COS 226	Algorithms and Data Structures	Fall 2005

Final Solutions

1. **Analysis of algorithms.**

   (a) It provides a worst-case running time of a sequence of operations, starting from an initially empty data structure. For example, starting from an initially empty Fibonacci heap, any sequence of x INSERT, y DECREASEKEY and z DELETEMIN operations takes at most $k(N + x + y + z \log N)$ steps for some constant $k > 0$.

   (b) Dijkstra's algorithm performs at most V INSERT, E DECREASEKEY and V DELETEMIN operations. Thus, the overall worst-case running time is $O(E + V \log V)$.

   (c) If we could implement INSERT and DELETEMIN in $O(1)$ time, then we could sort N elements in linear time (insert the N elements, then repeatedly delete the minimum). This would violate the $\Omega(N \log N)$ lower bound we have for sorting algorithm that access the data only through pairwise comparisons.

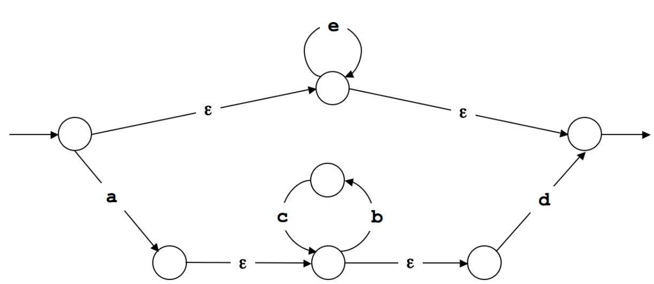
2. **String searching.**

   (a) Yes. `bbbabbb`

   (b) No.

   (c) Replace the edge labeled `b` from 2 to 1, and make it go from 2 to 2.

3. **Pattern matching.**



4. **Convex hull.**

   (a) `H G E F C D B A`

   (b) `1. I H`
   `2. I H G`
   `3. I H G E`
   `4. I H G E F`
   `5. I H C`
   `6. I H C D`
   `7. I H C B`
   `8. I H C A`

5. **Geometry.**

   For simplicity, we assume no two endpoints have the same value.

   (a) The $2N$ events are the left and right endpoints of each interval.

   (b) To implement the sweep line, sort the endpoints and process in ascending order, say using mergesort.

   (c) Store the set of intervals intersecting the sweep line in a priority queue (say, a binary heap), using the right endpoint as the key.

   (d)
   - Left endpoint: insert the interval onto the PQ. Check the number of elements on the PQ, if it is the most so far, record the x value of the current left endpoint.
   - Right endpoint: perform a delete the min on the PQ. This removes the corresponding interval from the PQ.

   Note that the PQ isn't strictly needed, since we could just increment a counter when processing a left endpoint, and decrement it when processing a right endpoint.

6. **Digraphs and DFS.**

   (a) Preorder: `A B C F D E G H I`.

   (b) Postorder: `C F B E I H G D A`.

   (c) Topological: `A D G H I E B F C`.

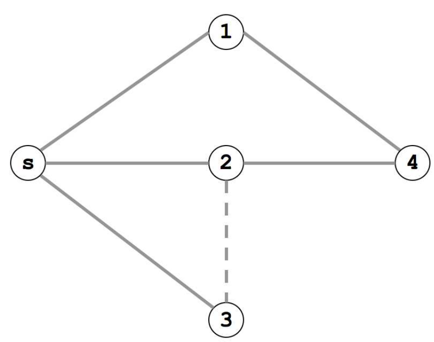
7. **Undirected graphs and BFS.**

The key idea is that a shortest cycle is comprised of a shortest path between two vertices, say v and w, that does not include edge v-w, plus the edge v-w. We can find the shortest such path by deleting v-w from the graph and running breadth-first search from v (or w).

```
For each edge v-w
   - Form a graph that is the same as G, except that edge v-w is removed.
   - Find the shortest path dist(v, w) from v to w using BFS.
   - Compute dist(v, w) + 1, which corresponds to the cycle consisting
     of the path from v to w, plus the edge v-w.
   - If this is shorter than the best cycle found so far, save it.
```

We run BFS E times and each run takes $O(E + V)$ time. The overall algorithm takes $O(E(E + V))$ time.

Note that if you run BFS from s and stop as soon as you revisit a vertex (using a previously ununused edge), you may not get the shortest path containing s. For example, in the following graph, BFS might consider the edges s-1, s-2, s-3, 1-4, 2-4, thereby finding the cycle s-1-4-2-s (instead of the shorter cycle s-2-3).



8. **Minimum spanning tree.**

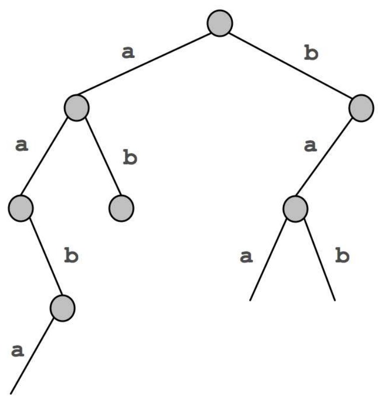
   (a) `C-D  A-C  E-F  H-I  G-I  A-B  E-G  D-I`

   (b) `A-C  C-D  A-B  D-I  H-I  G-I  E-G  E-F`

9. **Data compression.**

   (a) `a b aa ab ba aab bab baa aaba`

   (b)



10. **Linear programming.**

$$
\begin{array}{lrcrcrcrcrcrcrcl}
\text{maximize} & -26A & - & 30B & - & 20C & & & & & & & & \\
\text{subject to:} & A & + & B & + & C & & & & & & & = & 100 \\
 & 2A & + & 6B & + & 3C & - & S_1 & & & & & = & 145 \\
 & 7A & + & 1B & + & C & & & - & S_2 & & & = & 85 \\
 & 5A & + & 1B & + & 6C & & & & & + & S_3 & = & 95 \\
 & A & , & B & , & C & , & S_1 & , & S_2 & , & S_3 & \geq & 0
\end{array}
$$

11. **Reductions.**

Create a new weighted digraph G' as follows:

- G' has the same vertices as G plus two new vertices s and t.
- G' has the same edges as G plus a new edge from s to every vertex in G and an edge from every vertex in G to t.
- The weight of every edge is -1.

Observe that G has a Hamiltonian path if and only if G' has a shortest simple path from s to t of length exactly $-(V+1)$.