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)

(b) Compute $\mathrm{DFT}_4^{-1}(5, 7, -7, 0)$ 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)

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)

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)

(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)

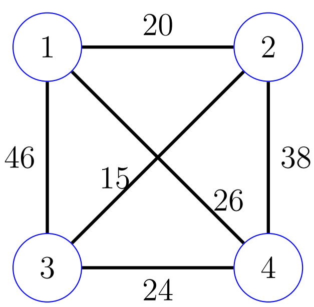
(c) Define $\hat{S}_{\text{opt}}$ as $S_{\text{opt}} - s + s_k$. From the algorithm, we know that s_k can be reached from s_{k-1}, and, since $s_k > s$, it is still possible to reach stations in S_{opt} that follow s. Continuing in this manner, we eventually construct an optimal set of stations S_{opt} 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)

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)



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)

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)

(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)

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

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

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)

(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)

3 Unit 1 LO Problems (0 pts each)

LO1. Solve the following.

a. Compute $2^{80} + 3^{70} \bmod 5$. Show all work.

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.

LO2. Solve each of the following problems.

a. Use the Master Theorem to determine the growth of $T(n)$ if it satisfies the recurrence $T(n) = 4T(n/2) + n^{\log_2 5} \log^3 n$.

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)

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.

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 $a = \text{LeftSums}$ and $b = \text{RightSums}$ 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.