THE UNIVERSITY OF WESTERN ONTARIO

DEPARTMENT OF COMPUTER SCIENCE
LONDON CANADA

Analysis of Algorithms
(Computer Science 340b)

ASSIGNMENT 3
Due date: Thursday, April 2, 2015, 11:55 PM

1. In the textbook, 15.4-2 (pp. 396).

2. In the textbook, 16.2-3 (pp. 427).

3. In the textbook, 16.2-7 (pp. 428).

4. (optional for bonus credit) In the textbook, 16.3-9 (pp. 437).

5. Compute the $next[\ ]$ function (the prefix function π in the text book) for the pattern $P = babbabbabbababbabb$.

6. Modify the KMP string matching algorithm to find the largest prefix of P that matches a substring of T. In other words, you do not need to match all of P inside T; instead, you want to find the largest match (but it has to start with p_1).

7. Modify minimum spanning tree algorithm to find the maximum spanning tree.

8. Find a counter example that shows Dijkstra's algorithm does not work when there is negative weight edge.

9. Given a weighted directed graph $G = (V, E)$, suppose that there are negative weights in G, but there is no negative cycle in G. Is all-pair-shortest-path algorithm still correct? Prove your answer.

10. Let $G = (V, E)$ be a weighted directed graph with no negative cycle. Design an algorithm to find a cycle in G with minimum weight. The algorithm should run in time $O(|V|^3)$.

11. Next page.

11. Implement the Prim's minimum spanning tree algorithm for a weighted undirected graph using a heap data structure. The time complexity of the algorithm should be $O((|V| + |E|) \log |V|)$.

The heap data structure should be implemented as an abstract data type (a class in C++) on a set of elements, where each element has an id and a key, with the following operations.

- $heap\_ini(keys, n)$: initializes a heap with the array keys of n elements.
- $in\_heap(hp, id)$: returns true if the element whose id is id is in heap hp;
- $min\_key(hp)$: returns the minimum key of hp;
- $min\_id(hp)$: returns the id of the element with minimum key in hp;
- $key(hp, id)$: returns the key of the element whose id is id in hp;
- $delete\_min(hp)$: deletes the element with minimum key from heap hp;
- $decrease\_key(hp, id, new\_key)$: sets the key of the element whose id is id to new_key if its current key is greater than new_key.

An input graph file will be available. The format of the input file is the following: the first line of the input file contains an integer indicating the number of vertices of the input graph, each of the remaining lines contains a triple $''i\ j\ w''$ indicating an edge between vertex i and vertex j with cost w.

The output of your program should be the input graph in list representation format as well as the edges (with their weights) of the minimum spanning tree in the order in which they are produced by the Prim's algorithm.

In your gaul account, you should have a directory called "asn3" which contains your program, the input file, and a makefile. The makefile should be written such that the command "make clean" will remove all the "*.o" files and the command "make" will generate an executable file "asn3" that can be run by typing "asn3 < infile". If you are using Java, you may not need the makefile. In that case, you should have script file, asn3, so that by typing "asn3 < infile" your java programs will run.

You should use script command to capture the screen of the execution of your program. The resulting file should also be in directory "asn3".