CS6170: Randomized Algorithms
Problem Set #1

Name: Your name — Marks: 25
Roll No: Your roll number — Due: August 27, 23:59

Problem 1 — 3 marks

Suppose that you want to generate a random permutation of the sequence of numbers 1 to n. You have at your disposal, a source of unbiased random bits. Give an efficient algorithm to generate a random permutation using as few random bits as possible from the source.

Solution:

Number of possible permutations = $n!$. We can number every single possible permutation and match it with a certain number. For example, we can use increasing order as the base queue and count the remaining permutations in a similar manner to alphabetic order.

This gives a requirement of $ciel(log(n!))$ as the number of random bit required to generate a number representing a permutation.

Problem 2 — 4 marks

While discussing Frievald's algorithm for verifying matrix multiplication, the following algorithms were proposed in class.

1. **Algorithm 1**: Given A, B, C, choose a number j uniformly at random from $\{1, 2, \ldots, n\}$ and multiply A with the j^{th} column of B and check if it matches the j^{th} column of C entrywise.
2. **Algorithm 2**: Given A, B, C, choose two numbers i and j uniformly at random from $\{1, 2, \ldots, n\}$ and multiply the i^{th} row of A and the j^{th} column of B and check if the product is equal to the $(i, j)^{th}$ entry of C.

Analyze the two algorithms and explain which one is better. Are these algorithms better than Frievald's algorithm? What is the running time of this algorithm if I want to make the error probability ϵ?

Solution:

Algorithm 1:

Let the j^{th} column of B be B_j and similarly for C_j

The two cases:

Case 1: $AB_j \neq C_j$: Then $Pr(error) = 0$

Case 2: $AB_j = C_j$: Then $Pr(Error) = Pr(AB \neq C)$

$D = AB - C \neq 0$, thus D must have at least 1 non zero element. $Pr(D[i][j] \neq 0) \geq 1/n^2$

The probability of choosing a non-zero element in D_j is $\geq n * \frac{1}{n^2} = 1/n$

$\therefore Pr(error) \leq 1 - 1/n$

Run-time of Algo 1: $O(n^2)$

Algorithm 2:

Let A_i be the i^{th} row of A and B_j be the j^{th} column of B.

Similarly this algo has a non-zero $Pr(error)$ if $A_i B_j = C_{i,j}$

$\therefore Pr(Error) \leq 1 - 1/n^2$

Run-time of Algo 1: $O(n)$

Comparison:
Let us compare to obtain an error of ϵ

Algo 1(Repeat k times):

$Pr(Error)^k \leq \epsilon$

$(1 - 1/n)^k \leq e^{k/n} \leq \epsilon$

$k = nln(\epsilon)$

Run-Time = $k * O(n^2) = O(n^3 \epsilon)$

Algo 2(Repeat k times):

$Pr(Error)^k \leq \epsilon$

$(1 - 1/n^2)^k \leq e^{k/n^2} \leq \epsilon$

$k = n^2 ln(\epsilon)$

Run-Time = $k * O(n) = O(n^3 \epsilon)$

Thus the effective performance is equivalent in both algorithms.

Both Algorithms are worse than Freivald's Algorithm with a much higher $Pr(error)$

Difference: Algo 1 uses 1 random variable(i) to pick a value from 1 to n whereas Algo 2 uses 2 random variables(i, j) from 1 to n. Algo 2 uses twice the amount of random bits to run.

Collaborator: A Balakrishnan (CS20B012)

Problem 3 — 5 marks

An s-t-cut in a graph is a set of edges such that their removal gives a new graph which does not contain a path from s to t. Consider the following modification of Karger's algorithm to compute the smallest s-t-cut in the graph: Choose a random edge in the graph such that it not between supernodes containing s and t, and contract it; keep continuing until the only two supernodes are the ones containing s and t, and output this as the minimum s-t-cut.

Show that there are graphs such that the success probability of this algorithm finding the minimum s-t-cut is exponentially small.

Solution:

The graph used to depict the given problem consists of 2 segments, one having s and the other having t. We have a min-cut of n edges between them and both the segments have $2n^2 + 2n$ edges and $3n + 1$ vertices each.

The configuration of the graph: We can look at the graph as 4 parts.

Part 1: A Star-like graph where s is the center connected to $2n$ vertices. $2n$ edges in Part 1.

Part 2: n vertices. Dense connection between the $2n$ vertices in Part 1 to n vertices. $2n * n$ edges between Part 2 and Part 1.

Part 3: n vertices. One on One connection between Part 2 and Part 3. n edges between Part 2 and 3.

Part 4: Similar to Part 1 along with a dense connection to Part 3, $2n$ vertices plus t at the center. $2n * n$ edges between 3, 4 and $2n$ in Part 4.

As we can see, the min-cut is n edges from Part 2 to Part 3. Let $m = 4n^2 + 5n$ be the total number of edges and $v = 6n + 2$ be the total number of vertices.

The probability that the min-cut edges are not chosen in each iteration can be calculated:

$P = \frac{m-n}{m} * \frac{m-n-1}{m-1} * \frac{m-n-2-\delta}{m-2-\delta} .....(v - 2terms)$

Delta is a possible variable because one contraction of vertices can remove more than one edge making P only have $v - 2$ terms representing $v - 2$ iterations.

$\frac{m-n-i-\delta}{m-i-\delta} < \frac{m-n-i}{m-i}$ for all $v - 2$ terms.

$P < \frac{m-n}{m} * \frac{m-n-1}{m-1} * \frac{m-n-2}{m-2} ..... \frac{m-n-v+1}{m-v+1}$

$P < (1-n/m)*(1-n/m - 1)*(1-n/m - 2).....(1-n/m - v + 1) < e^{-n/m}*e^{-n/m-1}....e^{-n/m-v+1}$

$P < e^{-n*1/m+1/m-1+1/m-2....1/m-v+1} < e^{-n*H(m)}$

$\therefore P < e^{-n*H(4n^2+5n)}$

Collaborator: Abdullah Mohammed (CS20B001)

Problem 4 — 8 marks

Consider the following randomized algorithm.

Algorithm 1:

Set $X \leftarrow 0$
repeat n **times**
  Set $X \leftarrow X + 1$ with probability $1/2^X$
Set $Y \leftarrow 2^X - 1$

(a) (5 marks) Compute $\mathbb{E}[Y]$.

Solution:

Let X_t denote the value of X after t iterations and $P(t, i)$ denote $Pr(X_t == i)$.

$E[Y] = \sum_{i=1}^{n} P(n, i) * (2^i - 1)$

Let us expand $P(n, i)$

$P(n, i) = P(n - 1, i - 1) * 1/2^{i-1} + P(n - 1, i) * (1 - 1/2^i)$

$P(n, i) = \frac{P(n-1,i-1)}{2^{i-1}} - \frac{P(n-1,i)}{1/2^i} + P(n - 1, i)$

$P(n, i) * (2^i - 1) = 2^i * P(n - 1, i) + 2P(n - 1, i - 1) - 2P(n - 1, i) + \frac{P(n-1,i)}{1/2^i} - \frac{P(n-1,i-1)}{2^{i-1}}$

$\sum_{i=1}^{n} P(n, i)(2^i - 1) = \sum_{i=1}^{n-1} P(n - 1, i)(2^i) + 2P(n - 1, 0) - 2P(n - 1, n) + \frac{P(n-1,n)}{1/2^n} - \frac{P(n-1,0)}{2^0}$

$\sum_{i=1}^{n} P(n, i)(2^i - 1) = \sum_{i=1}^{n-1} P(n - 1, i)(2^i - 1) + \sum_{i=1}^{n-1} P(n - 1, i)$

$\sum_{i=1}^{n} P(n, i)(2^i - 1) = \sum_{i=1}^{n-1} P(n - 1, i)(2^i - 1) + 1$

$\sum_{i=1}^{n} P(n, i)(2^i - 1) = \sum_{i=1}^{1} P(1, i)(2^i - 1) + n - 1$

$\sum_{i=1}^{n} P(n, i)(2^i - 1) = n$

$\therefore$ $E[Y]$ is n.

(b) (3 marks) Give a tight bound on the number of bits required to represent X. Notice that the number of bits required to represent X is a random variable.

Solution:

We use Markov's Inequality :

$Pr(Y \geq a) \leq \frac{E[y]}{a}$

$E[Y] = n$, Let us take $a = n * c$

$Pr(Y \geq nc) \leq 1/c \longrightarrow Pr(Y < nc) \geq 1 - 1/c$

$X = log_2(Y + 1) \longrightarrow Pr(X < log_2(nc + 1)) \geq 1 - 1/c$

Let us take $c = n$, $Pr(X < log_2(n^2 + 1)) \geq 1 - 1/n$

$\therefore$ With a probability at least of $1 - 1/n$, X requires at most $log_2(log_2(n^2 + 1))$

Or in a generalized manner: $c = 1/\delta$

$\therefore$ With a probability at least of $1 - \delta$, X requires at most $log_2(log_2(n/\delta + 1))$

Problem 5 — 5 marks

A collection of n bits $X_1, X_2, \ldots, X_n$ are said to be k-wise independent if for every subset S of

k bits among the n, and for $b_1, b_2, \ldots, b_k \in \{0, 1\}$, we have

$$\Pr\left[\bigcap_{i \in S} X_i = b_i\right] = \prod_{i \in S} \Pr\left[X_i = b_i\right]$$

Consider the following construction: Let $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \in \{0, 1\}^\ell$ be n vectors such that every set of k vectors are linearly independent over $\mathbb{F}_2$. Let $\mathbf{y} \in \{0, 1\}^\ell$ be chosen uniformly at random. Define X_is as follows:

$$X_i = \left(\sum_{j=1}^{\ell} \mathbf{x}_{i,j} \mathbf{y}_j\right) \bmod 2.$$

Show that X_is are k-wise independent.

Solution: Type your solution here.