CECS 528, Learning Outcome Assessment 7, 3/19/2025, Dr. Ebert

Problem

LO3. Solve each of the following problems.

(a) Recall the use of the disjoint-set data structure for the purpose of improving the running time of the `Unit Task Scheduling (UTS)` algorithm. For the set of tasks

Task	a	b	c	d	e	f
Deadline	3	5	5	5	5	4
Profit	60	50	40	30	20	10

show the M-Tree forest after it has been inserted (or at least has attempted to be inserted in case the scheduling array is full). Notice that the earliest deadline is 1, meaning that the earliest slot in the schedule array has index 1. Hint: to receive credit, your solution should show six different snapshots of the M-Tree forest.

(b) Demonstrate the greedy algorithm for `Unit-Task Scheduling (UTS)` for the `UTS` instance shown below. Make sure to show the sorted order.

Task	a	b	c	d	e	f	g	h	i	j	k	l
Deadline	5	4	6	2	6	8	4	2	2	7	3	5
Profit	40	80	40	90	70	10	80	80	70	90	90	10

LO4. Answer the following.

(a) The dynamic-programming algorithm that solves the `Edit Distance` optimization problem defines a recurrence for the function $d(i, j)$. In words, what does $d(i, j)$ equal? Hint: do *not* write the recurrence (see Part b).

See Notes

(b) Provide the dynamic-programming recurrence for $d(i, j)$.

See Notes

(c) Apply the recurrence from Part b to the words u = baccbb and v = abbbcc. Show the matrix of subproblem solutions and use it to provide an optimal solution.

d(i,j)

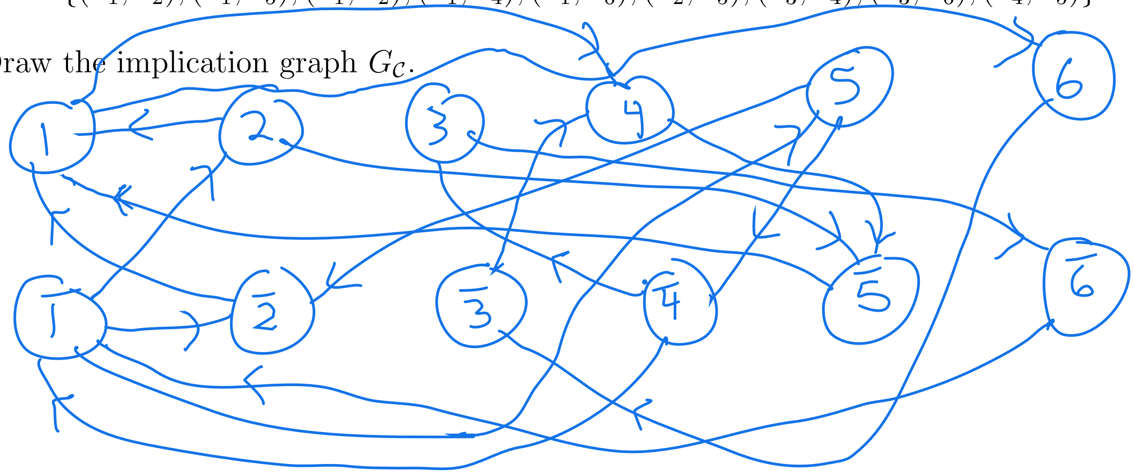
i \ j	λ	a	b	b	b	c	c
λ	0	1	2	3	4	5	6
b	1	↖1	↖1	↖←2	↖←3	←4	←5
a	2	↖1	↖↑←2	↖2	↖←3	↖←4	↖←5
c	3	↑2	↖2	↖↑←3	↖3	↖3	↖←4
c	4	↑3	↖↑3	↖3	↖↑←4	↖3	↖3
b	5	↑4	↖3	↖3	↖3	←4	↖↑4
b	6	↑5	↖↑4	↖3	↖3	↖4	↖↑←5

1. + c   baccbbc
2. b → c   baccbcc
3. c → b   bacbbcc
4. c → b   babbbcc
5. − b $\boxed{\text{abbbcc}}$

LO5. Consider the 2SAT instance

$$\mathcal{C} = \{(x_1, \overline{x}_2), (x_1, x_5), (x_1, x_2), (\overline{x}_1, x_4), (\overline{x}_1, x_6), (\overline{x}_2, \overline{x}_5), (x_3, x_4), (\overline{x}_3, \overline{x}_6), (\overline{x}_4, \overline{x}_5)\}.$$

(a) Draw the implication graph $G_{\mathcal{C}}$.



(b) Perform the `Improved 2SAT` algorithm by computing the necessary reachability sets. Use numerical order (in terms of the variable index) and positive literal before negative literal when choosing the reachability set to compute next. Draw the resulting reduced 2SAT instance whenever a consistent reachability set is computed. Either provide a final satisfying assignment for $\mathcal{C}$ or indicate why $\mathcal{C}$ is unsatisfiable.

$R_{x_1} = \{x_1, x_4, \overline{x}_5, x_6, \overline{x}_3\}$ is consistent $\alpha_{R_{x_1}} = (x_1 = 1, x_3 = 0, x_4 = 1, x_5 = 0, x_6 = 1)$

Reduced Graph: 2 $\quad$ $R_{x_2} = \{x_2\}$ is consistent

$\overline{2}$ $\quad$ $\alpha_{R_{x_2}} = (x_2 = 1)$

Satisfying Assignment: $\alpha = (x_1 = 1, x_2 = 1, x_3 = 0, x_4 = 1, x_5 = 0, x_6 = 1)$

(c) Suppose 2SAT instance $\mathcal{C}$ has three variables and, when running the original 2SAT algorithm the answer to each oracle query is shown in the table below. Is $\mathcal{C}$ satisfiable? If yes, provide a satisfying assignment for $\mathcal{C}$. If not, explain why.

Oracle Query	**Answer**
reachable$(G_{\mathcal{C}}, x_1, \overline{x}_1)$	Yes
reachable$(G_{\mathcal{C}}, \overline{x}_1, x_1)$	No
reachable$(G_{\mathcal{C}}, x_2, \overline{x}_2)$	Yes
reachable$(G_{\mathcal{C}}, \overline{x}_2, x_2)$	Yes
reachable$(G_{\mathcal{C}}, x_3, \overline{x}_3)$	No
reachable$(G_{\mathcal{C}}, \overline{x_3}, x_3)$	Yes

$\mathcal{C}$ is unsatisfiable because $G_{\mathcal{C}}$ has an inconsistent cycle passing through x_2 and $\overline{x}_2$.

LO6. Answer the following.

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

See Lecture Notes

(b) Given `Subset Sum` instance ($S = \{4, 6, 7, 16, 21, 25, 30, 37, 42\}, t = 96$), provide $f(S,t)$, where f : `SS` $\to$ `SP` is the mapping reduction from `SS` to `Set Partition` described in lecture.

$M = \sum_{s \in S} s = 188 \quad t = 96 > \frac{188}{2} = 94 \Rightarrow J = 2t - M = 4$

$f(S,t) = S' = S \cup \{J\} = S \cup \{4\} = \{4, 4, 6, 7, 16, 21, 25, 30, 37, 42\}$

(c) Verify that f is valid for input (S,t) from part b in the sense that both (S,t) and $f(S,t)$ are positive instances of their respective problems. Make sure to provide the solutions for both problem instances.

$S \supseteq A = \{4, 25, 30, 37\}$ satisfies $\sum_{a \in A} a = 96 = t$.

Hence, (S,t) is positive for SS.

Also, A and $B' = \{4, 6, 7, 16, 21, 42\}$ form a set partition for S' since $\sum_{a \in A} a = \sum_{b \in B'} b = 96$ and $A \cap B' = \emptyset$ and $A \cup B' = S'$. Note: here we assume that the two 4's in S' are distinct numbers. For example, $4_1, 4_2$ are two different numbers.

LO7. An instance of the `Odd Cycle` decision problem is a simple graph $G = (V, E)$. The problem is to decide if G has a an **odd cycle**, i.e. a sequence of $k+1$ vertices for which i) $k \geq 3$ is odd, ii) the first and last vertices are identical, iii) each vertex is adjacent to the vertex that immediately follows it, and iv) with the exception of the first/last vertex, no other vertex appears more than once. For example, the graph shown below has a 3-cycle via the sequence $C = c, d, e, c$.

(a) For a given instance G of `Odd Cycle`, describe a certificate in relation to G.

C is a sequence of vertices $(V_1, \ldots, V_K, V_{K+1})$ from V where $V_1 = V_{K+1}$ and K is odd.

(b) Provide a semi-formal verifier algorithm that takes as input i) an instance G of `Odd Cycle`, ii) a certificate for G as defined in part a, and decides if the certificate is valid for G.

VT is a vertex table initialized as empty.

For each $i \in \{1, \ldots, K\}$

If $(V_i, V_{i+1}) \notin E$, return 0.

If $(V_i \in VT)$ return 0.

Insert V_i into VT.

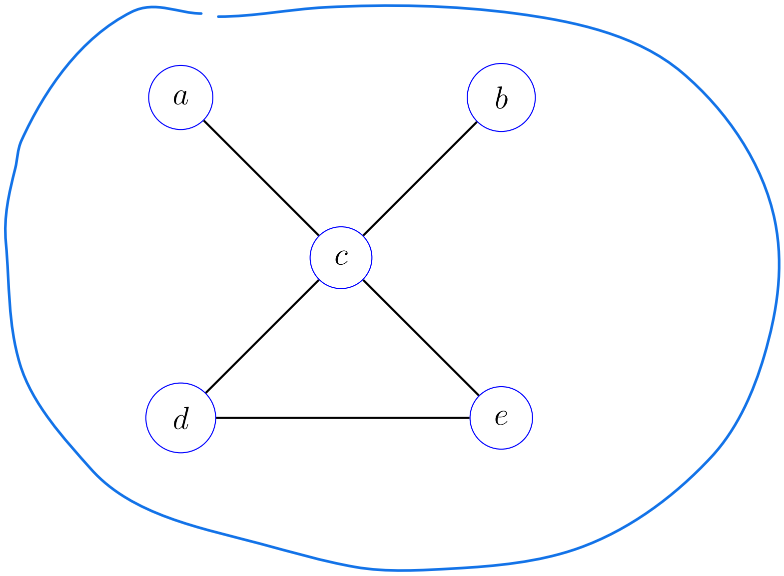
Return 1.

(c) Provide the two size parameters for `Odd Cycle`.

$m = |E|$ $n = |V|$

(d) Use the size paramters from part c to provide a big-O bound on the verifier's running time. Justify your answer.

The For-loop requires $K = O(n)$ iterations with each iteration requiring a membership query to the set of edges. Each query requires $O(1)$ steps to answer if we store the edges in a hash table which can be created in $O(m)$ steps. ∴ total steps equals $O(m+n)$.



a
b
c
d
e