

COT 5405 - Summer 2008

Homework 4(solution)

July 28, 2008

Grading Policy:

- Please contact TA Shahed Nejhum by email or in his office hours for any grading issues.
- Maximum score is 100 points.
- Each completed question worth 10 points.
- Partial credit is given if you dont answer a question completely.
- Problem 1 is graded for 50 points.
- Please notify the TA if you find anything wrong in this solution.

Problem 1 (Problem 24-2)

We consider d-dimensional boxes.

Part a:

The nesting relation is transitive.

Part b:

We can determine if a box nests within another by sorting the dimensions of each box and comparing them sequentially.

Part c:

To find the longest nest sequence we determine the nesting relations on all boxes by sorting each in $O(d \lg d)$ time and comparing then pairwise in $O(d)$ time for each of the $O(n^2)$ pairs. This produces a partial relation (by denition of nesting) and thus a directed acyclic graph. In this DAG we can find the longest length chain in the following way. From each node we determine the number and sequence of nesting boxes and remember the maximum length. This can be done in $O(n^2)$ time. The total running time is thus $O(d \lg d + dn^2 + n^2) = O(d(n^2 + \lg d))$.

Problem 2 (Exercise 25.1-10)

We can determine the presence of a negative-weight cycle by looking for a negative number in the diagonal. If $L^{(m)}$ is the first time for which this occurs then clearly the negative-weight cycle has length m . We can either use SLOW-ALL-PAIRS-SHORTEST-PATH in the straightforward manner or perform a binary search for m using FASTER-ALL-PAIRS-SHORTESTPATH.

Problem 3 (Exercise 25.2-4)

We observe that in Floyd-Warshall, we use the formula-

$$(d_{ij})^k = \min[(d_{ij})^{(k-1)}, (d_{ik})^{(k-1)} + (d_{kj})^{(k-1)}]$$

Now, if we drop the superscripts, that will imply we are using the same $n \times n$ table to compute and keep all the distances, and in each iteration we are updating the whole table.

Notice that to generate the next-iteration term for the LHS, we use 3 terms on the RHS, all belonging to the current iteration. So, if any of these terms on RHS change before calculating the term on LHS, we could potentially calculate the LHS term for the next generation incorrectly. By "changing" we mean by getting updated for the next generation.

Now, out of the three, the first term d_{ij} poses no problem as it is not possible for this term to get corrupted BEFORE the term on the LHS gets calculated (because they are the same term). However, out of the rest of the 2 terms (d_{ik} and d_{kj}), one or both of them may get calculated for the next generation already. The point is, can they have any different value after being calculated for the next iteration?

If you observe carefully, in the k -th iteration of the outermost loop, values in the k -th row and in the k -th column do not change. Why? Because, the d -values in these cells (d_{k*} and d_{*k}) represent the shortest paths between vertices where one of the two end vertices is k . Thus, if k is an end point, there is no way k is also an intermediate (non-end) vertex. Therefore, for these cells, the shortest paths between the end vertices do not change if the maximum index value of an intermediate vertex is k , because that is as good as saying that the maximum index value of an intermediate vertex is $k - 1$. Therefore, if k features in either as source or as destination, the d_{k*} and d_{*k} values do not change in the k -th iteration. Therefore, it does not matter whether d_{ik} and d_{kj} values have been tampered before calculating d_{ij} - even if they have been tampered, they have NOT changed. Thus, it does not affect the calculation of the d_{ij} of the k -th iteration in any way.

Problem 4 (Exercise 25.3-4)

It changes shortest paths. Consider the following graph. $V = \{s, x, y, z\}$, and there are 4 edges: $w(s, x) = 2, w(x, y) = 2, w(s, y) = 5,$ and $w(s, z) = -10$. now, we add 10 to every weight to make w^* . With w , the shortest path from s to y is $s \rightarrow x \rightarrow y$, with weight 4. With w^* , the shortest path from s to y is $s \rightarrow y$, with weight 15. (The path $s \rightarrow x \rightarrow y$ has weight 24.) The problem is that by just adding the same amount to every edge, you penalize paths with more edges, even if their weights are low.

Problem 5 (Problem 26-2)

Part a:

We construct a graph G' as given in hint of the problem. Then we apply a maximum flow algorithm on the graph G' . Since the graph is a bipartite graph, we will get a matching M in G' . Then a path cover P can be constructed as follows: moving from some vertex x_i to the matching y_j , we add edge from i to j . Then for x_j to its matching y_k , we augment path $i \rightarrow j$ by adding edge $j \rightarrow k$ and we continue like this.

Part b:

The algorithm does not work if the directed graphs contain cycles. For example: we consider directed graph with circle: $E = \{(1, 2), (2, 3), (3, 1), (4, 1)\}$ The maximum-flow found by max-flow algorithms could find the flow corresponding to either of the matchings described above, since both are maximal. But one of the matchings doesn't contain an edge incident to vertex 4, so given that matching, the path construction algorithm is forced to produce 2 paths, one of which contains just the vertex 4.