MCGILL UNIVERSITY – COMP 360

TEST 2 SOLUTION

Question 1 [10pt]

(a)[3] (i)[1]

$$B(1,V)=\begin{cases} w_1 & \text{if } V=v_1\\ \infty & \text{otherwise}\end{cases}$$

Comment: Most people put 0 instead of ∞ when $V\neq v_1$. This is wrong.

(ii)[2]

$$B(i,V)=\begin{cases} B(i-1,V) & \text{if } V<v_i\\ \min\{B(i-1,V), w_i+B(i-1,V-v_i)\} & \text{otherwise}\end{cases}$$

(b)[1] All-pairs shortest paths.

(c)[2] This is the same array as that of Bellman–Ford algorithm. So A is a two dimensional array of size $n\times n$, where $n=|V|$. $A[i,v]$ is the smallest cost of going from vertex v to t using at most i edges.

(d)[2] A directed graph with two distinguished vertices s and t, so that for every other vertex v there is a path from s to t via v. Also, there is no edge coming to s and no edge going out from t.

(e)[1] $\mathcal{O}(Cm)$ where m is the total number of edges of the network, and C is the total capacity of the edges leaving s.

Comment: Several people say $\mathcal{O}(mn)$, etc. which are wrong.

(f)[3]

1. $f\leftarrow$ 0-flow
2. $\Delta\leftarrow$ largest power of 2 not exceeding the maximum capacity leaving s
3. while $\Delta>0$ do
4.   while there are st-path in $G_f(\Delta)$ do
5.   let P be an st-path in $G_f(\Delta)$
6.   *augment*(f,P)
7.   end while
8. $\Delta\leftarrow\Delta/2$
9. end while

Comment: Many people left this blank. Many others gave only one while loop, which is not correct.

Question 2 [10pt]

(a)[2] A is a two dimensional array of size $n\times D$, where D is the maximum of all d_i.

Sort the jobs in increasing order of their deadlines. So

$$d_1\leq d_2\leq\ldots\leq d_n$$

For $1\leq i\leq n$ and $1\leq d\leq D$, $A[i,d]$ is the maximum number of jobs among jobs $\{1,2,\ldots,i\}$ that can be scheduled to be done at time d at the latest.

(b)[3] Assume without loss of generality that $d_i\geq t_i$ for all jobs i, because any jobs with $d_i<t_i$ cannot be scheduled anyway.

Initialization:

$$A[1, d] = \begin{cases} 1 & \text{if } t_1 \le d \le d_1 \\ 0 & \text{otherwise} \end{cases}$$

Recurrence:

$$A[i+1, d] = \begin{cases} A[i, d] & \text{if } d < t_{i+1} \text{ or } d < t_{i+1} \\ \max\{A[i, d], 1 + A[i, \min(d_{i+1}, d) - t_{i+1}]\} & \text{otherwise} \end{cases}$$

(c)[2]

1. sort the jobs so that $d_1 \le d_2 \le \ldots \le d_n$
2. $D \leftarrow \max\{d_1, d_2, \ldots, d_n\}$
3. for d from 1 to $t_1 - 1$ do $A[1, d] \leftarrow 0$ end for
4. for d from t_1 to d_1 do $A[1, d] \leftarrow 1$ end for
5. for d from $d_1 + 1$ to D do $A[1, d] \leftarrow 0$ end for
6. for i from 1 to $n - 1$ do
7.   for d from 1 to $t_{i+1} - 1$ do $A[i+1, d] \leftarrow A[i, d]$ end for
8.   for d from t_{i+1} to D do
9. $A[i+1, d] \leftarrow \max\{A[i, d], 1 + A[i, d_{i+1} - t_{i+1}]\}$
10.   end for
11. end for

(d)[2]

1. $S \leftarrow$ empty set (this is out solution set)
2. $i \leftarrow n - 1$, $d \leftarrow D$
3. while $A[i+1, d] > 0$ do
4.   if $A[i+1, d] = A[i, d]$ then $i \leftarrow i - 1$
5.   else
6.   add $(i+1, d_{i+1} - t_{i+1})$ to S
7. $i \leftarrow i - 1$, $d \leftarrow d_{i+1} - t_{i+1}$
8.   end if
9. end while
10. output S

(e)[1] Running time: $\mathcal{O}(nd)$. Space: $\mathcal{O}(nd)$.

Question 3 [8pt]

(a)[3] The flow network G consists of two vertices s, t, and the following vertices: there is a vertex v_i for each client i, and a vertex u_j for each base station j. The edges and their capacities are:

- (s, v_i) with capacity 1,
- (v_i, u_j) for all i, j such that the distance from client i to base j is at most r, i.e.,

$$\sqrt{(x_i - a_j)^2 + (y_i - b_j)^2} \leq r$$

- (u_j, t) with capacity L.

(b)[1] We need to compute the distance between every pair (client,base). There are nk such a pair. So this takes time $\mathcal{O}(nk)$. Other edges require no computation. So total time for constructing the network is $\mathcal{O}(nk)$.

(c)[1] Let f be a maximum flow in the network. We output YES if the value of f is equal to n, the number of clients.

(d)[3] Each way of connecting all n clients to the base station (satisfying the load and range conditions) gives a flow of value n, by letting the flow on edge (v_i, u_j) be 1 if and only if client i is connected to the base station j, and letting a flow of 1 on all edges (s, v_i), and letting a flow on (u_j, t) be the total number of clients that are connected to station j.

On the other hand, the maximum flow in the network has value at most n, because this is the sum of capacities on the edges leaving s. Also, a flow of value n (with integer flow on the edges) defines a way of assigning clients to station, by letting client i to be connected to station j iff the flow value on edge (v_i, u_j) is 1.