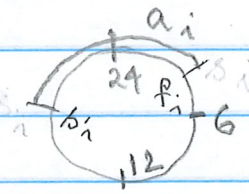
Homework 5 (Solution)

1. $S = \{a_1, a_2, \cdots, a_n\}$ set of activities in a 24 hr. day schedule.

Activity a_i has a starting time s_i + finish time f_i.
The duration of activity a_i is

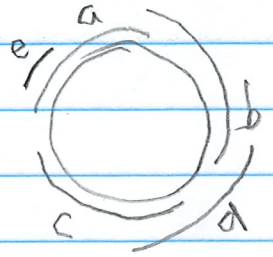
i) $f_i - s_i$ if $s_i < f_i$

ii) $24 + f_i - s_i$ if $s_i > f_i$



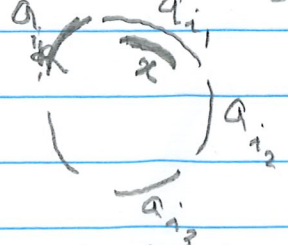
Two activities, $a_i \neq a_j$ are compatible if their aires don't overlap overlap.

Activity a_i is a proper activity if it does not contain another activity



e, b, d, c : proper
a : is not proper
b, d : not compatible
e, b, c : compatible

1(a) Suppose A^* contains an activity which is not proper. Let $\langle a_{i_1}, a_{i_2}, \cdots, a_{i_k} \rangle$ be the compatible activities with at least a_{i_1} activity being not proper.
Since a_{i_1} is not proper, there exists an activity x that is contained in a_{i_1}. Clearly, $\langle x, a_{i_2}, a_{i_3}, \cdots, a_{i_k} \rangle$ activities are mutually compatible.



This way one by one, one can replace a non-proper activity in A by a proper activity. Thus, there exists an optimal solution A^* where all the activities are proper activities.

(b) An $O(n^2)$ algorithm is easy.

Step 0: $S' = \phi$

Step 1: Select the activity with smallest duration. (takes $O(n)$ time)
Let $\bar{a}$ be the activity with the smallest duration.

Step 2: $S' = S' \cup \{\bar{a}\}$; $S = S - \{\bar{a}\}$

Step 3: Remove all activities from S which Contains $\bar{a}$.

Step 4: If $S \neq \phi$, repeat steps 1 through 4.

The process can be repeated $O(n)$ times & each iteration costs $O(n)$ time.

(c) Starting from a proper activity of S', the greedy algorithm is as follows:

Step 0: $A(x) = x$; current $= x$;

Step 1: Find the next clockwise compatible to x interval y.

Step 2: If $x = y$ stop. we are done.

Step 3: If x & y are not compatible, then stop.

Step 4: current $= y$, $A(x) = A(x) \cup \{y\}$. Repeat steps 1-4.

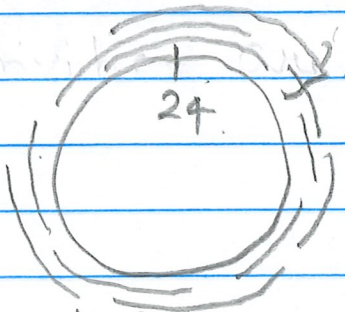
The above steps can be implemented in O(n) time once the endpoints of S' activities are ordered cyclically. (Endpoints are sorted)

(d) In the figure of my assignment.

$|A^*(a)| = 3$ & $|A^*(b)| = 4$

Note: both a & b are proper activities.

(e) Let B_0 be the set of all the activities that contain hour 24 in its activity time.



Let $B = S' - B_0$. We can treat B as a set of activity intervals on a line.

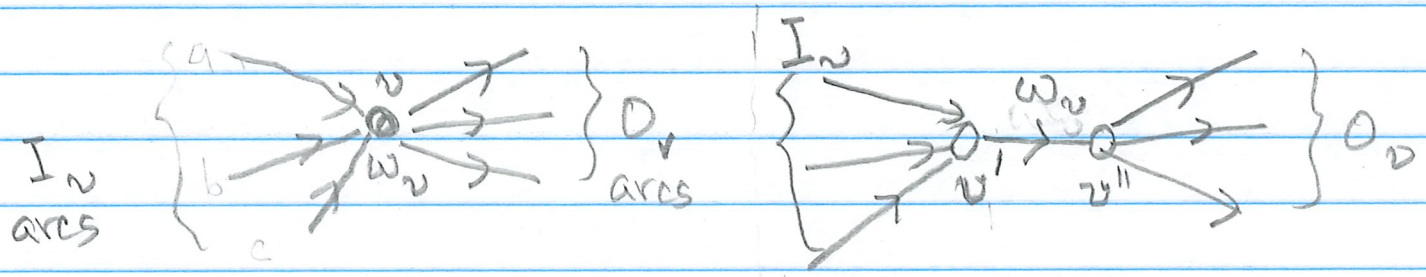
We know one can determine the set of largest compatible active intervals of B. Let B^* denote the optimal set. From B_0 only one compatible activity can be selected at most. Therefore, for any proper activities a & b, $|A^*(a)|$ & $|A^*(b)|$ can differ by at most one

(f) $O(n^2)$ algorithm is straightforward.

Step 1 : For every $a \in S'$, Compute $A^*(a)$.
Step 2 : Select the arc x for whom $|A^*(x)|$ is the largest.

We have seen that $A^*(a)$ for any proper activity a, can be determined in $O(n)$ time when the endpoints of S' are available in sorted order

2. We can modify each node v of $G=(V,E)$ with weight (w_v) as follows.



v is split into two nodes v' & v'' with a directed edge from v' to v'' & the cost of the edge is w_v (could be negative).

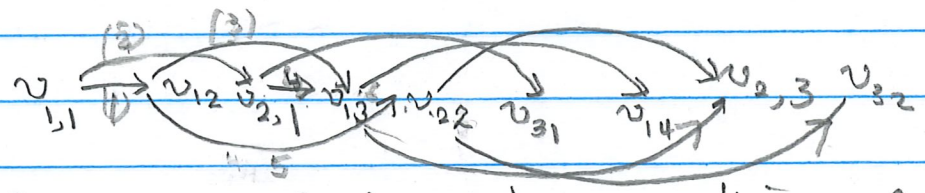
The new graph $G' = (V', E')$ where is

$$V' = V \cup \{v_i', v_i'', i = 1, 2, \ldots n\}$$
$$E' = E \cup \{(v_i', v_i''), i = 1, 3 \ldots, n\}.$$

We can now apply Bellman Ford algorithm (from an arbitrary vertex) to check if relaxations after $|V'|-1$ iterations are still improving the shortest path costs. The running time is $O(|V'|(|V'|+|E'|))$.

3. The graph is a DAG.
We first order the vertices topologically.
It takes $O(|V|+|E|)$ time.

We order the edges going from the left to right as follows



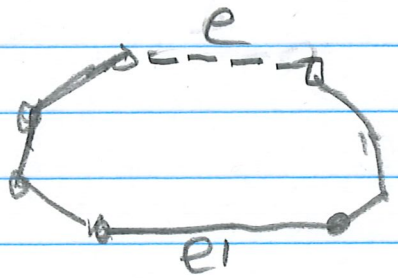
Bellman & Ford relaxes the edges from the left to the right. The shortest path tree from $v_{1,1}$ will thus be created in $O(|V|+|E|)$ time.

There is no problem, even when the edges are negative.

4 (a) Both the algorithms will do fine.

The cut property & the cycle property are still valid.

(b) Suppose it is otherwise. Let T_1 & T_2 be two msts of $G=(V,E)$ where the edge weights are all distinct. Let T_1 be the mst constructed by Kruskal's algorithm. Let e be the first edge which is selected in T_1 but not in T_2. Now adding e to T_2 will realize a cycle & $\exists$ an edge e' in the cycle whose cost is larger than e. Therefore $T_2 - \{e'\} \cup \{e\}$ will realize a spanning tree with lower cost. which is not possible.



5. We first preprocess the graph as follows.

Step 1: Find the shortest path costs from s to all the vertices in G.

Use Dijkstra once : Cost : $O(|V|\log|V| + |E|\log|V|)$

Step 2: Find the shortest path costs from all the vertices to the destination node t.

This can be done by reversing the direction of each arc & then compute the shortest path costs from t to all the vertices in the reversed graph. Dijkstra's algorithm is applied.

Step 3. For each edge $e=(u,v)$ compute $dist(s,u) + dist(v,t)$

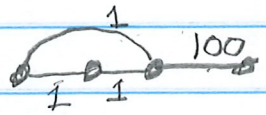
Select the smallest cost shortest path.

This step costs: $O(|E|)$

∴ Total costs : $O(|V|\log|V| + |E|\log|V|)$

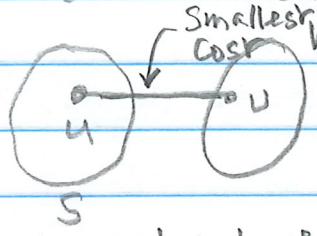
5.9 of the text Most of the questions are easy to answer.

a) False



b) True From cycle property

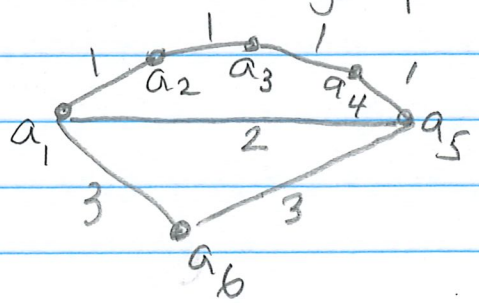
c) True From cut property. Let (u,v) has the smallest cost. Consider $S=\{u\}$ &

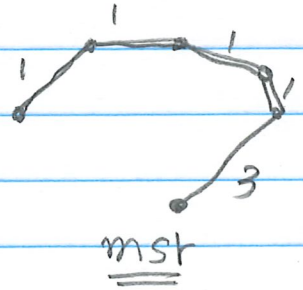


d) True From cycle property

e) True Using the cut property.

f) False: Consider the graph

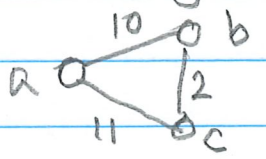




mst

(a_1, a_5) is the lightest edge in the cycle $\langle a_1, a_5, a_6 \rangle$

g) False.



shortest path tree from a

h) False; like above example.

i) True;

j) True