CSCI3160 Design and Analysis of Algorithms (2025 Fall)

Midterm Exam (A)

Name: **123456789** Student ID: **Xiao LIANG**

Please read the following instructions carefully.

- Please fill in your *Name* and *Student ID* in the corresponding fields above.
- This question booklet contains 10 questions, worth a total of 100 points. It consists of 12 pages. Please check to make sure no pages are missing.
- The last two pages are scratch paper. You may remove them from the booklet if you wish, but be sure to submit them along with your exam. If you need additional scratch paper, raise your hand to request it.
- This question booklet is printed single-sided. If necessary, feel free to use the blank space on the back of each page (except for the scratch paper) for your answers.

This table is intended for grading use only. Exam participants are requested to leave it blank.

Question:	1	2	3	4	5	6	7	8	9	10	Total
Points:	12	14	6	6	6	12	12	10	10	12	100
Score:											

Exam Questions:

1. (12 points) Problem statement hidden.

Solution: We prove it by contradiction. Suppose the length of codeword i (denote by $len(i)$) is larger than that of $i+1$ (denote by $len(i+1)$), then since the frequency of i (denote by $f(i)$) is strictly larger than that of $i+1$ (denote by $f(i+1)$), by switching the codeword of i and $i+1$, we can get

$$f(i+1) \cdot len(i) + f(i) \cdot len(i+1) + \delta < f(i+1) \cdot len(i+1) + f(i) \cdot len(i) + \delta,$$

where δ denotes the average length of other codewords.

This gives us a prefix code with smaller average length, contradicting the fact that the original prefix code is optimal. Thus, the length of codeword i cannot be longer that that of $i+1$.

2. (14 points) Answers to True or False problems will not be provided.

3. (6 points) Problem statement hidden.

Solution:

$\text{opt}(1) = 2$, $\text{opt}(2) = 5$, $\text{opt}(3) = 9$, $\text{opt}(4) = 11$, $\text{opt}(5) = 14$, $\text{opt}(6) = 18$, and $\text{opt}(7) = 20$.

4. (6 points) Problem statement hidden.

Solution:

$cf, ef, bc, ab, ad.$

5. (6 points) Problem statement hidden.

Solution:

Part 1: Since the product polynomial is of degree 3, we need to set $n = 4$. I.e., we will need to use the 4-th roots of unity $\{1, -1, i, -i\}$.

Part 2: The point-value representation for $f(x)$ is:

$$(1, 2), (-1, 0), (i, 1 + i), (-i, 1 - i).$$

The point-value representation for $g(x)$ is:

$$(1, 2), (-1, 2), (i, 0), (-i, 0).$$

Part 3: The point-value representation for $f \cdot g$ is:

$$(1, 4), (-1, 0), (i, 0), (-i, 0).$$

6. (12 points) Problem statement hidden.

Solution: Let T be a minimum spanning tree of G that *does not* contain the edge $e = \{u, v\}$, where $u \in S$ and $v \notin S$, and e is the lightest edge crossing the cut $(S, V \setminus S)$.

Since T is a spanning tree, it connects all vertices, so there is a unique path in T between u and v. Let P be the unique path in T connecting u to v.

Since $u \in S$ and $v \notin S$, the path P must contain at least one other edge $e' = \{x, y\}$ such that $x \in S$ and $y \in V \setminus S$ (i.e., e' also crosses the cut). This is because otherwise, the MST will not be connected.

Also note that by definition, e is the minimum-weight edge across the cut. So,

$$w(e) < w(e') \quad \text{for any other edge } e' \text{ crossing the cut.}$$

Now, construct a new tree T' by:

- Adding edge e to T, which creates a cycle (since T is a tree).
- Removing edge $e' \in P$.

Since e and e' both cross the cut, and $w(e) < w(e')$, we have:

$$w(T') = w(T) - w(e') + w(e) < w(T),$$

which contradicts the assumption that T is a minimum spanning tree.

7. (12 points) Problem statement hidden.

Solution: It suffices to find a counter example, which we present below.

Consider the setting where we have 3 bold bricks and $W = 5$. The weights and prices of the gold bricks are as follows

- $p_1 = 4$ and $d_1 = 40$;
- $p_2 = 3$ and $d_2 = 30$;
- $p_3 = 2$ and $d_3 = 20$;

For this input, the algorithm will select only the first brick, worth 40 dollars. However, the optimal solution is to pick the last two bricks, worth 50 dollars.

8. (10 points) Problem statement hidden.

Solution: Prof. Goofy's algorithm cannot achieve the k-selection in $O(n)$ deterministically.

Consider a worst-case example where $A = (n, n-1, n-1, \ldots, 1)$. For this input A, his algorithm will recurse infinitely; it will never stop.

9. (10 points) Problem statement hidden.

Solution: Let A be the input array for the inversion counting problem. Construct a set P of n points as follows: for each $i \in [1, n]$, add to P the point $p_i = (i, -A[i])$. Observe that (i, j) is an inversion if and only if point p_j dominates p_i. We run a dominance counting algorithm to find, for each point p_j ($j \in [1, n]$), the number c_j of points dominated by p_j. Then, the number of inversions in A can be obtained as $\sum_{j=1}^{n} c_j$. As P can be constructed in $O(n)$ time, the whole algorithm uses $f(n) + O(n)$ time to solve the counting inversion problem.

10. (12 points) Problem statement hidden.

Solution:

Subproblem Definition

Let the given sorted keys be $k_1 < k_2 < \cdots < k_n$, with associated search probabilities $p_1, p_2, \ldots, p_n$, where $\sum_{i=1}^{n} p_i = 1$.

We define a subproblem as follows:

- Let $e[i][j]$ denote the minimum expected search cost of a binary search tree that contains the keys $k_i, k_{i+1}, \ldots, k_j$ for $1 \leq i \leq j \leq n$. If $i > j$, we define $e[i][j] = 0$, representing an empty subtree.
- Let $w[i][j] = \sum_{l=i}^{j} p_l$. This is the total probability of the keys in the range $[i, j]$.

Recurrence Relation

To compute $e[i][j]$, we try all possible choices of root r between i and j. If we choose k_r as the root:

- Keys k_i to k_{r-1} will be in the *left subtree*.
- Keys k_{r+1} to k_j will be in the *right subtree*.
- When we connect the left and right subtrees to the root note k_r, all keys in the subtree will increase their depth by 1; moreover, we also need to include the cost p_r once for k_r. The total cost added in this step happens to be $w[i][j]$.

Thus, the recurrence is:

$$e[i][j] = \begin{cases} 0 & \text{if } i > j \\ \min\limits_{r=i}^{j} \left(e[i][r-1] + e[r+1][j] + w[i][j]\right) & \text{if } i \leq j \end{cases}$$

Algorithm Description

The $w[i][j]$ values can be pre-computed. We can then fill out the dynamic programming table $e[1 \ldots n][1 \ldots n]$ following a bottom-up, left-to-right order. After filling the table, the minimum expected search cost for the entire optimal BST is stored in $e[1][n]$.

Time Complexity

- There are $O(n^2)$ subproblems.
- For each subproblem $e[i][j]$, we try $O(n)$ possible roots.
- Total time complexity: $O(n^3)$

Scratch Paper I

Scratch Paper II