CECS 528, Exam 2, Spring 2024, Dr. Ebert

Directions: Solve AT MOST SIX problems. Closed Notes but you may use a non-programmable scientific calculator

1 Unit 2 LO Problems (25 pts each)

LO4. Answer the following.

(a) Provide a definition for both DFT_n and DFT_n^{-1}. How is each one used to solve the problem of multiplying two polynomials? Explain. (15 pts)

Solution. See FFT Lecture Notes.

(b) Compute $\text{DFT}_4^{-1}(5, 7, -7, 0)$ using the IFFT algorithm. Show the solution to each of the seven subproblem instances and, for each one, clearly represent it using DFT^{-1} notation and apply the formula for computing it. Show all work. (10 pts)

Solution. $\text{DFT}_4^{-1}(5,7,-7,0) = \frac{1}{2}\left((-1,6,-1,6)+(1,-i,-1,i)\odot\left(\frac{7}{2},\frac{7}{2},\frac{7}{2},\frac{7}{2}\right)\right)$

$= \boxed{\left(\frac{5}{4}, 3-\frac{7}{4}i, -\frac{9}{4}, 3+\frac{7}{4}i\right)}$

$\text{DFT}_2^{-1}(5,-7) = \frac{1}{2}\left((5,5)+(1,-1)\odot(-7,-7)\right) = \boxed{(-1,6)}$

$\text{DFT}_2^{-1}(7,0) = \frac{1}{2}\left((7,7)+(1,-1)\odot(0,0)\right) = \boxed{\left(\frac{7}{2},\frac{7}{2}\right)}$

$\text{DFT}_1^{-1}(5) = \boxed{5}$ $\text{DFT}_1^{-1}(-7) = \boxed{-7}$ $\text{DFT}_1^{-1}(7) = \boxed{7}$ $\text{DFT}_1^{-1}(0) = \boxed{0}$

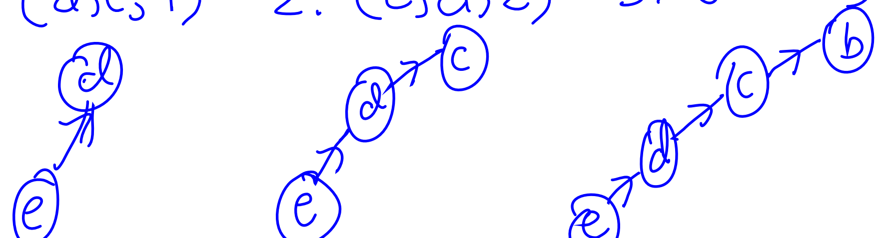
LO5. For the weighted graph with edges

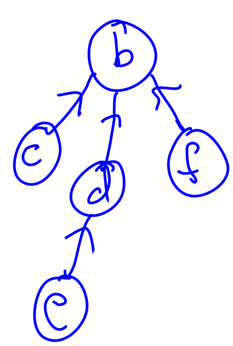
$$(a, e, 6), (b, d, 3), (c, d, 2), (c, f, 5), (d, e, 1), (d, f, 4),$$

Show how the forest of M-Trees changes when processing each edge in Kruskal's sorted list of edges. When unioning two trees, use the convention that the root of the union is the root which has the *lower* alphabetical order. For example, if two trees, one with root a, the other with root b, are to be unioned, then the unioned tree should have root a. (25 pts)

Solution. (Only Trees with 2 or more nodes are drawn)

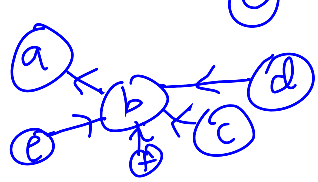
1. (d,e,1) 2. (c,d,2) 3. (b,d,3) 4. (d,f,4) 5. (c,f,5)





No change

6. (a,e,6):



LO6. Recall that the greedy algorithm to the Fuel Reloading problem chooses a sequence $S = s_1, \ldots, s_n$ of stations for which $s_1 < \cdots < s_n < d$, where s_{i+1} is the furthest station from s_i that can be reached from s_i on a full tank of fuel, and d is the final destination, and can be reached from s_n. Let S_{opt} be a minimal set of stations, and let k be the least integer for which $s_k \notin S_{\text{opt}}$. To prove correctness, answer the following questions.

(a) Let $s \in S_{\text{opt}}$ be the station in S_{opt} that comes after s_{k-1}, and is closest to s_{k-1}. Why must such an s exist? Hint: what contradiction arises if such s did not exist? (10 pts)

Solution. Otherwise, the traveler could reach the final destination from s_{k-1} without refueling, which contradicts the algorithm's calculated need for refueling at s_k.

(b) Assuming that different stations have different positions, why must it be the case that $s < s_k$? Hint: what contradiction arises in case $s > s_k$? (10 pts)

Solution. If $s > s_k$, then s is reachable from s_{k-1} and is further away from s_{k-1} than is s_k which implies the algorithm would have selected s over s_k, a contradiction.

(c) Define $\hat{S}_{\text{opt}}$ as $S_{\text{opt}} - s + s_k$. From the algorithm, we know that s_k can be reached from s_{k-1}, and, since $s_k > s$, it is still possible to reach stations in S_{opt} that follow s. Continuing in this manner, we eventually construct an optimal set of stations S_{opt} for which $S \subseteq S_{\text{opt}}$. Why does this imply that $S = S_{\text{opt}}$? Hint: what contradiction arises if S were a proper subset of S_{opt}? (5 pts)

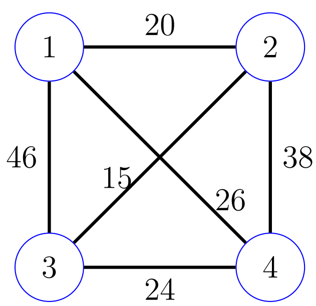
Solution. `Fuel Reloading` is an optimization problem for which the goal is to minimize the number of stations visited. Thus, if S were a proper subset S_{opt}, then S would be a better solution, contradicting the assumption that S_{opt} is optimal.

LO7. Answer/Solve the following questions/problems.

(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). Note: we call it "Runaway TSP" because the salesperson does *not* return to home after visiting each city. (5 pts)

(b) Provide the dynamic-programming recurrence for $\mathrm{mc}(i, A)$. (10 pts)

(c) Apply the recurrence from Part b to the graph below. Show all the necessary computations. Provide the least cost path and give its total cost. (10 pts)



Solution. Start with $\mathrm{mc}(1, \{2, 3, 4\})$ and proceed to compute other mc values as needed.

$$\mathrm{mc}(1, \{2, 3, 4\}) = \min(20 + \mathrm{mc}(2, \{3, 4\}), 46 + \mathrm{mc}(3, \{2, 4\}), 26 + \mathrm{mc}(4, \{2, 3\})).$$

$$\mathrm{mc}(2, \{3, 4\}) = \min(15 + \mathrm{mc}(3, \{4\}), 38 + \mathrm{mc}(4, \{3\})) = \min(15 + 24, 38 + 24) = 39.$$

$$\mathrm{mc}(3, \{2, 4\}) = \min(15 + \mathrm{mc}(2, \{4\}), 22 + \mathrm{mc}(4, \{2\})) = \min(15 + 38, 24 + 38) = 53.$$

$$\mathrm{mc}(4, \{2, 3\}) = \min(38 + \mathrm{mc}(2, \{3\}), 24 + \mathrm{mc}(3, \{2\})) = \min(38 + 15, 24 + 15) = 39.$$

Therefore,

$$\mathrm{mc}(1, \{2, 3, 4\}) = \min(20 + \mathrm{mc}(2, \{3, 4\}), 46 + \mathrm{mc}(3, \{2, 4\}), 26 + \mathrm{mc}(4, \{2, 3\})) =$$

$$\min(20 + 39, 46 + 53, 26 + 39) = 59.$$

This gives the optimal path $P = 1, 2, 3, 4$.

2 Advanced Problems

A1. Compute the 8th roots of unity and verify that their squares yield the 4th roots of unity. Hint: $\cos(\frac{\pi}{4}) = \frac{\sqrt{2}}{2}$. (25 pts)

Solution. 8th roots of unity: $\{1, \frac{\sqrt{2}}{2} + \frac{\sqrt{2}}{2} i, i, -\frac{\sqrt{2}}{2} + \frac{\sqrt{2}}{2} i, -1, -\frac{\sqrt{2}}{2} - \frac{\sqrt{2}}{2} i, -i, \frac{\sqrt{2}}{2} - \frac{\sqrt{2}}{2} i\}$. Squares: $\{1, \frac{1}{2} - \frac{1}{2} + i, -1, \frac{1}{2} - \frac{1}{2} - i, \frac{1}{2} - \frac{1}{2} + i, -1, \frac{1}{2} - \frac{1}{2} - i\} = \{1, i, -1, -i, 1, i, -1, -i\} = \{1, i, -1, -i\} =$ 4th Roots of unity

A2. Consider the three-coin denomination set $S = \{1, 10, 25\}$. Let $m(c)$ denote the minimum number of coins that are needed to return c cents in change using set S.

(a) Provide a dynamic-programming recurrence for computing $m(c)$. Remember to include the base case(s). (20 pts)

Solution.

$$m(c) = \begin{cases} 0 & \text{if } c = 0 \\ m(c-1)+1 & \text{if } 0 < c < 10 \\ \min(m(c-1), m(c-10))+1 & \text{if } 10 < c < 25 \\ \min(m(c-1), m(c-10), m(c-25))+1 & \text{otherwise} \end{cases}$$

(b) Apply the recurence from part a to the problem of determining the minimum number of coins needed for $c = 33$ cents. Include a dynamic-programming array that has solutions to every subproblem of 33 cents or less. (10 pts)

Solution.

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
$m(i)$	0	1	2	3	4	5	6	7	8	9	1	2	3	4	5	6	7	8

A3. Provide a *recursive* implementation of the following function.

```
Boolean uses_item(int p[][], int n, int M, int weight[], int i);
```

which takes as inputs the completed 0-1 knapsack dynamic-programming solution matrix p, the number of items n, the knapsack capacity M, and an item index $1 \leq i \leq n$, and returns 1 iff, item i was placed in the optimal knapsack (in accordance with matrix p). (25 pts)

Solution.

Base Case

```
if(i == n)
    return (p[i][M] != p[i-1][M]);
```

Recursive Case

if(p[n][M]==p[n-1][M])
return uses_item(p, n-1, M, weight, i)

[redacted]

Else
return uses_item(p, n-1, M-W[n], weight, i)

A4. Answer the following.

(a) Let $G = (V, E, c)$ be a positive-weighted graph with integer vertices and $P = i, \ldots, j, \ldots, k$ is a least-cost simple path from i to k, where $i < j < k$. Prove that P_1 and P_2 must both be least-cost paths, where $P_1 = i, \ldots, j$, $P_2 = j, \ldots, k$, and $P = P_1 \circ P_2$ is the concatenation of P_1 with P_2. Hint: use a proof by contradiction. (10 pts)

Solution. If P_1' were a less-costly path from i to j, then we may assume that, except for j, it does not visit any vertices that are visited by P_2 (why?). Thus $P\prime = P_1' \circ P_2$ is a path from i to k that has lesser cost, a contradiction.

(b) Give an example of a *maximum-cost* simple path $P = i, \ldots, j, \ldots, k$ from i to k, but for which neither P_1 nor P_2 are maximum-cost paths, where $P_1 = i, \ldots, j$, $P_2 = j, \ldots, k$, and $P = P_1 \circ P_2$ is the concatenation of P_1 with P_2. Hint: use a directed graph. (15 pts)

Solution. See the solution to Problem 20 of the Dynamic Programming Lecture. Modify it so that P_2 is also not a maximum-cost paths. Hint: add an edge from c to a.

3 Unit 1 LO Problems (0 pts each)

LO1. Solve the following.

a. Compute $2^{80} + 3^{70} \bmod 5$. Show all work.
Solution. $2^4 \equiv 1 \bmod 5$ and $3^2 \equiv -1 \bmod 5$ implies

$$2^{80} \equiv (2^4)^{20} \equiv 1^{20} \equiv 1 \bmod 5$$

and

$$3^{70} \equiv (3^2)^{35} \equiv (-1)^{35} \equiv -1 \bmod 5.$$

Therefore, $2^{80} + 3^{70} \equiv 1 + (-1) \equiv 0 \bmod 5$.

b. In the Strassen-Solovay test, is $a = 4$ a witness or accomplice for $n = 21$? Show work in computing both the left and right sides of the mod-21 congruence.
Solution. We have $4^{\frac{21-1}{2}} = 2^{20}$. Since $2^6 \equiv 64 \equiv 1 \bmod 21$, we have

$$2^{20} \equiv 2^2 \equiv 4 \bmod 21.$$

Also,

$$\left(\frac{4}{21}\right) = \left(\frac{2}{7}\right)^2 \left(\frac{2}{3}\right)^2 = 1.$$

Hence, $4 \not\equiv 1 \bmod 21$, and so 4 is a witness to $n = 21$ being composite. □

LO2. Solve each of the following problems.

a. Use the Master Theorem to determine the growth of $T(n)$ if it satisfies the recurrence $T(n) = 4T(n/2) + n^{\log_2 5} \log^3 n$.
Solution. Since $f(n) = n^{\log_2 5} \log^3 n = \Omega(n^{2+\epsilon})$ for $\epsilon = \log_2 5 - 2$, it follows by Case 3 of the Master Theorem that $T(n) = \Theta(n^{\log_2 5} \log^3 n)$. □

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

$$T(n) = 4T(n/2) + 3n,$$

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

Inductive Assumption: $T(k) \geq Ck^2 \log k$ for all $k < n$, some const. $C > 0$

Show $T(n) \geq Cn^2 \log n$.

$$T(n) = 4T(n/2) + 3n \geq 4C\left(\frac{n}{2}\right)^2 \log\left(\frac{n}{2}\right) + 3n =$$

$$Cn^2(\log n - 1) + 3n = Cn^2 \log n - Cn^2 + 3n \geq Cn^2 \log n \Leftrightarrow Cn^2 \leq 3n \Leftrightarrow C \leq \frac{3}{n}$$

which is impossible if we assume $C > 0$.

$\therefore$ The statement cannot be proven.

LO3. Solve the following problems.

a. Recall the `Randomized Find-Statistic` algorithm. For an in input array a of size 128, and $k = 18$, suppose a pivot is randomly selected from the indices 0-127. What is the probability that, after using this pivot for the partitioning step, the next array to consider will have a size that is no greater than 96? Explain and show work. How many random pivots would we expect would have to be generated before finding one that that reduces the array to the desired size (of 96 or fewer elements). Explain.

Solution. If the pivot is chosen from 0-17, then the $k = 18$ least member will be located in a_{right} for which $|a_{\text{right}}| \geq 111$. Also, if the pivot is chosen as 96 or greater, then $k = 18$ least member will be located in a_{left} for which $|a_{\text{left}}| \geq 97$. Therefore, the only acceptable pivots range from 17 to 95, and so the probability that the next array will have a size of 96 or less is equal to 79/128, and so the expected number of pivot selections before the size becomes 96 or less is no more than $128/79 = 1.62$. □

b. Recall that the `Minimum Positive Subsequence Sum (MPSS)` problem admits a divide-and-conquer algorithm that, on input integer array a, requires computing the mpss of any subarray of a that contains both $a[n/2-1]$ and $a[n/2]$ (the end of a_{left} and the beginning of a_{right}). For

$$a = 46, -37, 23, -47, 11, -36, 46, -40, 14, -29$$

provide the two sorted arrays a = LeftSums and b = RightSums from which the minimum positive sum $a[i] + b[j]$ represents the desired mpss (for the middle), where i in the index range of a and j is within the index range of b. Also, demonstrate how the minimum positive sum $a[i] + b[j]$ may be computed via the movement of left and right markers.

Solution. Unsorted Left Sums: 11, −36, −13, −50, −4

Unsorted Right Sums: −36, 10, −30, −16, −45

a = −50, −36, −13, −4, 11

b = −45, −36, −30, −16, 10

mpss middle = 6

i	j	a[i]+b[j]	mpss
0	4	−50+10=−40	n/a
1	4	−36+10=−26	n/a
2	4	−13+10=−3	n/a
3	4	−4+10=6	6
3	3	−4+−16=−20	6
4	3	11+−16=−5	6