CECS 528, LO10 Assessment, April 24th, 2024, Dr. Ebert

NO NOTES, BOOKS, ELECTRONIC DEVICES, OR INTERPERSONAL COMMUNICATION ALLOWED. Submit each solution on a separate sheet of paper.

Problems

LO6. The following pertains to a correctness-proof outline for the Unit Task Scheduling (UTS) algorithm. Let $S = (a_1, t_1), \ldots, (a_m, t_m)$ represent the tasks that were selected by the algorithm for scheduling, where a_i is the task, and t_i is the time that it is scheduled to be completed, $i = 1, \ldots, m$. Moreover, assume that these tasks are ordered in the same order for which they appear in the sorted order. Let S_{opt} be an optimal schedule which also consists of task-time pairs. Let k be the first integer for which $(a_1, t_1), \ldots, (a_{k-1}, t_{k-1})$ are in S_{opt}, but $(a_k, t_k) \notin S_{\text{opt}}$ because a_k is scheduled by S_{opt}, but at time $t \neq t_k$.

(a) Explain why $t < t_k$. Assume that $t > t_k$ and explain why this creates a contradiction.
Solution. If $t > t_k$, then UTS Algorithm would have scheduled a_k at t since it looks for an open time that is closest to its deadline $d_k > t$.

(b) Assume that S_{opt} has scheduled some task a at time t_k. Explain why

$$\hat{S}_{\text{opt}} = S_{\text{opt}} - \{(a_k, t), (a, t_k)\} + \{(a_k, t_k), (a, t)\}$$

is a valid schedule. In words, the new schedule swaps schedule times for a_k and a. Explain why this does not create a scheduling problem for either task.
Solution. There is no conflict since a_k is moved back to $t_k < d_k$ while a is moved forward to a time $t < t_k < d$ where d is its deadline.

(c) Continuing in this manner we eventually arrive at an optimal schedule S_{opt} for which $S \subseteq S_{\text{opt}}$. Moreover, explain why it is not possible for S_{opt} to possess a task-time pair (a, t) that is *not* a member of S. Assuming it did have such a pair, what contradiction arises?
Solution. If such a existed then, when algorithm encountered a, it would have been able to schedule a at t; since no other previous task was scheduled at t.

LO7. Do the following.

(a) The dynamic-programming algorithm that solves the `Runaway Traveling Salesperson` optimization problem (Exercise 30 from the Dynamic Programming Lecture) defines a recurrence for the function $\mathrm{mc}(i, A)$. In words, what does $\mathrm{mc}(i, A)$ equal? Hint: do *not* write the recurrence (see Part b). Hint: we call it "Runaway TSP" because the salesperson does *not* return home.

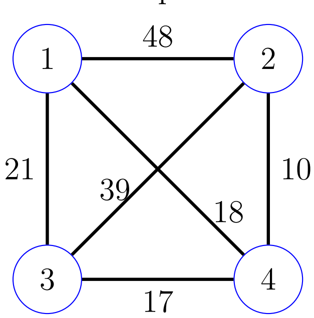
Solution. $\mathrm{mc}(i, A)$ gives the minimum cost of any path that starts at vertex i and must visit every vertex in A.

(b) Provide the dynamic-programming recurrence for $\mathrm{mc}(i, A)$.

Solution.

$$mc(i,A) = \begin{cases} 0 & \text{if } A = \emptyset \\ c(i,j) & \text{if } A = \{j\} \\ \min\limits_{j \in A} \left(c(i,j) + mc(j, A - \{j\}) \right) \end{cases}$$

(c) Apply the recurrence from Part b to the graph below in order to calculate $\mathrm{mc}(1, \{2, 3, 4\})$ Show all the necessary computations and use the solutions to compute an optimal path for the salesperson.



$$mc(1,\{2,3,4\}) = \min\left(48 + mc(2,\{3,4\}),\ 21 + mc(3,\{2,4\}),\ 18 + mc(4,\{2,3\})\right)$$
$$= \min(48+27,\ 21+27,\ 18+49) = 48$$

Solution.

$$mc(2,\{3,4\}) = \min\left(39 + mc(3,\{4\}),\ 10 + mc(4,\{3\})\right) = 27$$

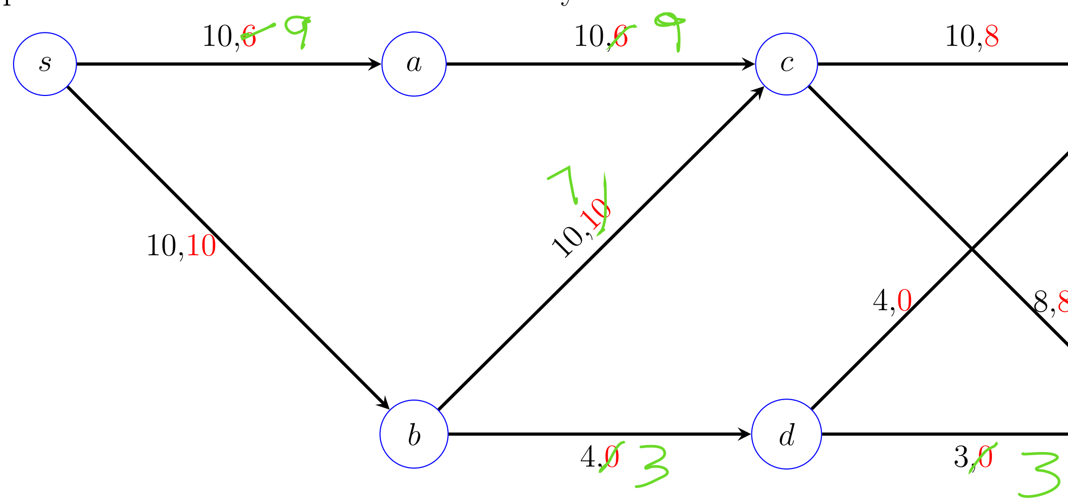
$$mc(3,\{2,4\}) = \min\left(39 + mc(2,\{4\}),\ 17 + mc(4,\{2\})\right) = \min(49, 27) = 27$$

$$mc(4,\{2,3\}) = \min\left(10 + mc(2,\{3\}),\ 17 + mc(3,\{2\})\right) = \min(10+39,\ 17+39) = 49$$

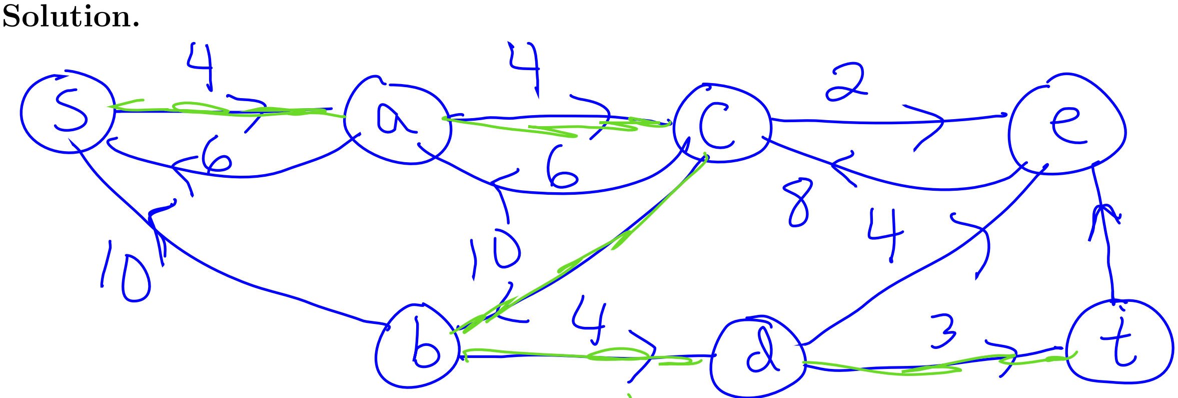
Optimal Path: $P = 1, 3, 4, 2$

LO8. A flow f (2nd value listed on each edge) has been placed in the network G below.

(a) Draw the residual network G_f and use it to determine an augmenting path P. Highlight path P in the network so that it is clearly visible.



Solution.



P = s, a, c, b, d, t

$C_{min} = \min(4, 4, 10, 4, 3) = 3$

(b) In the original network, cross out any flow value that changed, and replace it with its updated value from $f_2 = \Delta(f, P)$.

Solution. See above.

(c) What one query can be made to a `Reachability` oracle to determine if f_2 is a maximum flow for G? Hint: three inputs are needed for the `reachable` query function. Clearly define each of them.

Solution. `reachable`(G_f, s, t).

LO9. Answer the following.

(a) Provide the definition of what it means to be a mapping reduction fromm problem A to problem B.

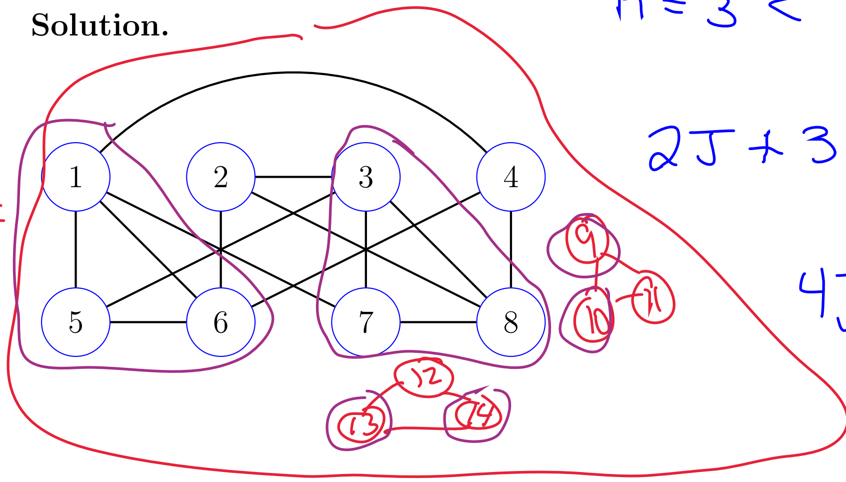
Solution. See Turing and Mapping Reducibility lecture notes.

(b) Suppose $(G, k = 3)$ is an instance of the `Vertex Cover` decision problem, where G is drawn below. Draw $f(G, k)$, where f is the mapping reduction from `Vertex Cover` to the `Half Vertex Cover` decision problem.

$k = 3 < \frac{8}{2} = 4$

Solution.

$f(G,k) =$



$2J + 3 = \frac{8 + 3J}{2} \Rightarrow$

$4J + 6 = 8 + 3J$

$J = 2$

(c) Verify that f is valid for input (G, k) in the sense that both (G, k) and $f(G)$ are either both positive instances or both negative instances of their respective problems. Defend your answer. Hint: it takes two vertices to cover each edge of a triangle in a graph.

Solution. We need at least 4 vertices to cover the edges of G since G has two disjoint Δs. Hence, (G,K) is negative for VC. Also, f(G,K) is negative for HVC since f(G,K) has a half-VC iff the original graph has a cover of size 3, which is false. 7 = 4+3

LO10. Do the following.

(a) An instance of `Set Cover` is a triple $(\mathcal{S}, m, k)$, where $\mathcal{S} = \{S_1, \ldots, S_n\}$ is a collection of n subsets, where $S_i \subseteq \{1, \ldots, m\}$, for each $i = 1, \ldots, n$, and a nonnegative integer k. The problem is to decide if there are k subsets $S_{i_1}, \ldots, S_{i_k}$ for which

$$S_{i_1} \cup \cdots \cup S_{i_k} = \{1, \ldots, m\}.$$

Verify that $(\mathcal{S}, m, k)$ is a positive instance of `Set Cover`, where $m = 9$, $k = 4$, and

$$\mathcal{S} = \{\{1, 3, 5, 8\}, \{3, 7, 9\}, \{2, 4, 5\}, \{2, 6, 7\}, \{6\}, \{2, 4, 7, 9\}, \{1, 3, 7\}, \{4, 5\}\}.$$

Solution.

$$\{1, 3, 5, 8\} \cup \{2, 4, 5\} \cup \{2, 6, 7\} \cup \{3, 7, 9\} = \{1, 2, \ldots, 9\}.$$

(b) For a given instance $(\mathcal{S}, m, k)$ of `Set Cover` describe a certificate in relation to $(\mathcal{S}, m, k)$.
Solution. A certificate for instance $(\mathcal{S}, m, k)$ is a subset $\mathcal{A} \subseteq S$ of size k.

(c) Provide a semi-formal verifier algorithm that takes as input i) an instance $(\mathcal{S}, m, k)$ of `Set Cover`, ii) a certificate for $(\mathcal{S}, m, k)$ as defined in part a, and decides if the certificate is valid for $(\mathcal{S}, m, k)$.

Solution. Initialize array $a = \underbrace{[0, 0, \ldots, 0]}_{\text{size } m+1}$

For each $A \in \mathcal{A}$

  For each $i \in A$

    $a[i] = 1$.

Return $\forall_{i \in \{1, \ldots, m\}} (a[i] = 1)$

(d) Provide size parameters that may be used to measure the size of an instance of `Set Cover`.

Solution. m is a bound on the size of any subset of $\mathcal{S}$ while $n = |\mathcal{S}|$ bounds the number of sets.

(e) Use the size parameters from part d to describe the running time of your verifier from part c. Defend your answer in relation to the algorithm you provided for the verifier.

Solution.

~~Iteration i of~~ The outer loop requires $k = O(n)$ iterations, while the inner loop requires $|A_i| = O(m)$ iterations to update the array which takes $O(1)$ steps. Therefore, the number of steps equals $O(mn)$ which is quadratic in the size parameters.