

# Verifying String Equalities

**Application:** Comparing the consistency of two replicates of a database.

Given two long strings  $a = a_1, \dots, a_n$  and  $b = b_1, \dots, b_n$  we want to check equality.

For a string  $x = x_1, \dots, x_n$  define the “fingerprint” of  $x$

$$F_p(x) = \left( \sum_{i=1}^n x_i 2^{i-1} \right) \text{ mod } p$$

**Theorem 1.** Let  $p$  be a random prime number in the range  $[1, \dots, 2n^2 \log n]$ .

$$\Pr(F_p(a) = F_p(b) \mid a \neq b) = O\left(\frac{1}{n}\right).$$

**Proof.** If  $a \neq b$  then we get equality only if  $p$  divides  $c = \sum_{i=1}^n (a_i - b_i)2^{i-1}$ .

Since  $c \leq 2^n$ , the number of different primes that divide  $c$  is bounded by  $n$ .

The number of primes in the range  $[1, \dots, T]$  is about  $T/\log T$ .

Thus, there are  $\Omega(n^2)$  primes in the range  $[1, \dots, 2n^2 \log n]$ . Less than  $n$  primes can divide  $c$ .  $\square$

**Application:** We can compare the consistency of two replicates of a database of  $n$  bits by exchanging only  $O(\log n)$  bits.

# Min-Cut Algorithm

**Input:** A  $n$  node graph

**Output:** A minimal set of edges that disconnects the graph.

1. **repeat**  $n - 2$  **times:**

(a) Pick an edge uniformly at random.

(b) Contract the edge.

**endrepeat**

2. **output** the set of edges connecting the two remaining vertices.

**Theorem 2.** *The algorithm outputs a min-cut set of edges with probability  $\geq \frac{2}{n(n-1)}$ .*

**Lemma 1.** *Contraction operation (step 1(b)) does not reduce the size of the min-cut set.*

**Proof.** Every cut set in the new graph is a cut set in the original graph.  $\square$

## Analysis of the Algorithm

Assume that the graph has a min-cut set of  $k$  edges.

We compute the probability of finding one such set  $C$ .

**Lemma 2.** *Let  $C$  be a min-cut set of  $G$ . If the run of the algorithm did not contract any edge of  $C$ , it also did not eliminate any edge of  $C$ .*

Since the minimum cut-set has  $k$  edges, all vertices have degree  $\geq k$ , and the graph has  $\geq nk/2$  edges.

Let  $E_i =$  "the edge contracted in iteration  $i$  is not in  $C$ ".

We want

$$\Pr(\cap_{i=1}^{n-2} E_i) = \Pr(E_1) \Pr(E_2|E_1) \Pr(E_3|E_2 \cap E_1) \dots \Pr(E_{n-2} | \cap_{i=1}^{n-3} E_i).$$

$$\Pr(E_1) \geq 1 - \frac{k}{nk/2} = 1 - \frac{2}{n}$$

$$\Pr(E_2|E_1) \geq 1 - \frac{k}{(n-1)k/2} = 1 - \frac{2}{n-1}$$

$$\Pr(E_i | \cap_{j=1}^{i-1} E_j) \geq 1 - \frac{2}{n-i+1}$$

$$\text{Thus, } \Pr(\cap_{i=1}^{n-2} E_i) \geq \prod_{i=1}^{n-2} \left(1 - \frac{2}{n-i+1}\right) = \prod_{i=1}^{n-2} \left(\frac{n-i-1}{n-i+1}\right) = \frac{2}{n(n-1)}$$

# Min-Cut: High Probability Algorithm

**Input:** A  $n$  node graph

**Output:** A minimal set of edges that disconnects the graph.

1. **for**  $i = 1$  **to**  $n^2 \ln n$

    1.1. **repeat**  $n - 2$  **times:**

        (a) Pick an edge uniformly at random.

        (b) Contract the edge.

**endrepeat**

    1.2 Let  $C_i$  be the set of edges connecting the two remaining vertices.

**endfor**

2. **output** the set with the minimum size among the  $C_i$ 's.

**Theorem 3.** *The high probability algorithm outputs a min-cut set with probability at least  $1 - 1/n^2$ .*

# Random Variable

Let  $(\mathcal{S}, Pr)$  be a discrete probability space.

Let  $V$  be a set of values.

A random variable  $X$  defined on  $(\mathcal{S}, Pr)$  is a function

$$X : \mathcal{S} \rightarrow V$$

Let  $\mathcal{E}(r) = \{s \in \mathcal{S} \mid X(s) = r\}$

$$Pr(X = r) = Pr(\mathcal{E}(r)) = \sum_{s \in \mathcal{E}(r)} Pr(s).$$

Two random variables  $X$  and  $Y$  (defined on the same sample space) are called independent if for all  $x$  and  $y$

$$Pr\{X = x \text{ and } Y = y\} = Pr\{X = x\} Pr\{Y = y\}$$

Example 1: In rolling a dice, the number that comes up is a random variable.

Example 2: Consider a gambling game in which a player flips two coins, if he gets head in both coins we wins \$3, else he losses \$1. The payoff of the game is a random variable.

**Definition 1.** *The expectation of a discrete random variable  $X$  is*

$$E[X] = \sum_{i \in \text{range}(X)} i \Pr(X = i).$$

The expectation (or mean or average) is a weighted sum over all possible values of the random variable.

Example: The expected value of one dice roll is:

$$E[X] = \sum_{i=1}^6 i \Pr(X = i) = \sum_{i=1}^6 \frac{i}{6} = 3\frac{1}{2}.$$

Consider a game in which a player chooses a number in  $[1, \dots, 6]$  and then rolls 3 dice.

The player wins \$1 for each dice the matches the number, he losses \$1 if no dice matches the number.

What is the expected outcome of that game:

$$-1\left(\frac{5}{6}\right)^3 + 1 \cdot 3\left(\frac{1}{6}\right)\left(\frac{5}{6}\right)^2 + 2 \cdot 3\left(\frac{1}{6}\right)^2\left(\frac{5}{6}\right) + 3\left(\frac{1}{6}\right)^3 = -\frac{17}{216}.$$

Which gambling game would you prefer?

- We flip one coin, you win \$1 if head, loose one \$1 if tail.
- We flip 10 coins, you win  $\$2^{10} = 1K$  if all heads, else you pay \$1.
- We flip 20 coins, you win  $\$2^{20}/2 = M/2$  if all heads, else you pay \$1.

# Linearity of Expectation

**Theorem 4.** For any two random variables  $X$  and  $Y$

$$E[X + Y] = E[X] + E[Y].$$

**Proof.**

$$\begin{aligned} E[X + Y] &= \\ \sum_{i \in \text{range}(X)} \sum_{j \in \text{range}(Y)} (i + j) \Pr((X = i) \cap (Y = j)) &= \\ \sum_i \sum_j i \Pr((X = i) \cap (Y = j)) + & \\ \sum_j \sum_i j \Pr((Y = j) \cap (X = i)) &= \\ \sum_i i \Pr(X = i) + \sum_j j \Pr(Y = j). & \end{aligned}$$

□

(Since we sum over all possible choices of  $i$  ( $j$ ).)

**Theorem 5.** *If  $E_1, E_2, \dots, E_k$  are disjoint events such that  $\sum_{i=1}^k Pr(E_i) = 1$  then for any event  $B$ ,*

$$\sum_{i=1}^k Pr(B \cap E_i) = Pr(B).$$

## Examples:

1. The expectation of the sum of two dice is 7, even if they are not independent.

2. Assume that we flip  $N$  coins, what is the expected number of heads?

Using linearity of expectation we get  $N \cdot \frac{1}{2}$ .

By direct summation we get  $\sum_{i=0}^N i \binom{N}{i} 2^{-N}$ .

Thus we prove

$$\sum_{i=0}^N i \binom{N}{i} 2^{-N} = \frac{N}{2}.$$

3. Assume that  $N$  people checked coats in a restaurant. The coats are mixed and each person gets a random coat.

How many people got their own coats?

It's hard to compute  $E[X] = \sum_{k=0}^N k Pr(X = k)$ . Instead we define  $N$  0-1 random variables  $X_i$ , where  $X_i = 1$  iff  $i$  got his coat.

$$E[X_i] = 1 \cdot Pr(X_i = 1) + 0 \cdot Pr(X_i = 0) =$$

$$Pr(X_i = 1) = \frac{1}{N}.$$

$$E[X] = \sum_{i=1}^N E[X_i] = 1.$$

# Randomized Quicksort

Procedure  $Q\_S(S)$ ;

**Input:** A set  $S$ .

**Output:** The set  $S$  in sorted order.

1. If  $|S| \leq 1$  then return  $S$ , else
- 2.(a) Choose a random element  $y$  uniformly from  $S$ .  
(b) Compare all elements of  $S$  to  $y$ . Let

$$S_1 = \{x \in S - \{y\} \mid x \leq y\}$$

$$S_2 = \{x \in S - \{y\} \mid x > y\}.$$

(Elements in  $S_1$  and  $S_2$  are in the same order as in  $S$ .)

- (c) Return the list:

$$Q\_S(S_1), y, Q\_S(S_2).$$

Let  $T$  = number of comparisons in a run of QuickSort.

### Theorem 6.

$$E[T] = O(n \log n).$$

#### Proof:

Let  $s_1, \dots, s_n$  be the elements of  $S$  in sorted order.

For  $i = 1, \dots, n$ , and  $j > i$ , define 0-1 random variable  $X_{i,j}$ , s.t.

$X_{i,j} = 1$  iff  $s_i$  is compared to  $s_j$  in the run of the algorithm.

The number of comparisons in running the algorithm is

$$T = \sum_{i=1}^n \sum_{j>i} X_{i,j}.$$

We are interested in  $E[T]$ .

What is the probability that  $X_{i,j} = 1$ ?

$s_i$  is compared to  $s_j$  iff either  $s_i$  or  $s_j$  is chosen as a “split item” before any of the  $j - i - 1$  elements between  $s_i$  and  $s_j$  are chosen.

Elements are chosen uniformly at random  $\rightarrow$  elements in the set  $[s_i, s_{i+1}, \dots, s_j]$  are chosen uniformly at random.

$$\Pr(X_{i,j} = 1) = \frac{2}{j - i + 1}.$$

$$E[X_{i,j}] = \frac{2}{j - i + 1}.$$

$$\begin{aligned}
E[T] &= E\left[\sum_{i=1}^n \sum_{j>i} X_{i,j}\right] = \\
&\sum_{i=1}^n \sum_{j>i} E[X_{i,j}] = \sum_{i=1}^n \sum_{j>i} \frac{2}{j-i+1} \leq \\
&\sum_{i=1}^n \sum_{k=1}^{n-i+1} \frac{2}{k} \leq 2 \sum_{i=1}^n \sum_{k=1}^n \frac{1}{k} = 2nH_n = n \log n + O(n)
\end{aligned}$$

# Probabilistic Analysis of QuickSort

**Theorem 7.** *The expected run time of (deterministic) Quicksort on a random input, uniformly chosen from all possible permutation of  $S$  is  $O(n \log n)$ .*

**Proof.**

Set  $X_{i,j}$  as before.

If all permutations have equal probability, all permutations of  $S_i, \dots, S_j$  have equal probability, thus

$$Pr(X_{i,j}) = \frac{2}{j - i + 1}.$$

$$E\left[\sum_{i=1}^n \sum_{j>i} X_{i,j}\right] = O(n \log n).$$

□

## Randomized Algorithms:

- Analysis is true for **any** input.
- The sample space is the space of random choices made by the algorithm.
- Repeated runs are independent.

## Probabilistic Analysis:

- The sample space is the space of all possible inputs.
- If the algorithm is **deterministic** repeated runs give the same output.

# Randomized Algorithm classification

A **Monte Carlo Algorithm** is a randomized algorithm that may produce an incorrect solution.

For decision problems: A **one-side error** Monte Carlo algorithm errs only on one possible output, otherwise it is a **two-side error** algorithm.

A **Las Vegas** algorithm is a randomized algorithm that **always** produces the correct output.

In both types of algorithms the run-time is a random variable.