Purdue University
CS 381

S. Hambrusch-C. Guerra
Fall 1999

Solution Sketches to Assignment 8

1) (Graded by Cenyu Zhang) Prove or disprove each of the following claims.
(i) Let G be a weighted, directed graph in which edge weights can be negative. Dijkstra's algorithm for solving the Single-Source-Shortest-Paths problem works correctly on G, provided that G contains no negative-weight cycles.

The claim is FALSE. Using Dijkstra's algorithm on the graph shown below, the shortest distance from Node 1 to Node 2 will be 2, while it should be 1.

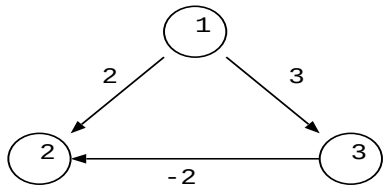


Figure 1: *Example*

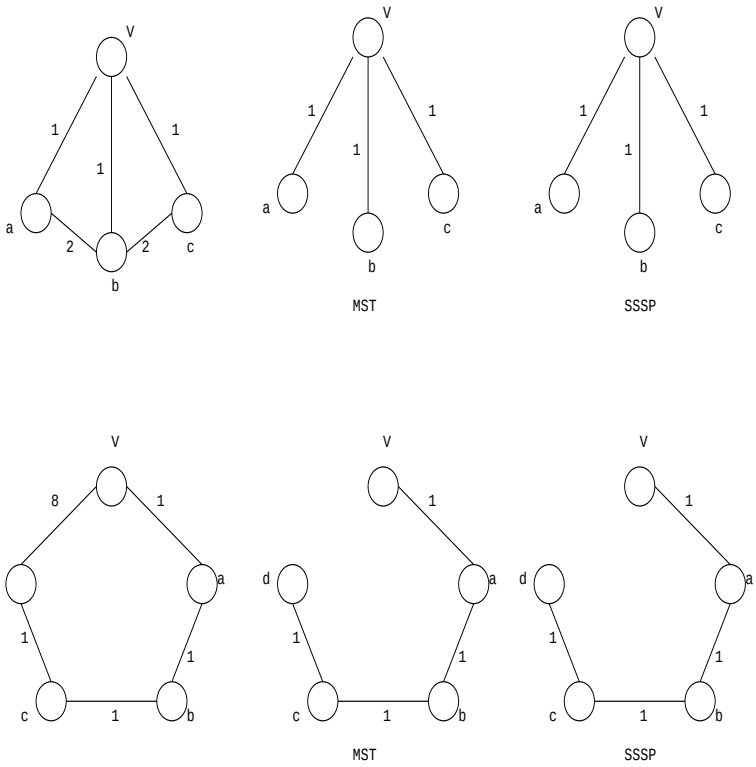
(ii) Let G be a weighted, directed graph in which some edge weights are negative. Let G' be the graph obtained from G by adding to every edge weight the absolute value of the minimum weight among all edges in G. From the Single-Source-Shortest-Paths tree for G' the Single-Source-Shortest-Paths tree for G can easily be generated.

Use the above graph G starting from Node 1. The SSSP tree from node 1 contains edges (1,3) and (3,2). After we transform G to G', then the SSSP tree contains (1,2) and (1,3). So, we cannot generate the SSSP tree for G from only the SSSP tree for G'.

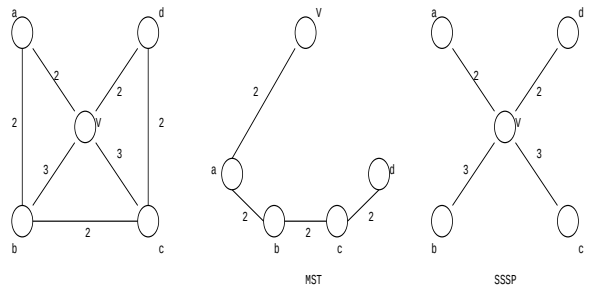
2)(Graded by Tian Luan) Let G be a weighted, undirected graph with positive edge weights. A spanning tree of G is a subgragh that is an undirected tree containing all the vertices. A Minimum Spanning Tree (MST) of a weighted graph is a spanning tree with mininum weight.
A Single Source Shortest Path tree (SSSP) is a tree with some root v such that the

path from v to every vertex u in the tree is the shortest path from v to u. Note that an MST and a SSSP tree contain all the vertices of the graph.

(i) Here are some graphs whose MST and SSSP rooted at v are the same.



(ii) Here are some graphs that generate very different MST and SSSP trees.



The two trees generated can not have disjoint edge sets. At the beginning, either algorithms chooses the lowest cost edge out of the root. Even if there are several edges with identical minimum weight out of the root, like in the above example, no matter which of them is chosen by MST, SSSP will include it eventually since no other path can have smaller weight than the lowest cost edge out of the root. So, the two trees will have at least one common edge.

3) (Graded by Cenyu Zhang) The Tippe Outpost Company maintains n posts along the Wabash River. At any of these posts you can rent a canoe which can be returned at any other post downstream. (It is next to impossible to paddle against the current.) For each possible departure point i and each possible arrival point j, $c(i, j)$ represents the cost of a rental between i and j. It is possible that the cost of renting from i to j is higher than the total cost of a series of shorter rentals. In this case you can return the canoe at some intermediate post k between i and j and continue the trip in another canoe. There is no extra charge for changing canoes.

(i) Describe an dynamic programming algorithm to determine the minimum cost of a trip by canoe from each possible departure point i to each possible arrival point j. Analyze the running time of your algorithm in terms of n.

We can use the all-pairs-shortest-distance algorithm to solve this problem. Define $D[i, j]$ as the distance of shortest path from post i to post j. If $i = j$ then $C[i, j] = 0$; If $i > j$ then $C[i, j] = \infty$; If $i < j$, then if i and j are adjacent then $C[i, j] = cost[i, j]$ else if they are not adjacent then $C[i, j] = min\ \{\ C[i, j], C[i, k] + C[k, j]\ \}$. Following is the $O(n^3)$ algorithm:

```
for k = 1 to n do
  for i = 1 to n do
    for j = 1 to n do
     if (k > i) and (k < j) then
      value = C[i,k] + C[k,j]
      if (value < C[i,j]) then
        C[i,j] = value
        B[i,j] = k
```

(ii) Describe the changes to be made to your algorithm so that not only the minimum costs are generated, but the canoe changing locations can be determined, if needed. Then, describe an $O(n)$ time algorithm to determine, for a given pair i and j, the canoe changing locations of a minimum cost trip.

In the above algorithm, a back pointer $B[i, j]$ is already added to record the changing locations. Initially all the entries in $B[i, j]$ are set to 0. In order to determine the changing locations of a given pair i and j, we only need to recursively trace the back pointer and print out the path. Following is the $O(n)$ time algorithm:

```
FindPath(i, j)

    k = B[i,j];
    if (k = 0) then if (i = j) then return i;
                    else return (i, j);
    else return (FindPath(i,k), k, FindPath(k,j));
```

4) (Graded by Tian Luan) (i)What is the order of the running time of this algorithm? As done in class, we consider the running time of determining which two matrices to multiply at which step. (The cost of performing the actual matrix multiplications is used to determine this order, but it does not enter the running time of the algorithm.) Without the support of an additional data structure, it takes $O(n)$ time to find the largest dimension in the array, and altogether there are n steps, so it is $O(n^2)$ time. Consider now setting up a max-heap using the entries of the dimension array. At each step we get the max value and reheapify the heap, which will take $O(\log n)$ time. The running time is now $O(n \log n)$.

(ii) This strategy does not always generate an ordering which minimizes the number of multiplication. The major difference between the dynamic programming and greedy algorithms is that dynamic programming considers all possible subproblems and saves the result of subproblems in a table (an optimal solution is then computed as a post-precessing step). The greedy algorithm makes whatever choice seems best at the moment and solves the subprograms arising after the choice is made. So the decision is the best for this moment, but it may not be best for the future.

In the following matrix multiplication problem, this greedy algorithm fails to generate an optimal solution.

$A1 : 1 * 2$

$A2 : 2 * 4$

$A3 : 4 * 3$

The greedy algorithm generates $A1(A2A3)$ which gives a cost of $2*4*3+1*2*3 = 30$

A better ordering is $(A1A2)A3$ and it gives a cost of $1 * 2 * 4 + 1 * 4 * 3 = 20$.