CECS 528, Exam 1, Friday Spring 2026, Dr. Ebert

Directions: Show all necessary work to receive full credit. Please number/letter your problem solutions and make sure your name is on each solution page. You may use both front and back of a page.

Unit 1 LO Problems (25 pts each)

LO1. Solve the following problems.

(a) Use the Master Theorem to determine the growth of $T(n)$ if it satisfies the recurrence $T(n) = 27T(n/3) + n^{\log_2 8}$. Defend your answer. (10 pts)

Solution. Since $n^{\log_3 27} = n^{\log 8} = n^3$, by Case 2 of the Master Theorem, we have $T(n) = \Theta(n^3 \log n)$.

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

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

then $T(n) = \Omega(n^2 \log n)$. (15 pts)

Solution. Inductive assumption. $T(k) \geq ck^2 \log k$ for all $k < n$ and some constant $c > 0$.

Then $T(n) = 4T(n/2) + n^2 \geq 4c\left(\frac{n}{2}\right)^2 \log\left(\frac{n}{2}\right) + n^2 =$

$cn^2 \log n - cn^2 + n^2 \geq cn^2 \log n \iff$

$-cn^2 + n^2 \geq 0 \iff cn^2 \leq n^2 \iff c \leq 1.$

LO2. Solve the following problems.

(a) What is its worst-case running time of Hoare's `Quicksort` and what must happen in order for it to occur? Explain and defend your answer using basic math. (13 pts)

Solution. Worst-case running time is $\Theta(n^2)$ which happens when, e.g., the pivot is always selected as the second-to-least element in the array. When this happens, and assuming a is of even size, all the a_{right} subarrays require a total of

$$\Theta(n + (n-2) + (n-4) + \cdots + 2) = \sum_{i=1}^{\frac{n}{2}} 2i = \Theta(n^2)$$

steps.

(b) Consider Karatsuba's algorithm which we'll call `multiply` for multiplying two n-bit binary numbers x and y, where we assume that n is even. Let x_L and x_R be the leftmost $n/2$ and rightmost $n/2$ bits of x respectively. Define y_L and y_R similarly. Let P_1 be the result of calling `multiply` on inputs x_L and y_L, P_2 be the result of calling `multiply` on inputs x_R and y_R, and P_3 the result of calling `multiply` on inputs $x_L + x_R$ and $y_L + y_R$. Then return the value $P_1 \times 2^n + (P_3 - P_1 - P_2) \times 2^{\frac{n}{2}} + P_2$. Demonstrate the algorithm (only at the root level of the recursion tree!) with $x = 39$ and $y = 11$. Hint: convert x and y to two binary numbers, each having the same (even) number of bits. (12 pts)

Solution.

$X = \underbrace{100}_{X_L}\underbrace{111}_{X_R} = (39)_2$

$y = \underbrace{001}_{y_L}\underbrace{011}_{y_R} = (11)_2$

$P_1 = x_L \cdot y_L = 4 \cdot 1 = 4 \quad P_2 = x_R \cdot y_R = (7)(3) = 21$

$P_3 = (x_L + x_R)(y_L + y_R) = (4+7)(1+3) = 11 \cdot 4 = 44.$

$P_1 \times 2^6 + (P_3 - P_1 - P_2) \times 2^3 + P_2 = (4)(64) + (44 - 21 - 4)(8)$

$+ 21 = 256 + (19)(8) + 21 = 256 + 152 + 21 = 429$

$= 39 \cdot 11$ ✓

$\therefore$

LO3. Do the following.

(a) If $p(x) = c_0 + c_1 x + \cdots + c_{n-1}x^{n-1}$ is an unknown polynomial of degree $n-1 = 2^d - 1$, and $\mathrm{DFT}_n(p) = (y_0, y_1, \ldots, y_{n-1})$, then how does one compute the coefficients $(c_0, c_1, \ldots, c_{n-1})$ of $p(x)$? (13 pts)

Solution. $(c_0, c_1, \ldots, c_{n-1}) = \mathrm{DFT}_n^{-1}(y_0, y_1, \ldots, y_{n-1})$.

(b) If $p(x) = 4 - 4x + 5x^2 - 5x^3$, then compute $\mathrm{DFT}(p)$ using the FFT algorithm. Show the entire recursion tree as was done in the lecture notes. (12 pts)

$DFT(P) = (0, -1+i, 18, -1-i)$

Verify using FFT

$DFT(4, -4, 5, -5) = (9, -1, 9, -1) + (1, i, -1, -i) \odot (-9, 1, -9, 1) = (0, -1+i, 18, -1-i)$

$DFT_2(4, 5) = (4, 4) + (1, -1) \odot (5, 5) = (9, -1)$

$DFT_2(-4, -5) = (-4, -4) + (1, -1) \odot (-5, -5) = (-9, 1)$

$DFT_1(4) = 4$ $DFT_1(5) = 5$ $DFT_1(-4) = -4$ $DFT_1(-5) = -5$

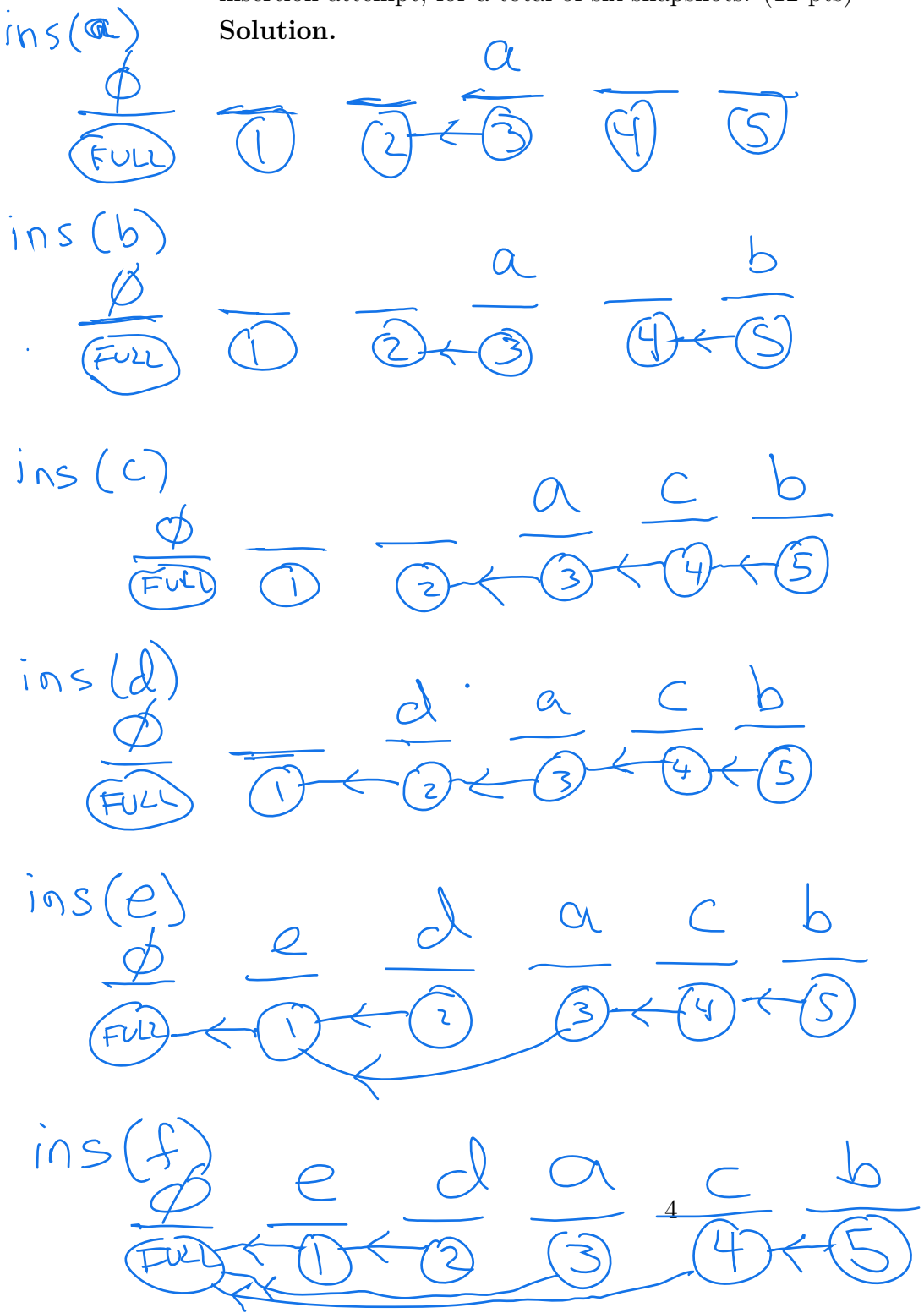
LO4. Do the following.

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

Task	a	b	c	d	e	f
Deadline Index	3	5	5	3	3	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). Note that the earliest possible deadline index is 1, meaning that the earliest slot in the schedule array has index 1. To receive credit, your solution should show the state of the scheduling array as well as MTree forest *after* each insertion attempt, for a total of six snapshots. (12 pts)

Solution.



(b) Given an instance $(\{(x_1, w_1, p_1), \ldots, (x_n, w_n, p_n)\}, M)$ of `Fractional Knapsack (FK)`, where (x_i, w_i, p_i) gives the weight w_i and profit p_i of item x_i, and M is the knapsack capacity, describe in general (do not give an example) what must be done in order to determine the knapsack payload that maximizes total profit. (7 pts)

Solution. The items should be sorted in decreasing order of **profit density**, i.e. p_i/w_i. Following this ordering, the greedy algorithm tries to place as much of the item in the knapsack as possible. The algorithm terminates once either the knapsack has no remaining capacity or the sorted array of items has been exhausted.

(c) For the `FK` instance having $M = 30$ and the following items

Item	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
Weight	4	8	8	8	9	2	1	6	6	9
Profit	27	18	11	10	19	24	23	18	23	15

provide the optimal knapsack payload, indicating how much of each item is placed in the knapsack, and the total profit earned. (6 pts)

Solution. Below is a revised table of items in which Profit is replaced with Profit Density.

Item	1	2	3	4	5	6	7	8	9	10
Weight	4	8	8	8	9	2	1	6	6	9
Profit Density	6.75,	2.25,	1.38,	1.25,	2.1	12	23	3	3.8	1.67

The optimal payload is 1 unit of item 7, 2 units of item 6, 4 units of item 1, 6 units of item 9, 6 u

6 units of item 8, 8 units of item 2, 3 units of item 5

Additional Problems

A1. Solve the following.

(a) Let G be a strongly connected directed graph with vertices $0, 1, \ldots, n-1$, and let parent be an array, where `parent`$[i]$ denotes the parent of i in the Dijsktra Distance Tree (DDT) that has vertex 0 as its root. Hence, `parent`$[0]$ = UNDEFINED, where UNDEFINED < 0 is some constant. Provide a *recursive* implementation of the function

```
void print_optimal_path(int v, int parent[ ])
```

that prints from left to right the unique path from the root 0 to vertex v in the DDT. You may assume access to the `print` function, where `print`(i) has no return value, but has the side effect of printing the number i, along with some white space so that the printed sequence of vertices will seem more readable. (13 pts)

Solution.

```
void print_optimal_path(int v, int parent[ ])
{
    if(v == 0)
        print(0);

    print_optimal_path(parent[v], parent);
    print(v);
}
```

(b) There are times when Strassen's algorithm may be used to efficiently multiply non-square matrices. For example, describe how Strassen's algorithm can be used to efficiently multiply a $kn \times n$ matrix with an $n \times kn$ matrix, where $k \geq 1$ is an arbitrary positive integer and n is a power of 2. Provide the big-O number of steps (in terms of both k and n) that is required by your method. (12 pts)

Solution. We may view the $kn \times n$ matrix as consisting of k $n \times n$ submatrices

$$\begin{pmatrix} A_1 \\ A_2 \\ \vdots \\ A_n \end{pmatrix},$$

while we may view the $n \times kn$ matrix as also consisting of the k $n \times n$ submatrices

$$\begin{pmatrix} B_1 & B_2 & \cdots & B_n \end{pmatrix}$$

Then the product of the two consists of a $kn \times kn$ matrix which may be viewed as consisting of a matrix of $n \times n$ submatrices, where submatrix $C_{ij} = A_i B_j$. Thus, using Strassen's algorithm to compute each of the k^2 $A_i B_j$ products, we get a number of steps equal to $O(k^2 n^{\log 7})$.

A2. Solve the following.

(a) Recall from the lecture notes that F_n is the $n \times n$ matrix for which

$$\mathrm{DFT}(a_0, a_1, \ldots, a_{n-1}) = F_n \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{n-1} \end{pmatrix}.$$

Provide the entries for F_6. (13 pts)

Solution. let $0 \leq r, s \leq 5$

$$(F_6)_{rs} = e^{(2\pi i r s)/6}.$$

(b) An instance of the `Set Cover` decision problem is a pair $(\mathcal{S}, m)$, where $\mathcal{S} = \{S_1, \ldots, S_n\}$ is a collection of n subsets, where $S_i \subseteq \{1, \ldots, m\}$, for each $i = 1, \ldots, n$. The problem is to find a (preferably) small subset $\mathcal{C} = \{B_1, \ldots, B_k\}$ of $\mathcal{S}$ for which

$$B_1 \cup B_2 \cup \cdots \cup B_k = \{1, 2, \ldots, m\}.$$

In this case we say that $\mathcal{C}$ **covers** $\{1, \ldots, m\}$. The following is a greedy heuristic for finding a small set of subsets to cover $\{1, \ldots, m\}$. At each stage, the greedy choice is to choose the set that possesses the most number of **uncovered** elements of $\{1, \ldots, m\}$, where element j is uncovered iff there is currently no subset $B \in \mathcal{C}$ for which $j \in B$. Demonstrate the heuristic on the `Set Cover` instance with $m = 20$ and

$$\mathcal{S} = \{\{5, 14, 18\}.\{1, 2, 16\}, \{2, 5, 8, 11, 13\}, \{3, 8, 13, 15, 17, 19\}, \{2, 4, 5, 15\},$$

$$\{10, 15, 18\}, \{10, 14, 15, 16, 19, 20\}, \{1, 6, 9, 14, 16\}, \{5, 6, 7, 8, 20\}, \{1, 2, 9, 12, 16, 17\}\}.$$

Break any ties by giving precedence to the first set to appear in $\mathcal{S}$ (from left to right) that has the most uncovered elements. (12 pts)

Round 1: {3, 8, 13, 15, 17, 19}

Round 2: {1, 6, 9, 14, 16}

Round 3: {2, 5, 8, 11, 13}

Round 4: {10, 15, 18}

Round 5: {5, 6, 7, 8, 20}

Round 6: {2, 4, 5, 15}

Round 7: {1, 2, 9, 12, 16, 17}