have reference to parents). So if the original tree is:



Then `Children` contains:

```
1: 2, 3, 4
2:
3: 5, 6
4:
5:
6:
```

Now, after going through Lecture20, discuss why the space complexity of this DP algorithm is $O(V)$ and the time complexity is also $O(V)$?

3 Lecture 20 Material - Graph Problems Solvable with DP

Last week, you were presented with two UVa online judge problems: one interesting graph modeling exercise which turns out to be a SSSP problem on non-negative weighted graph (UVa 10801) and one problem that combines both SSSP problem on non-negative weighted graph and SSSP problem on unweighted graph (UVa 11635).

This time, you are presented with another three online judge problems. These problems look like graph problems but... see... most (if not all) of the classical graph algorithms that we have learned in Lecture13-17 do not work. You have to devise DP solutions for these problems and you will need knowledge from Lecture20. There is no better way to understand DP other than solving some DP-related problems.

1. UVa 907 - Winterim Backpacking Trip
   http://uva.onlinejudge.org/external/9/907.html
   Is the given graph a 'line graph'? Or is it a tree?
   Isn't this the same as the minimax problem presented in Quiz 2? Or is it not?

2. UVa 10496 - Collecting Beepers
   http://uva.onlinejudge.org/external/104/10496.html
   What is the type of the given graph?
   Isn't this another shortest path problem discussed in Lecture16-17? Or is it not?

3. SPOJ 101 - Fishmonger
   http://www.spoj.pl/problems/FISHER/
   Should we minimize time or minimize toll? Hm...
   Can we use Dijkstra's algorithm to solve this problem?

After reading these problems, three students will answer these questions.

1. What is the graph (the vertices, edges, weighted/unweighted, directed/undirected, etc)?
2. The graph of these problems are *general*. How to convert them into DAGs that are friendly for DP technique? What parameter(s) that you have to add to each vertex?
3. What is the space and time complexity of your DP solution?
   This is an important exercise! Make sure you can analyze DP solutions!
4. Try to sketch a *pseudo code* that implements the chosen data structure and algorithm.
   If possible, write a working Java code and present it during DG :).

4 Discussion on Current Problem Set (PS10)

You are encouraged to spend some time during DG9 to discuss the solutions for problems in PS10. Both problems are from the recent Singapore NOI 2011 and considered hard for high school level in 4.5 hours contest setting. However, we believe that with the knowledge that you have gained in CS2020 are sufficient to solve them.

Note that you are allowed to discuss! You are only prohibited from coding the solution together/copying the solution. Write down the list of collaborators in your solution.