CECS 528, Exam 2 Solutions, Friday Spring 2026, Dr. Ebert

IMPORTANT: READ THE FOLLOWING DIRECTIONS. Directions,

- For each LO and Additional problem, write your solutions to all parts using **ONE SHEET OF PAPER ONLY (BOTH FRONT AND BACK)**. Write **NAME** and **PROBLEM NUMBER** on each sheet.
- Use **SEPARATE SHEETS** for different LO/Additional problems.
- **-5 POINTS FOR EACH PROBLEM THAT DOES NOT FOLLOW THE ABOVE RULES**
- Make up LO solutions may be written on the same sheet.

Unit 2 LO Problems

LO5. Do the following.

(a) The dynamic-programming algorithm that solves the `Runaway Traveling Salesperson` optimization problem defines a recurrence for the function $\text{mc}(i, A)$. State in words the meaning of $\text{mc}(i, A)$ and provide its recurrence. (7 pts)

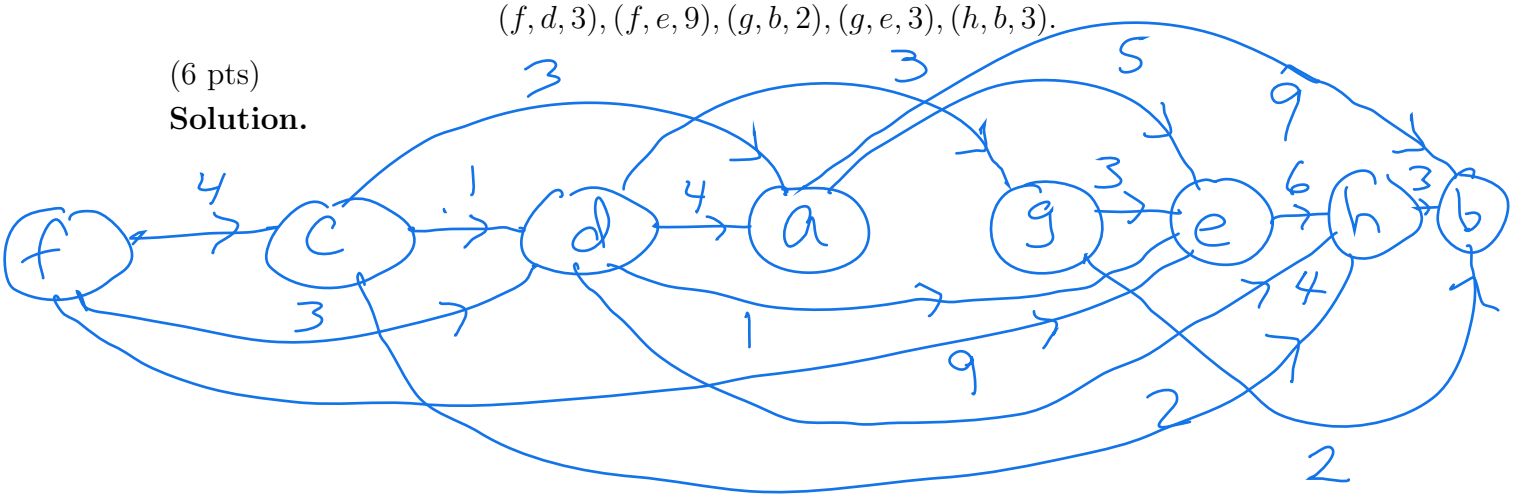
Solution. See solution to Exercise 16 in Dynamic Programming lecture.

(b) Draw the vertices and edges of the weighted DAG $G = (V, E, c)$ in a linear left-to-right manner so that the vertices are topologically sorted, meaning, if $(u, v) \in E$, then u appears to the left of v. The vertices of G are $\{a, b, \ldots, h\}$, while the weighted edges of G are

$$(a, b, 9), (a, e, 5), (c, a, 3), (c, d, 1), (c, h, 2), (d, a, 4), (d, e, 1), (d, g, 3), (d, h, 4), (e, h, 6), (f, c, 4),$$
$$(f, d, 3), (f, e, 9), (g, b, 2), (g, e, 3), (h, b, 3).$$

(6 pts)

Solution.



(c) Provide the dynamic-programming recurrence for the `Single Source DAG` distance problem. Hint: for each edge $e = (u, v) \in E$, assume that $c(u, v)$ denotes the cost of traversing e. (6 pts)

Solution. See Dynamic Programming lecture.

(d) Use the recurrence from part c) to compute the distances from the source vertex to every vertex in G. Show all work. Hint: the source vertex is *not* vertex a. (6 pts)

Solution.

$d(f,f) = 0 \quad d(f,c) = 4$

$d(f,d) = \min(d(f,c)+1,\ d(f,f)+d(f,d)) = 3$

$d(f,a) = \min(d(f,c)+3,\ d(f,d)+4) = 7$

$d(f,g) = d(f,d)+3 = 6$

$d(f,e) = \min(d(f,a)+5,\ d(f,g)+3,\ d(f,d)+1,\ d(f,f)+9) = 4$

$d(f,h) = \min(d(f,c)+2,\ d(f,d)+4,\ d(f,e)+6) = 6$

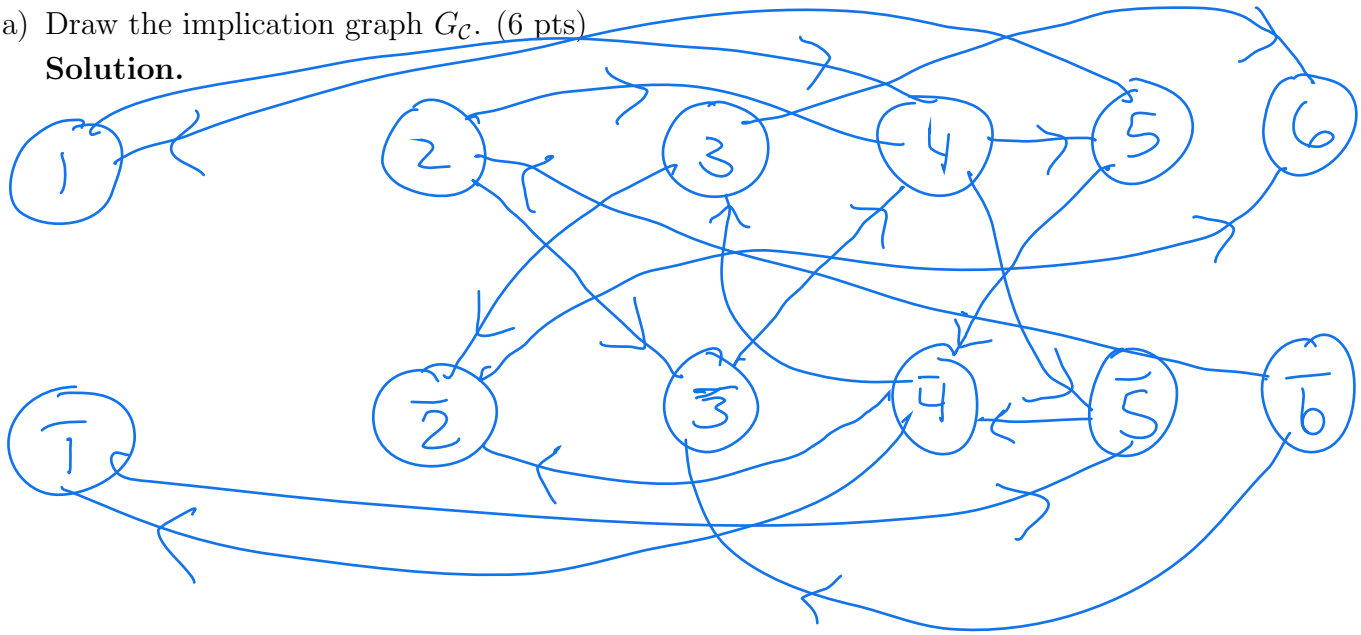
$d(f,b) = \min(d(f,a)+9,\ d(f,g)+2,\ d(f,h)+3) = 8$

LO6. Do the following for the `2SAT` instance

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

(a) Draw the implication graph $G_{\mathcal{C}}$. (6 pts)

Solution.



(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. (12 pts)

Solution. $R(x_1) = \{x_1, x_5, x_4, \overline{x}_4, \overline{x}_1, x_3, \overline{x}_2, x_6\}$ is inconsistent.

$R(\overline{x}_1) = \{\overline{x}_1, \overline{x}_2, x_3, \overline{x}_4, \overline{x}_5, x_6\}$

$\alpha = (x_1 = x_2 = x_4 = x_5 = 0, x_3 = x_6 = 1)$ satisfies $\mathcal{C}$

(c) Given `2SAT` instance $\mathcal{C}$, if the reachability set $R(x_2)$ is consistent and contains the literal $\overline{x}_3$, then provide the reasoning for why the assignment $\alpha_{R_{x_2}}$ induced by $R(x_2)$ satisfies the clause (x_3, x_4). (7 pts)

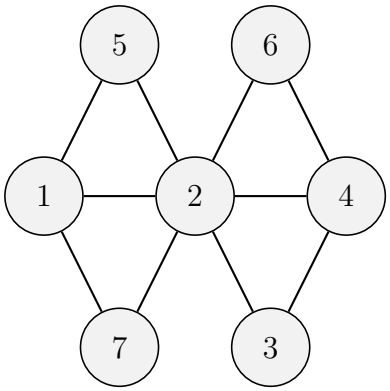
Solution. Clause (x_3, x_4) implies that $G_{\mathcal{C}}$ has the edge $(\overline{x}_3, x_4)$. Thus, by transitivity, there is a path from x_2 to x_4 since there is a path from x_2 to $\overline{x}_3$. Therefore, $\alpha_{R_{x_2}}(x_4) = 1$ and the clause is satisfied.

LO7. Answer the following.

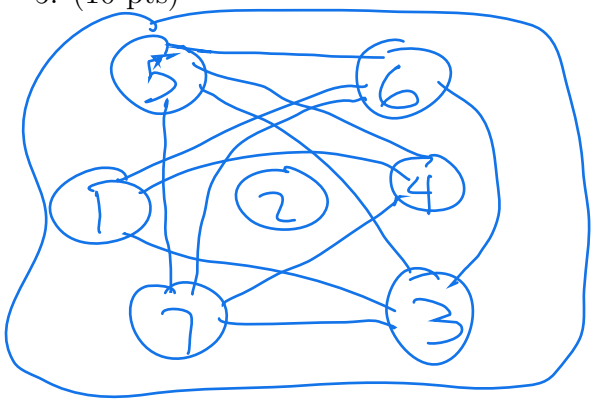
(a) Provide the definition of what it means to be a mapping reduction from decision problem A to decision problem B. (5 pts)

Solution. See Mapping Reducibility lecture.

(b) For the mapping reduction f : `IS` $\rightarrow$ `Clique`, draw $f(G, k)$ for the `Independent Set` instance whose graph is shown below, and for which $k = 5$. (10 pts)



$f(G, k) = G'$



(c) Verify that both (G, k) and $f(G, k)$ are either both positive instances, or are both negative ones. **Provide justification for your evaluation of both problem instances**. (10 pts)

Solution. By visual inspection, the largest IS in G is $\{3, 5, 6, 7\}$. Therefore, $(G, k = 5)$ is a negative instance of `IS`. Again by visual inspection, we see that $\{3, 5, 6, 7\}$ is the largest clique in G' and so $(G', k = 5)$ is a negative instance of `Clique`.

LO8. Do the following.

(a) An instance of `Set Packing` is a positve integer $k > 0$ and a collection of sets $\mathcal{S}$, where each set $C \in S$ is a subset of $\{1, \ldots, n\}$ for some $n \geq 1$. The problem is to decide if there are k sets $C_1, \ldots, C_k$ in $\mathcal{S}$ that are **pairwise disjoint** meaning that, for every $1 \leq i < j \leq n$, $C_i \cap C_j = \emptyset$. Thus a natural certificate for `Set Packing` is an array a of length k, where $a[i] \in \mathcal{S}$, for $i = 1, \ldots, k$. To show that `Set Packing` is an `NP` problem, do the following.

i. Provide the pseudocode for a verifier program for `Set Packing` that takes as inputs $\mathcal{S}$, k, and a and returns 1 iff the members of a are pairwise disjoint. (8 pts)
Solution. It remains to show that the $a[i]$ sets are pairwise disjoint. This will be true iff each item in the union of the subsets of a belongs to exactly one subset. This can be checked by making a table and ensuring that, for all $i \in \{1, \ldots, k\}$, no encountered item in $a[i]$ exists in the table.

Initialize table T to be an empty.
For each $i \in \{1, \ldots, k\}$,
  For each item $x \in a[i]$,
    If $x \in T$, return 0.
    Insert x into T.
Return 1.

ii. Given that the size parameters for `Set Packing` are $m = |\mathcal{S}|$ and n, determine the big-O number steps required by your verifier. Defend your answer. Hint: your analysis should accurately address the number of steps required to check the intersection of two sets. (5 pts)
Solution. $O(n)$ steps are needed to define T. Also, the outer loop requires at most $k = O(m)$ iterations and, for each iteration, the inner loop requires $k = O(n)$ iterations with at most two $O(1)$ table operations per inner iteration. Therefore, the total number of steps is $O(n+mn) = O(mn)$ steps which is quadratic in the size parameters, and thus the verifier proves that `SP` is in `NP`.

(b) Classify each of the following problems as being in `P`, `NP`, or co-`NP`. Note: at least three correct answers is necessary for passing this part of LO8. (3 points each).

Solution. NP, P, P, co-NP. For a, to find an x may require $O(c)$ steps which is exponential in the size of the problem instance. For b, `Longest Increasing Subsequence` can be solve in a polynomial number of steps by constructing a DAG and computing the longest path in the graph (see AP problem H in the Dynamic Programming lecture). For c, the `LCS` decision problem in solvable in a polynomial number of steps via dynamic programming (see Exercise 5 in the Dynamic Programming lecture).

i. An instance of `Quadratic Congruences` is a triple of positive integers (a, b, c) and the problem is to decide if there is a positive integer $x < c$ for which $x^2 \bmod b = a$. Hint: the size of the problem instance is $\lfloor \log(\max(a, b, c)) \rfloor + 1$.

ii. An instance of `Increasing Subsequence` is an array a of size n and a positive integer k the problem is to decide if there are indices $0 \leq i_1 < i_2 < \cdots < i_k < n$ for which $a[i_1] < a[i_2] < \cdots < a[i_k]$.

iii. An instance of `Longest Common Subsequence` is a pair of words (w_1, w_2) each over some alphabet Σ, and a nonnegative integer $k \geq 0$. The problem is to decide if

there is a word v over Σ that has length k and appears in both w_1 and w_2, where the appearance can be non-contiguous. For example, the word `coin` appears non-contiguously in the word **co**nctentat**i**o**n**.

iv. An instance of the `Strongly Cyclic` decision problem is a pair (G, k) where $G = (V, E)$ is a directed graph and k is a nonnegative integer. The problem is to decide if G always has at least one cycle after k vertices (and the edges that are incident with them) are removed from G, regardless of what k vertices are selected.

Additional Problems

A1. Do the following.

(a) Explain why class `P` is closed under complement, meaning that if decision problem $L \in$ `P`, then it is also true that $\overline{L} \in$ `P`. (8 pts)

Solution. Since $L \in$ `P`, there is a polynomial-step algorithm that decides L and can be modified by negating each of its return values, thus creating a new program that decides $\overline{L}$ in a a polynomial number of steps.

(b) Use the result of part a) to prove that `P` $\subseteq$ `NP` $\cap$ `co-NP`. (9 pts)

Solution. Since any polynomial-step algorithm that decides a problem L may serve as a polynomial-step verifier algorithm for L (and that uses an empty certificate), and thus $L \in$ `NP`. Similarly, by part a, $\overline{L} \in$ `NP`. Then by definition, $\overline{\overline{L}} = L \in$ co-`NP`. Therefore, `P` $\subseteq$ `NP` $\cap$ `co-NP`.

(c) Prove or provide a counterexample: if $\mathcal{C}$ is a satisfiable `2SAT` instance and the reachability set $R(x)$ is consistent for some variable x, then $\mathcal{C}$ will surely have a satisfying assignment α for which $\alpha(x) = 1$. (8 pts)

Solution. This is true since $\mathcal{C}$ is a satisfiable and thus there are only consistent cycles in $G_{\mathcal{C}}$. Thus, when performing the improved `2SAT` algorithm and starting with variable x, $R(x)$ is consistent (by assumption) which yields a partial satisfying assignment for $\mathcal{C}$. Next, remove from $G_{\mathcal{C}}$ all vertices and edges that have been assigned by α_{R_x} to get the reduced graph G'. This graph also only has consistent cycles and thus admits a satisfying assignment α'. Finally, combining α_{R_x} with α' provides a full satisfying assignment for $\mathcal{C}$ and which assigns x the value 1.

A2. Do the following.

(a) Provide a *recursive* implementation of the function

```
void print_path(int d0[ ][ ], int dn[ ][ ], int i, int j);
```

which has the effect of printing the optimal path from vertex i to vertex j, where $d0$ is the initial edge-cost matrix that begins the Floyd-Warshall algorithm, and dn is the final matrix of the Floyd-Warshall algorithm. For example, if $i = 2$, $j = 4$, and the optimal path is $P = 2, 5, 6, 7, 4$, then your algorithm should print

2 5 6 7 4 .

You may assume the existence of a `print()` function that is capable of printing an integer that is automatically surrounded by whitespace. (12 points)

Solution. We show how to implement the helper function

```
void print_path_helper(int d0[ ][ ], int dn[ ][ ], int i, int j,
    Boolean print_right_end_vertex);
```

which is the same function as print_path, except now it controls whether the right vertex is printed. This is necessary so that no vertex is printed more than once. Of course, this flag will be intially set to true by its calling wrapper function print_path.

```
void print_path_helper(int d0[ ][ ], int dn[ ][ ], int i, int j,
    Boolean print_right_end_vertex)

    if(d0[i][j] == dn[i][j]) //the path is a direct connection

        print(i).

        if(print_right_end_vertex)
            print(j).

    //The optimal path must have at least one intermediate vertex k
    int n = rows(d0).

    for(k=0; k < n; k++)
        if(dn[i][k] + dn[k][j] == dn[i][j])
            print_path_helper(d0,dn,i,k,false).
            print_path_helper(d0,dn,k,j,print_right_end_vertex).
}
```

(b) The `Half Independent Set (HIS)` decision problem consists of a simple graph $G = (V, E)$, and the problem is to decide if G has an independent set of size $|V|/2$. Without using any specific examples, define a valid mapping reduction from the `Independent Set` decision problem to `HIS`. In other words, given an instance $(G = (V, E), k)$ of `IS`, describe the procedure for determining the instance of `HIS` to which it maps. (13 pts)

Solution. Let $n = |V|$. If $k = n/2$, then $f(G, k) = G$ (already an `HIS` problem instance). If $k < n/2$, then $f(G, k) = G'$, where G' is G with $n - 2k$ additional isolated vertices which are needed to "pump up" the size of the graph so that there will be a Half IS iff the original graph as an IS of size k. If $k > n/2$, the graph order again needs to be increased but now the additional vertices should *not* increase the IS size (it is already too large). Therefore, add $2k - n$ additional vertices to G and make sure they are adjacent to each other and to every vertex in G.

Makeup Problems

LO1. Solve the following problems.

(a) Use the Master Theorem to determine the growth of $T(n)$ if it satisfies the recurrence $T(n) = 81T(n/3) + n^4 \log^2 n$. Defend your answer.

(b) Use the substitution method to prove that, if $T(n)$ satisfies

$$T(n) = 4T(n/2) + 6n^2,$$

then $T(n) = \Omega(n^2 \log n)$.

LO2. Solve the following problems.

(a) Use Strassen's products $P_1 = a(f-h) = af - ah$, $P_2 = (a+b)h = ah + bh$, $P_3 = (c+d)e = ce + de$, $P_4 = d(g-e) = dg - de$, $P_5 = (a+d)(e+h) = ae + ah + de + dh$, $P_6 = (b-d)(g+h) = bg + bh - dg - dh$, $P_7 = (a-c)(e+f) = ae - ce - cf + af$. to compute the matrix product

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} e & f \\ g & h \end{pmatrix}$$

Show all work.

(b) For the `Maximum Subsequence Sum (MSS)` divide-and-conquer algorithm described in the Divide and Conquer core exercises, provide the entire recursion tree for instance $a = 4, -3, 4, -2, 3, -4, 2, -3$. Label each node with the problem instance that is being solved at that stage of the recursion. Also, next to each node write its MSS value and (in case the node is internal) indicate if the MSS comes from the left subproblem, right subproblem, or middle of both subproblems. For example, iff the MSS for some subproblem equals 7 and that value came from the left subproblem, then write 7/L. Assume a base case occurs when the array has size equal to 1.

LO3. Do the following.

(a) Consider the FFT algorithm when applied to a polynomial $A(x)$ having degree $2^n - 1$. Provide the equation that relates $A(x)$ to the two subproblem polynomials $A_e(x)$ and $A_o(x)$. What are the degrees of these two polynomials? Based on this equation, why is it essential that, for even n, the nth roots of unity come in additive-inverse pairs?

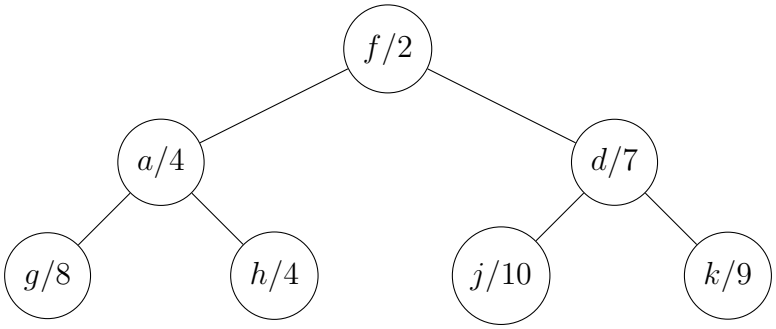
(b) If $p(x) = 1 - 3x + 4x^2 - 2x^3$, then compute $\text{DFT}^{-1}(p)$ using the IFFT algorithm. Show the entire recursion tree as was done in the lecture notes.

LO4. Do the following.

(a) The tree below shows the state of the binary min-heap at the beginning of some round of Prim's algorithm, applied to some weighted graph G. If G has edges

$$(a, f, 6), (d, g, 4), (f, g, 3), (f, h, 5), (f, k, 3),$$

then draw a plausible state of the heap at the end of the round.



(b) Do the following.

i. State the greedy choice that is being made in each round of Kruskal's algorithm.

ii. The weighted edges of a graph $G = (V, E)$ are

$$E = \{(1,2,11),(1,3,19),(1,4,15),(1,5,13),(1,6,8),(2,3,14),(2,4,17),$$

$$(2,5,16),(2,6,22),(3,4,10),(3,5,12),(3,6,18),(4,5,15),(5,6,20)\}.$$

For each round of Kruskal's algorithm applied to G, indicate the selection for that round and whether or not it is used in the final mst. After the final round, provide a drawing of the final output of the algorithm. Hint: you do *not* need to use a disjoint-set structure.