Problem Set 6

This problem set is divided into two parts: Part A problems are theory questions, and Part B problems are programming tasks.

Part A questions are due **Friday, May 7th** at **11:59PM**.

Part B questions are due **Friday, May 7th** at **11:59PM**.

Solutions should be turned in through the course website in PDF form using LaTeX or scanned handwritten solutions.

A template for writing up solutions in LaTeX is available on the course website.

Remember, your goal is to communicate. Full credit will be given only to the correct solution which is described clearly. Convoluted and obtuse descriptions might receive low marks, even when they are correct. Also, aim for concise solutions, as it will save you time spent on write-ups, and also help you conceptualize the key idea of the problem.

Part A: Due Friday, May 7th

1. **(25 points)** Placing Parentheses

   You are given an arithmetic expression containing n real numbers and $n-1$ operators, each either $+$ or $\times$. Your goal is to perform the operations in an order that maximizes the value of the expression. That is, insert $n-1$ pairs of parentheses into the expression so that its value is maximized.

   For example:

   - For the expression $6 \times 3 + 2 \times 5$, the optimal ordering is to add the middle numbers first, then perform the multiplications: $((6 \times (3+2)) \times 5) = 150$.
   - For the expression $0.1 \times 0.1 + 0.1$, the optimal ordering is to perform the multiplication first, then the addition: $((0.1 \times 0.1) + 0.1) = 0.11$.
   - For the expression $(-3) \times 3 + 3$, the optimal ordering is $((-3) \times 3) + 3) = -6$.

   (a) **(10 points)** Clearly state the set of subproblems that you will use to solve this problem.

   Solution: First, we define some notation. Denote the numbers by $a_1, a_2, \ldots, a_n$, and the operators by $op_1, op_2, \ldots, op_{n-1}$, so the given expression is $a_1\ op_1\ \ldots\ op_{n-1}\ a_n$.

   Let $M[i,j]$ be the *maximum* value obtainable from the subexpression beginning at a_i and ending at a_j (i.e., $a_i\ op_i\ \ldots\ op_{j-1}\ a_j$), and let $m[i,j]$ be the *minimum* value obtainable from the subexpression beginning at a_i and ending at a_j.

   We must keep track of both the minimum and the maximum because the maximal value of an expression may result from multiplying two negative subexpressions.

(b) **(10 points)** Write a recurrence relating the solution of a general subproblem to solutions of smaller subproblems.

Solution: To solve the subexpression $a_a \dots a_b$, we can split it into two problems at the kth operator, and recursively solve the subexpressions $a_a \dots a_k$ and $a_{k+1} \dots a_b$. In doing so, we must consider all combinations of the minimizing and maximizing subproblems.

The base cases are $M[i,i] = m[i,i] = a_i$, for all i.

$$\begin{aligned} M[a,b] = \max_{a \le k < b} (\max(&M[a,k] \; op_k \; M[k+1,b], \\ &M[a,k] \; op_k \; m[k+1,b], \\ &m[a,k] \; op_k \; M[k+1,b], \\ &m[a,k] \; op_k \; m[k+1,b])) \end{aligned}$$

$$\begin{aligned} m[a,b] = \min_{a \le k < b} (\min(&M[a,k] \; op_k \; M[k+1,b], \\ &M[a,k] \; op_k \; m[k+1,b], \\ &m[a,k] \; op_k \; M[k+1,b], \\ &m[a,k] \; op_k \; m[k+1,b])) \end{aligned}$$

(c) **(5 points)** Analyze the running time of your algorithm, including the number of subproblems and the time spent per subproblem.

Solution: There are $O(n^2)$ subproblems, two for each pair of indices $1 \le a \le b \le n$. The subproblem $M[a,b]$ must consider $O(b-a) = O(n)$ smaller subproblems. Thus the total running time is $O(n^3)$.

2. **(25 points)** Pots of Gold

There are N pots of gold arranged linearly. Alice and Bob are playing the following game. They take alternate turns, and in each turn they remove (and *win*) either of the two pots at the two ends of the line. Alice plays first. Given the amount of gold in each pot, design an algorithm to find the maximum amount of gold that Alice can assure herself of winning.

(a) **(10 points)** Clearly state the set of subproblems that you will use to solve this problem.

Solution: Let $g_1, g_2, \dots, g_N$ be the amount of gold in the N pots. For any $1 \le i \le j \le N$, let $G[i,j]$ be the maximum amount of gold that the player who plays first can assure herself of winning if the problem instance only had pots $i, i+1, \dots, j$.

(b) **(10 points)** Write a recurrence relating the solution of a general subproblem to solutions of smaller subproblems.

Solution: If $j > i$,

$$G[i,j] = \max(\sum_{k=i}^{j} g_k - G[i+1,j], \sum_{k=i}^{j} g_k - G[i,j-1]),$$

else if $j = i$, $G[i,j] = g_i$.

(c) **(5 points)** Analyze the running time of your algorithm, including the number of subproblems and the time spent per subproblem. Hint: your overall algorithm should be $O(N^2)$.

Solution: Precompute $\sum_{k=i}^{j} g_k$ for every $1 \leq i \leq j \leq N$ in $O(N^2)$ time. Then, each subproblem takes $O(1)$ time; since there are $O(N^2)$ subproblems, the total time complexity is $O(N^2)$.