Data Structure and Algorithm
Homework #4

Due: 2:00pm, Thursday, May 17, 2012

TA email: dsa1@csie.ntu.edu.tw

=== Homework submission instructions ===

- For Problem 1, submit your source code, a shell script to compile the source, and a brief documentation to the SVN server (katrina.csie.ntu.edu.tw). You should create a new folder "hw4" and put these three files in it.
- The filenames of the source code, the shell script, and the documentation file should be "sollin.c", "compile.sh", and "report.txt", respectively. **The shell script should compile your source codes to generate an executable binary with the filename 'sollin'**. You will get some penalties in your grade if your submission do not follow the naming rule.
- The documentation file should be in plain text format (.txt file). In the documentation file you should explain how your code works, and anything you would like to convey to the TAs.
- For Problem 2 to Problem 5, submit the answers through the CEIBA system (electronic copy) or to the TA in R508 (hard copy).
- Each student may only choose to submit the homework in one way; either all as hard copies or all through CEIBA except the programming assignment. If you submit your homework partially in one way and partially in the other way, you might only get the score of the part submitted as hard copies or the part submitted through CEIBA (the part that the TA chooses).
- If you choose to submit the answers of the writing problems through CEIBA, please combine the answers of all writing problems into only ONE file in the doc/docx or pdf format; otherwise, you might only get the score of one of the files (the one that the TA chooses).
- Discussions with others are encouraged. However, you should write down your solutions by your own words. In addition, you have to cite the sources you consulted from or people you discussed with on the first page of your solution to that problem. The TA can deduct up to 100% of the score for any problem without the citation.
- No late submission of the homework will be given any score (for that portion).

Problem 1. Sollin's Algorithm (30%)

Sollin's algorithm could be seen as a hybrid version of Kruskal's and Prim's algorithms for the minimum spanning tree problem. In problem 1, you are asked to implement Sollin's algorithm to obtain the MST of an N-vertex undirected graph.

- Input Format

  N ← The number of the vertices of the undirected graph. That means we have vertex 0 - vertex N-1, where N $\leq$ 10.

  i j cost(i, j) ← The edge is between vertex i and vertex j and the cost of this edge is cost(i, j). The numbers are separated by space characters. The index of the vertices is from 0 to N-1 and cost(i, j) is a distinct positive integer. These 3 numbers can be stored into 32-bit integers.

  ⋮

  (You have to read lots of edges until the EOF.)

- Output Format

  You are required to output two parts, as shown in the following description.

  – List all the added edges and its cost, (i,j,cost(i,j)), produced by each iteration in an ascending order according to the cost of each added edge. Note that the order of two vertices for representing each edge must be identical with the input. For example, if one edge in input is represented as (i,j), output for this edge must be (i,j), not (j,i).

  – Print the sequence of the Depth-first search of the final MST starting from vertex 0.

- Sample Input

```
5
0 2 50
1 2 25
0 1 40
3 4 30
1 3 5
2 3 15
4 2 10
```

- Sample Output

```
iteration 1: (1,3,5)(4,2,10)(0,1,40)
iteration 2: (2,3,15)
0 1 3 2 4
```

Figure 1 shows the graph for sample input. Figure 2 and Figure 3 show the progress after iteration 1 and iteration 2.

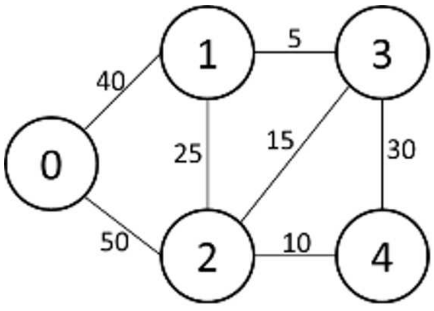

Figure 1: The graph of the sample input for problem 1

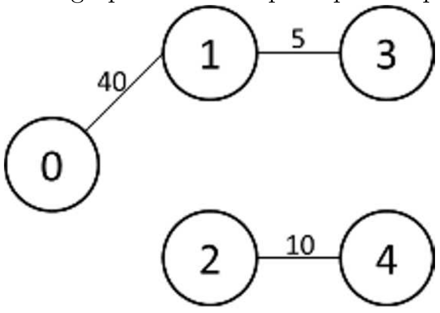

Figure 2: The progress of Sollin's algorithm after iteration 1

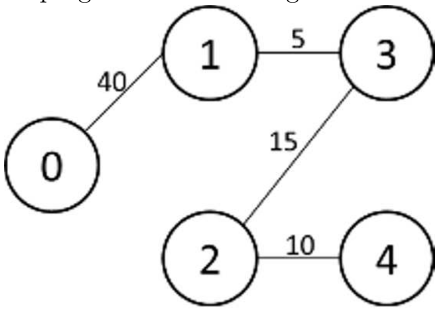

Figure 3: The progress of Sollin's algorithm after iteration 2

Problem 2. Off-line minimum problem (17%)

The off-line minimum problem asks us to maintain a dynamic set T of elements from the domain $\{1, 2, \dots, n\}$ under the operations INSERT and EXTRACT-MIN. INSERT is an operation to insert an element into a list, and EXTRACT-MIN is an operation to find the minimum value of the list. We are given a sequence S of n INSERT and m EXTRACT-MIN calls, and each key in $\{1, 2, \dots n\}$ is inserted exactly once. We wish to determine which key is returned by each EXTRACT-MIN call. Specifically, we wish to fill in an array $extracted[1..m]$, where for $i = 1, 2, \dots, m$, $extracted[i]$ is the key returned by the i-th EXTRACT-MIN call. The problem is "offline" in the sense that we are allowed to process the entire sequence S before determining any of the returned keys.

1. (2%) In the following instance of the off-line minimum problem, each INSERT is represented by a number and each EXTRACT-MIN is represented by the letter E, please fill in the values in the *extracted* array by the following input sequence S_1.
   $S_1 : 7, 4, E, 1, E, 9, 3, E, E, 8, E, 6, 2, E, 5.$

2. (5%) Design an algorithm to find the extracted array T. The input of the algorithm is a sequence S. In addition, what is the time and space complexity of your algorithm? Please explain the reason. Note that if your algorithm is written by C code or pseudo code, you should explain it in detail.

3. (5%) Here is an algorithm to solve off-line minimum problem. The input of the algorithm is a sequence of sets $K = \{K_1, K_2, \ldots, K_{m+1}\}$, where K_j represents a sequence of INSERT calls and m is the number of EXTRACT-MIN calls. Besides, K_j and K_{j+1} are separated by an EXTRACT-MIN call. Note that if there are two consecutive EXTRACT-MIN calls, the set between these calls is an empty set.
   Here is an example of K, which is generated by a sequence S_2:
   $S_2 : 2, 3, E, 4, E, E, 1.$
   $K_1 = \{2, 3\}, K_2 = \{4\}, K_3 = \{\phi\}, K_4 = \{1\}.$
   The pseudo code of the algorithm is listed as follows: (K_j is denoted as K[j] in the algorithm)

```
OFF-LINE-MINIMUM(m,n,K)
1 for i = 1 to n
2   determine j such that i belongs to K[j]
3   if j is not equal to m+1
4     extracted[j]=i
5     let l be the smallest value greater than j for which set K[l] exists
6     K[l] = K[j] union K[l], destroying K[j]
7 return extracted
```

   Please explain that why the array *extracted* returned by `OFF-LINE-MINIMUM` is correct, as well as the time complexity of this algorithm. You can assume that there are n INSERT and m EXTRACT-MIN calls.

4. (5%) How to modify `OFF-LINE-MINIMUM` with a disjoint-set data structure to make the algorithm more efficient? What is the time complexity of the modified version?

Problem 3. Activities On Edge (18%)

Please read about Activitie-On-Edge (AOE) (from AOE.pdf, also posted on the course website) and answer the following questions:

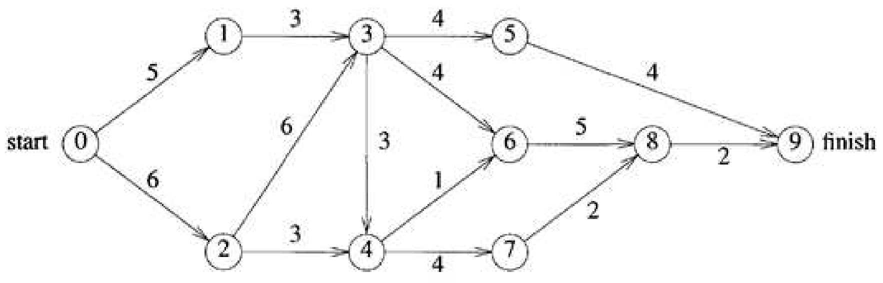

Figure 4: AOE network

1. (3%) Please explain what is a "critical" activity?
2. (3%) If we shorten the time needed to finish a non-critical activity, does it also help shorten the total time needed to finish the whole project? Please briefly explain your reason.
3. (3%) For the AOE network of Figure 4, obtain the early and late starting times for each activity. Use the forward-backward approach.
4. (3%) What is the earliest time the project can finish?
5. (3%) Which activities are critical?
6. (3%) Is there a single activity whose speed up would result in a reduction of the project length?

Problem 4. Graph algorithms: Breadth-first search & Depth-first search (25%)

1. (5%) In the lecture, we have learned that BFS runs in $O(V+E)$ time on a graph $G=(V,E)$ represented by adjacency lists. If we utilize an adjacency matrix to represent the input graph, what is the running time of BFS? Please explain.
2. (5%) Design an EFFICIENT algorithm to compute the diameter of an undirected tree. The **diameter** of a tree $T=(V,E)$ is defined as the largest of all shortest-path distances in the tree. Please also analyze the running time of the algorithm. Note that if your algorithm is written by C code or pseudo code, you should explain it in detail.
3. (5%) Design an iterative version of DFS by using a stack, instead of using recursive function calls. If your algorithm is written by C code or pseudo code, you should explain it in detail.
4. (10%) Prove that a strongly connected, directed graph $G=(V,E)$ has an Euler tour if and only if in-degree(v) =out-degree(v) for each vertex $v \in V$. The definition of an Euler tour is a cycle that traverses each edge of G exactly once, although it may visit a vertex more than once.

Problem 5. (10%) Most graph algorithms that take an adjacency-matrix representation as input require time $\Omega(V^2)$, but there are some exceptions. Derive an algorithm to determine whether a directed graph G contains a universal sink - a vertex with in-degree $|V| - 1$ and out-degree 0 in time $O(V)$, given an adjacency matrix for G. Note that you have to explain how the algorithm works in detail.