CSC373

Midterm 2 answers and notes

Question 1

a) The DP reccurence is:
$C[j]= 0$ if $j=1$. Otherwise: $C[j] = \min_{1<=I<j} \{C[i]+p_{i,j}\}$

Common mistakes:

1. People basically gave a Bellman-Ford recurrence, solving for every endpoints: $C[i,j]=\min_{i<k<j}\{C[i,k]+C[k,j],p_{i,j}\}$. This is unnecessary since the question explicitly asked for the cost of getting to post j from the **starting** post - 3 points deduction.
2. Some people forgot to give the base case (j=1) - 1 point deduction.

b)

```
Initialize array: C[1 . . . n]
C [1] = 0
for i ← 2 to n  do:
        C [i] ← ∞
        for j ← 1 to i-1 do:
            C [i] ← min{C [i], C [l] + p_ik}

return C [n]
```

Common mistakes:

1.Not initializing the array, or not initializing properly. 1 point deduction in most of the cases.

2.For people who used the Bellman-Ford solution: No extra deduction was made, however many people didn't implement their algorithms correctly. Namely, they gave 3 nested for-loops, traversing on i (1 <= i <= n-1), j (2 <= j <= n) and an intermediate index: k, where: i < k < j, and evaluated the expression C[i,k]+C[k,j]. However, this expression will not work since it uses values which haven't been set yet. 3 points deduction.

c) The reduction is correct. Most people got this part right.

d) Using the reduction from part (c): $O(n^2 \log n)$. Using the reduction from part (a): $O(n^2)$. The algorithm from part (a) is faster. People who used the Bellman-Ford solution for part (a) and analyzed their algorithms correctly got full marks for this part (they got an $O(n^3)$ running-time).

Question 2

Part (a)

We can simply replace every edge e of the edge-set with two opposite edges of the same capacity and reduce the problem to finding a minimum s-t cut in a directed graph. We know from the lectures that this can be done by a maximum flow computation using the Ford-Fulkerson algorithm. To find the min-cut, run an exploration algorithm on the (final) residual graph of the FF algorithm (BFS or DFS) starting from the vertex s. Set A the set of the vertices that you visited on this exploration, and B = V \A . This will corresponds to the undirected min-cut of G.

Common mistakes:

- Unclear construction of the directed graph., i.e., not specifying capacities, not replaced all undirected edges (2 marks off)
- Wrong Reduction to directed graph. (3-5 marks off).
- Don't specify how to find the min-cut from the residual graph, i.e., run BFS or argue how to find it: (1-2 marks off).

Part (b)

A naive solution is to run algorithm of part (a) for every distinct pair of vertices s,t and keep the minimum cut over all distinct pairs. This takes O(n^2) calls to the algorithm of part (a).

It can be done using O(n) calls to the algorithm of part (a). First we can arbitrary choose a vertex s. Let S^* be a subset of vertices of V, that gives an optimal cut ((S, V\S) is an optimal cut), then s will either be in S or in V \ S.. Thus, for every t in V\S, we solve two maximum flow problems, one giving us the minimum s->t cut, the other giving us the minimum t-s cut. Taking the minimum over all such cuts, we get the global min-cut in a undirected graph.

Common mistakes:

- Design an algorithm that gets as input vertices s, t. (2-4 marks off)
- Wrong algorithm, don't run algorithm of part (a) over all pairs or run it only on two arbitrary pairs. (2-4 marks off)

Question 3

Most students got this question right. There are two possible solutions**:**

- Using the max-flow min-cut theorem immediately implies the claim.
- Argue that the Ford-Fulkerson algorithm holds the invariant that at every iteration the current flow is multiple of 3; therefore at the last iteration will have a flow which is multiple of 3 and by correctness of FF is the max-flow.