Practice Problems 6

Problem 1. Matrix Chain Multiplication. You are given five matrices

$$A_1 : 30 \times 5, \qquad A_2 : 5 \times 60, \qquad A_3 : 60 \times 2, \qquad A_4 : 2 \times 20, \qquad A_5 : 20 \times 10.$$

Recall that multiplying an $a \times b$ matrix by a $b \times c$ matrix requires abc scalar multiplications. Let $\mathrm{OPT}(i, j)$ denote the minimum number of scalar multiplications needed to compute

$$A_i A_{i+1} \cdots A_j.$$

Also let $\mathrm{split}(i, j)$ denote a value of k at which the optimal recurrence for $\mathrm{OPT}(i, j)$ attains its minimum.

(a) Fill in the following two DP tables for $\mathrm{OPT}(i, j)$ and $\mathrm{split}(i, j)$. Entries below the diagonal do not need to be filled. Think about how the optimal parenthesization can be obtained by splitting the matrix chain at some position k.

$\mathrm{OPT}(i,j)$	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$
$i=1$	0				
$i=2$	\	0			
$i=3$	\	\	0		
$i=4$	\	\	\	0	
$i=5$	\	\	\	\	0

$\mathrm{split}(i,j)$	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$
$i=1$	–				
$i=2$	\	–			
$i=3$	\	\	–		
$i=4$	\	\	\	–	
$i=5$	\	\	\	\	–

Solution: The completed tables are:

$\mathrm{OPT}(i,j)$	1	2	3	4	5
1	0	9000	900	2100	1900
2	\	0	600	800	1100
3	\	\	0	2400	1600
4	\	\	\	0	400
5	\	\	\	\	0

$\mathrm{split}(i,j)$	1	2	3	4	5
1	–	1	1	3	3
2	\	–	2	3	3
3	\	\	–	3	3
4	\	\	\	–	4
5	\	\	\	\	–

(b) Using your split(i, j) values, give an optimal parenthesization of

$$A_1 A_2 A_3 A_4 A_5$$

and state the minimum number of scalar multiplications.

Solution: The optimal parenthesization is

$$\big(A_1(A_2A_3)\big)(A_4A_5).$$

Its cost is

$$5 \cdot 60 \cdot 2 + 30 \cdot 5 \cdot 2 + 2 \cdot 20 \cdot 10 + 30 \cdot 2 \cdot 10 = 1900.$$

(c) Let $p_0, p_1, \ldots, p_5$ be the matrix dimensions, so that A_i has dimensions $p_{i-1} \times p_i$. Give the base case and recurrence for $\mathrm{OPT}(i, j)$. Explain in what order the subproblems should be computed in a bottom-up algorithm, and state its time and space complexities for a chain of n matrices.

Solution: The base case is $\mathrm{OPT}(i, i) = 0$. For $i < j$,

$$\mathrm{OPT}(i, j) = \min_{i \le k < j} \{\mathrm{OPT}(i, k) + \mathrm{OPT}(k + 1, j) + p_{i-1} p_k p_j\}.$$

The final multiplication splits the chain after A_k. In a bottom-up implementation, process intervals by increasing length $j - i + 1$. There are $O(n^2)$ intervals and $O(n)$ candidate splits per interval, giving $O(n^3)$ time and $O(n^2)$ space.

(d) Let $P(n)$ be the number of different valid ways to fully parenthesize a product of n matrices while keeping their order fixed. Give a recurrence for $P(n)$ with an appropriate base case, and use it to compute $P(5)$.

Solution: Every full parenthesization has a last multiplication separating the first k matrices from the remaining $n - k$. Thus,

$$P(1) = 1, \qquad P(n) = \sum_{k=1}^{n-1} P(k) P(n - k) \quad (n \ge 2).$$

The values are $P(2) = 1$, $P(3) = 2$, $P(4) = 5$, so

$$P(5) = P(1)P(4) + P(2)P(3) + P(3)P(2) + P(4)P(1) = 14.$$

Problem 2. Randomized Algorithms and Amplification. A randomized algorithm may use random choices during its execution. Its success probability is taken over these random choices, for each fixed input.

(a) Consider the following two algorithms:

- Algorithm A always produces a correctly sorted array, but its running time depends on randomly selected pivots.

- Algorithm B always terminates within a fixed time bound, but may return an incorrect answer depending on its random choices.

Which algorithm is *Las Vegas* and which is *Monte Carlo*? Briefly explain.

Solution: Algorithm A is *Las Vegas*: the answer is always correct, but running time depends on randomness. Algorithm B is *Monte Carlo*: it has bounded running time, but may produce a wrong answer.

(b) Consider the problem of deciding whether an n-bit string contains at least one 1. An algorithm always outputs `YES`, without reading its input. If inputs are chosen uniformly from $\{0,1\}^n$, what fraction of inputs does it answer correctly? Does this imply that it has success probability at least $2/3$ for *every* input? Explain.

Solution: All inputs except the all-zero string contain a 1, so the fraction of correct answers is

$$\frac{2^n - 1}{2^n} = 1 - 2^{-n}.$$

No. On the fixed input 0^n, the algorithm is always wrong, so its success probability is 0. High average accuracy over inputs does not imply a success guarantee for every input.

(c) Suppose a randomized algorithm answers a `YES/NO` question correctly with probability at least $2/3$ on every fixed input. Run it three times using independent randomness and output the majority answer. What is the probability that the majority answer is correct? Give a lower bound and show your calculation.

Solution: For each independent run, the success probability is at least $2/3$. The majority is correct when at least two runs are correct; hence its success probability is at least

$$\binom{3}{2}\left(\frac{2}{3}\right)^2\left(\frac{1}{3}\right) + \left(\frac{2}{3}\right)^3 = \frac{20}{27}.$$

Equality holds when each run succeeds with probability exactly $2/3$.

(d) Generalize part (c) to an odd number t of independent runs. Show that the majority-vote error probability is at most $e^{-t/18}$. You may use the following inequality for independent Bernoulli random variables $X_1, \ldots, X_t$:

$$\Pr\left[\sum_{i=1}^{t} X_i \le \mathbb{E}\left[\sum_{i=1}^{t} X_i\right] - a\right] \le e^{-2a^2/t} \qquad (a > 0).$$

How large should t be to make the error probability at most δ, where $0 < \delta < 1$?

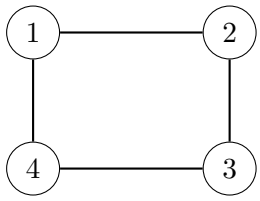
Solution: Let X_i indicate correctness of run i, and put $X = \sum_i X_i$. Since $\mathbb{E}[X] \ge 2t/3$, an incorrect majority requires $X < t/2$, hence a deviation of at least $t/6$ below its mean. The supplied inequality gives

$$\Pr[\text{majority wrong}] \le \exp\left(-\frac{2(t/6)^2}{t}\right) = e^{-t/18}.$$

Choosing any odd integer $t \ge 18\ln(1/\delta)$ makes the error probability at most δ, so $t = O(\log(1/\delta))$ suffices.

Problem 3. Karger's Randomized Minimum Cut. Recall that a cut of an undirected graph partitions its vertices into two nonempty sets. The *size* of a cut is the number of edges crossing between the two sets. Karger's algorithm repeatedly selects a uniformly random edge and contracts it until only two super-vertices remain. Self-loops are removed, but parallel edges are retained.

(a) Consider the following undirected cycle graph. Give a minimum cut by specifying the two vertex sets, and state the size of the minimum cut.



Solution: One minimum cut is $\{1\}$ versus $\{2, 3, 4\}$; the crossing edges are $(1, 2)$ and $(1, 4)$. Every vertex has degree 2, and this cycle is connected, so no cut has fewer than two crossing edges. The minimum cut size is 2.

(b) In the graph from part (a), suppose Karger's algorithm first contracts edge $(1, 2)$, and then contracts the edge between the super-vertex $\{1, 2\}$ and vertex 3. Describe the final multigraph. How many parallel edges remain between its two super-vertices, and what cut does this execution output?

Solution: After the two contractions, the remaining super-vertices are $\{1, 2, 3\}$ and $\{4\}$. The original edges $(3, 4)$ and $(4, 1)$ become two parallel edges between them; the other edges become self-loops and are deleted. The resulting cut is $(\{1, 2, 3\}, \{4\})$, with size 2.

(c) Now consider an arbitrary connected, undirected graph with $n \geq 2$ vertices and minimum cut size k. Fix one particular minimum cut. Show that a single run of Karger's algorithm finds this cut with probability at least

$$\frac{2}{n(n-1)}.$$

Hint: Condition on never contracting an edge of the fixed cut. When r super-vertices remain, what can you say about their degrees and the probability that the next contraction crosses this cut?

Solution: Fix a minimum cut of size k, and condition on never contracting any edge crossing it. With r super-vertices remaining, every super-vertex has degree at least k; otherwise its edges would give a cut smaller than k in the original graph. Therefore at least $rk/2$ edges remain, and

$$\Pr[\text{next edge crosses the fixed cut} \mid \text{survival}] \leq \frac{k}{rk/2} = \frac{2}{r}.$$

The probability of preserving this cut through all $n - 2$ contractions is at least

$$\prod_{r=3}^{n} \left(1 - \frac{2}{r}\right) = \prod_{r=3}^{n} \frac{r-2}{r} = \frac{2}{n(n-1)}.$$

(d) Suppose we run Karger's algorithm independently t times and return the smallest cut found. Give an upper bound on the probability that none of the runs finds a minimum cut. How many runs suffice to find a minimum cut with probability at least $1 - 1/n$? You may use $1 - x \le e^{-x}$.

Solution: Write $p = 2/[n(n-1)]$. Each independent run finds a minimum cut with probability at least p, so

$$\Pr[\text{all } t \text{ runs fail}] \le (1-p)^t \le e^{-pt}.$$

To make this at most $1/n$, it suffices to choose

$$t \ge \frac{n(n-1)}{2} \ln n,$$

rounding up to an integer. Thus $O(n^2 \log n)$ repetitions suffice.

Problem 4. Polynomial Identity Testing. You have black-box access to two polynomials $f(x)$ and $g(x)$ with real coefficients, each of degree at most d. You may query the value of either polynomial at any chosen real number. The goal is to determine whether $f(x)$ and $g(x)$ are identical polynomials. Let $h(x) = f(x) - g(x)$.

For this problem, you may use the fact that a nonzero polynomial of degree at most d has at most d distinct real roots. Let $S = \{1, 2, \ldots, N\}$, where $N > d \ge 1$.

(a) Consider $f(x) = x^2 - 6x + 8$ and $g(x) = 0$. Evaluate $h(2)$, $h(3)$, and $h(4)$. Why does finding a point where $f(x) = g(x)$ not necessarily mean that the two polynomials are identical?

Solution: Here $h(x) = x^2 - 6x + 8 = (x-2)(x-4)$, so

$$h(2) = 0, \qquad h(3) = -1, \qquad h(4) = 0.$$

Two different polynomials may agree at some points.

(b) Consider the following randomized test: choose r uniformly at random from S, query $f(r)$ and $g(r)$, and output `DIFFERENT` if their values differ; otherwise output `EQUAL`. Show that this algorithm is always correct when $f = g$. When $f \ne g$, what is an upper bound on the probability that it incorrectly outputs `EQUAL`?

Solution: If $f = g$, then $h(r) = 0$ for every r, so the answer is always correct. If $f \ne g$, then h is a nonzero polynomial of degree at most d, with at most d roots in S. Therefore,

$$\Pr[\text{incorrectly output EQUAL}] \le \frac{d}{N}.$$

The test has one-sided error: it never incorrectly declares equal polynomials different.

(c) Repeat the test from part (b) independently t times, outputting `DIFFERENT` if any test finds unequal values. Bound the probability of incorrectly outputting `EQUAL` when $f \ne g$. If $N \ge 2d$, how many independent tests suffice to make this error probability at most $1/16$?

Solution: For unequal polynomials, all t independent tests must hit roots for the algorithm to err. Hence

$$\Pr[\text{incorrectly output EQUAL}] \leq (d/N)^t.$$

If $N \geq 2d$, this is at most 2^{-t}. Taking $t = 4$ gives error probability at most $1/16$.

(d) Suppose instead that the t query points are selected uniformly *without replacement* from S. If the nonzero polynomial h has exactly r distinct roots in S, give the probability that all t queries return zero. Compare this probability with independent sampling *with replacement*, and explain what happens when $t > d$.

Solution: For $0 \leq t \leq N$, sampling without replacement gives

$$\Pr[\text{all zeros}] = \frac{\binom{r}{t}}{\binom{N}{t}} \quad (\text{with } \binom{r}{t} = 0 \text{ if } t > r).$$

For sampling with replacement, the probability is $(r/N)^t$. Without replacement,

$$\frac{\binom{r}{t}}{\binom{N}{t}} = \prod_{i=0}^{t-1} \frac{r-i}{N-i} \leq \left(\frac{r}{N}\right)^t.$$

Since $r \leq d$, if $t > d$ (and $t \leq N$), sampling without replacement detects that $f \neq g$ with certainty.