Algorithms CSs545 — Homework #5
Due: 11/27/02.

December 16, 2002

1. Assume that we need to find the longest common subsequence LCS(X, Y) between two sequences X and Y. Assume that the length of X and the length of Y are both n, and n is very large compares to the memory size of our computer. Thus we have to execute the algorithm while moving memory back and forth from the disk. Assume that the memory size is m bytes, and that at each I/O operations we transfer b bytes, and that $m \leq 10b$. How would you execute that algorithm in order to minimize the number of I/O operations ?

2. CLRS 15-1 (=CLR 16-1).

3. CLRS 16-3.1 (=CLR17.3-1).

4. CLRS 16-3.2 (CLR 17.3-2)

5. CLRS 16-3.3 (CLR 17.3-3)

6. CLRS 16-3.4 Prove that if we order the characters in the alphabet so that their frequencies decreases monotonically, then there exist an optimal code where the length of the code words monotonically increases.

7. CLRS 16.3-6 (= CLR 17.3-6)

8. Let $G = (V, E)$ be an undirected graph, where each vertex $v_i \in V$ is associated with a positive weight w_i. The cost of we path in the graph is the sum of weights of vertices of the path. Give an algorithm for the following problem: Given vertices $s, t \in V$, find the path of minimal cost connecting s to t. Your algorithm should be as efficient as possible. What is the running time of you algorithm ?

   Answer: *It is easy to modify Dijkstra algorithm to handle this case. Instead, we would should different approach that would save the need to repeat the proof.*

We first create from G a directed *graph $G'(V, E')$ by replacing each (undirected) edge of $(u, v) \in E$ by two directed edges $(u, v), (v, u) \in E'$. Clearly, the path π from s to a vertex v exists.*

Next we replace every vertex $v_i \in V$ with a pair of vertices v'_i, v''_i, where all incoming edges to v points now to v'_i, all edges originally leaving v_i are now leaving v''_i, and we add to E' the edge (v'_i, v''_i) with weight w_i. We find shortest path from s to all vertices v''_i.

9. Let $G = (V, E)$ be a graph with positive weights associated with its edges, and let F be a set of vertices of V. The cost of a path connecting two vertices is the sum of weights of its edges. The distance of a vertex $v \in V$ from F is the the cost of the cheapest path connecting v and some vertex of F. Give an efficient algorithm to find the distance of every vertex $v \in V$ from F. What is the running time ?

   ans

   Solution 1) Use Dijkstra's algorithm, but initial the set S used by the algorithm to be F, where $d[v] = 0$ for every $v \in S$.

   Solution 2) create a new vertex $s \notin V$, and connect it to all vertices of F, by edges of weight 0.

10. Let $G = (V, E)$ be a graph with positive weights associated with its edges, where the weight of each edge is an integer number between 1 and 17. The cost of a path connecting two vertices is the sum of weights of its edges. Describe an algorithm with running time $O(|E| + |V|)$ for finding the length of the cheapest path connecting each vertex in V to a fixed vertex $s \in V$.

    Answer: *Solution (1) For those who know about BFS algorithm for finding path in a graph without weights: Replace each edge of weight w with a path of length w, and run BFS.*

    Solution (2) Run Dijkstra algorithm. The values of the variables $d[v]$ used by the algorithm are always in the range $1 \ldots n$, or ∞. Create an array of G of size n, where $G[i]$ contains a pointer to a liked list of all nodes v for which $v \notin S$ and $d[v] = i$. Is is not hard to show that only a constant number of non-empty cells exists in G. (prove).