CS180 Homework 4

Due: 4:00pm, 04/27/2016

1. Consider an undirected graph that weights of edges are *distinct*. We can define an order on the set of spanning trees of the graph as follows: each spanning tree has $n-1$ edges, by sorting the weights of all $n-1$ tree edges in the *increasing* order, we obtain a tuple $(w_1, \ldots, w_{n-1})$ where $w_1 < w_2 < \ldots < w_{n-1}$. We can easily define a lexicographic order on tuples by imagining that each tuple is a word with $n-1$ letter. More precisely, we say tuple $(w'_1, \ldots, w'_{n-1})$ is bigger than tuple $(w_1, \ldots, w_{n-1})$ if for the smallest index i at which $w'_i \neq w_i$, we have $w'_i > w_i$. We say tree $T_1 > T_2$ if T_1's tuple is lexicographically bigger than T_2's tuple. We define the *lexicographically minimum tree* to be the smallest tree in this ordering of trees. Prove that the lexicographically minimum tree is a minimum spanning tree.

2. Given an undirected graph with *distinct* weights. Consider a spanning tree for the graph: given any pair of vertices u and v, the tree can be used to return a path between u and v such that maximum-weight edge on the path has a smaller weight than the maximum-weight edge of any other path between u and v.

   (a) Prove that the spanning tree with the above property is the minimum spanning tree (mst).

   (b) We define a order between two paths A and B. We sort the weights of edges in the *decreasing* order and $A = \{w_1, w_2, \ldots, w_i\}, B = \{w'_1, w'_2, \ldots, w'_j\}$. We say A is bottleneck-smaller than B, if the first different weight edge in A has smaller weight than the edge in B, more precisely if for the smallest index k in A at which $w'_k \neq w_k$, we have $w_k < w'_k$. If there is not such k that $w'_k \neq w_k$, the path with smaller number of edges is bottleneck-smaller. We define the minimum bottleneck path as the path that is bottleneck-smaller than any other path.

   Consider the following variation of Dijkstra algorithm. Instead of computing the distance, we calculate the minimum bottleneck path from the source s to any other node. Each node v maintain a variable $P(v)$ that stores the minimum bottleneck path from the s and ends at v. Moreover, the operation on the priority queue is base on the order of $P(v)$.

```
DIJKSTRAMST($s$)
  put $s$ in the priority queue
  $T \leftarrow \{\}$
  while the priority queue is not empty
      extract node $u$ from the priority queue with the minimum bottleneck path
      remove $u$ from priority queue
      $T \leftarrow T \cup \{u\}$
      for all edge $u - v$ that $v$ is not in $T$
          if $v$ is not in priority queue
              $P(v) \leftarrow \{P(v), u - v\}$
              put $v$ in the priority queue
          else if $P' = \{P(u), u - v\}$ is bottleneck-smaller than $P(v)$
              $P(v) \leftarrow \{P(v), u - v\}$
```

   i. Assume that in the execution of the DIJKSTRAMST you maintain a set contains all edges in all $P(v)$. Prove that at the end of the execution, such set defines a spanning tree of the graph.

   ii. Prove that the spanning tree defined by DIJKSTRAMST is the mst.

   iii. Assume that you execute DIJKSTRAMST and the Prim's algorithm separately on node s. Prove or disprove that the two algorithms construct the mst in the same order.

3. There is a multi-day tennis tournament with $n = 2^k$ players. On a given day, each player plays one match, so $n/2$ matches are played per day. Over the course of the tournament, each player plays against every other player exactly once, so a total of $n(n-1)/2$ matches will be played.

Write a recursive algorithm to schedule these matches in the minimum number of days. It should take as input a number $n = 2^k$ and return as output a table with $n - 1$ rows, in which row i is a list of the $n/2$ matches to be played on day i.

- ★ Homework assignments are **STRICTLY** due on the exact time indicated. Please submit a hard copy of your homework solution with your **Name**, **Bruin ID**, **Discussion Number**, clearly indicated on the first page. If your homework consists of multiple pages, please **staple** them together. Email attachments or other electronic delivery methods are not acceptable.
- ★ We recommend to use LaTeX, LyX or other word processing software for submitting the homework. This is not a requirement but it helps us to grade the homework and give feedback. For grading, we will take into account both the correctness and the clarity. Your answer are supposed to be in a simple and understandable manner. Sloppy answers are expected to receiver fewer points.
- ★ Unless specified, you should justify your algorithm with proof of correctness and time complexity.