HW#5 Solution, CSCI 323, Spring 2003
Department of Computer Science
Queens College
Dr. Song

Problem 1 (R-7.12) Solution

(omitted. If you have trouble figuring out this problem, please talk the the instructor.)

Problem 2 (C-7.7) Solution

We can model this problem using a graph. We associate a vertex of the graph with each switching center and an edge of the graph with each line between two switching centers. We assign the weight of each edge to be its bandwidth. Vertices that represent switching centers that are not connected by a line do not have an edge between them.

We use the same basic idea as in Dijkstras algorithm. We keep a variable $d[v]$ associated with each vertex v that is the bandwidth on any path from a to this vertex. We initialize the d values of all vertices to 0, except for the value of the source (the vertex corresponding to a) that is initialized to infinity. We also keep a π value associated with each vertex (that contains the predecessor vertex). The basic subroutine will be very similar to the subroutine Relax in Dijkstra. Assume that we have an edge (u,v). If $\min\{d[u], w(u,v)\} > d[v]$ then we should update $d[v]$ to $\min\{d[u], w(u,v)\}$ (because the path from a to u and then to v has bandwidth $\min\{d[u], w(u,v)\}$, which is more than the one we have currently).

Algorithm 1 Max-Bandwidth(G, a, b)

```
Input: a graph G, vertices a and b
Input: the maximum bandwidth between a and b
for each vertex v in V do
    d[v] ← 0
end for
d[a] ← ∞
Unknown ← V
Known ← φ
while Unknown is not empty do
    do u ← ExtractMax(Unknown)
    Known ← Known ∪{u}
    for each vertex v in Adj[u] do
        if min{d[u], w(u,v)} > d[v] then
            d[v] ← min{d[u], w(u,v)}
        end if
    end for
end while
return d[b]
```

Here, the function ExtractMax(Unknown) finds the node in Unknown with the maximum value of d.

Complexity (this has not been asked for): If we maintain Unknown as a heap, then there are $n = |V|$ ExtractMax operations on it and upto $m = |E|$ changes to the values of elements in the heap. We can implement a data structure which takes $O(\log n)$ time for each of these operations. Hence, the total running time is $O((n+m)\log n)$.

Problem 3 (C-7.8) Solution

Consider the weighted graph $G = (V,E)$, where V is the set of stations and E is the set of channels between the stations. Define the weight $w(e)$ of an edge $e \in E$ as the bandwidth of the corresponding channel.

Given below are two ways of solving the problem:

1. At every step of the greedy algorithm for constructing the minimum spanning tree, instead of picking the edge having the least weight, pick the edge having the greatest weight.

2. Negate all the edge-weights. Run the usual minimum spanning tree algorithm on this graph. The algorithm gives us the desired solution.

Problem 4 Solution

Table 1: Alignment value matrix V to align $S = ATGACTCATCTG$ and $T = ATCGAATAAG$.

			A	T	C	G	A	A	T	A	A	G
	V	0	1	2	3	4	5	6	7	8	9	10
	0	0	-3	-6	-9	-12	-15	-18	-21	-24	-27	-30
A	1	-3	1	-2	-5	-8	-11	-14	-17	-20	-23	-26
T	2	-6	-2	2	-1	-4	-7	-10	-13	-16	-19	-22
G	3	-9	-5	-1	-1	0	-3	-6	-9	-12	-15	-18
A	4	-12	-8	-4	-4	-3	1	-2	-5	-8	-11	-14
C	5	-15	-11	-7	-3	-6	-2	-2	-5	-8	-11	-14
T	6	-18	-14	-10	-6	-6	-5	-5	-1	-4	-7	-10
C	7	-21	-17	-13	-9	-9	-8	-8	-4	-4	-7	-10
A	8	-24	-20	-16	-12	-12	-8	-7	-7	-3	-3	-6
T	9	-27	-23	-19	-15	-15	-11	-10	-6	-6	-6	-6
C	10	-30	-26	-22	-18	-18	-14	-13	-9	-9	-9	-9
T	11	-33	-29	-25	-21	-21	-17	-16	-12	-12	-12	-12
G	12	-36	-32	-28	-24	-20	-20	-19	-15	-15	-15	-11

All optimal alignments:

1.
$$\begin{array}{lcccccccccccc} S' = & A & T & - & G & A & C & T & C & A & T & C & T & G \\ & | & | & & | & | & & | & & | & & & & | \\ T' = & A & T & C & G & A & A & T & A & A & - & - & - & G \end{array}$$

2.
$$\begin{array}{lcccccccccccc} S' = & A & T & - & G & A & C & T & C & A & T & C & T & G \\ & | & | & & | & | & & | & & | & & & & | \\ T' = & A & T & C & G & A & A & T & - & A & A & - & - & G \end{array}$$

3.
$$\begin{array}{lcccccccccccc} S' = & A & T & - & G & A & C & T & C & A & T & C & T & G \\ & | & | & & | & | & & | & & | & & & & | \\ T' = & A & T & C & G & A & A & T & - & A & - & A & - & G \end{array}$$

4.
$$\begin{array}{lcccccccccccc} S' = & A & T & - & G & A & C & T & C & A & T & C & T & G \\ & | & | & & | & | & & | & & | & & & & | \\ T' = & A & T & C & G & A & A & T & - & A & - & A & - & G \end{array}$$

5.
$$\begin{array}{lcccccccccccc} S' = & A & T & - & G & A & C & T & C & A & T & C & T & G \\ & | & | & & | & | & & & & | & | & & & | \\ T' = & A & T & C & G & A & - & - & - & A & T & A & A & G \end{array}$$