Types of randomized algorithms

Monte Carlo

- running time is deterministic
- correctness is a random variable
- example: minimum cut

Las Vegas

- always correct
- running time is a random variable
- example: quicksort

Errors and certainty (1)

Success probability amplification: run the Monte Carlo algorithm repeatedly many times.
If one run succeeds with probability $\geq 1/2$, then with probability $\geq 1 - \frac{1}{2^k}$ at least one out of k independent runs succeeds.

Transformation

Monte Carlo $\longrightarrow$ Las Vegas
Suppose that the algorithm succeeds with probability $\geq 1/2$ and we can efficiently verify the correctness of a solution.
Run the Monte Carlo algorithm repeatedly, until it succeeds.
The expected number of iterations is at most 2.

Markov's inequality

Let X be a random variable that takes only nonnegative values. Then,

$$\Pr[X \geq k\mathbf{E}X] \leq \frac{1}{k}.$$

Chebyshev's inequality

Let X be a random variable. $\mathbf{Var}X = \mathbf{E}[(X - \mathbf{E}X)^2]$. Then,

$$\Pr[|X - \mathbf{E}X| \geq t\sqrt{\mathbf{Var}X}] \leq \frac{1}{t^2}.$$

(Proof: apply Markov's inequality to the r.v. $Y = (X - \mathbf{E}X)^2$.)

Example: binomial r.v.

X_n = the number of heads in n tosses of a fair coin.

$$\mathbf{E}X_n = n \cdot \mathrm{Pr}[\text{heads}] = \frac{n}{2}.$$

$$\mathbf{Var}X_1 = \frac{1}{4}, \mathbf{Var}X_n = \frac{n}{4}.$$

(variance of sum = sum of variances for independent r.v.)
For an unfair coin ($\mathrm{Pr}[\text{heads}] = p$),

$$\mathbf{E}X_n = np, \quad \mathbf{Var}X_n = np(1-p).$$

Randomized selection

Input: set S of n numbers, integer $k \leq n$.
Output: the k-th smallest element $S_{(k)}$ of S.

Idea:

Sample S to get a smaller subset P, then find the right element in P.

- With high probability, $S_{(k)} \in P$.
- P is not very large so sorting it is not too expensive.

Randomized selection

Input: set S of n numbers, integer $k \leq n$.
Output: the k-th smallest element $S_{(k)}$ of S.

1. Select $n^{3/4}$ elements of S uniformly *with replacement* $\to R$.
2. Sort R in time $O(n^{3/4} \lg n)$.
3. Let $a = R_{(l)}$ and $b = R_{(h)}$, where $l, h = \frac{k}{n^{1/4}} \pm \sqrt{n}$.
4. Let P be the elements of S between a and b.
   If $S_{(k)} \notin P$, or if $|P| > 4n^{3/4} + 2$, repeat steps 1–3.
5. Sort P, output $S_{(k)} = P_{(k - r_S(a) + 1)}$.

Case 1: $n^{1/4} < k < n - n^{1/4}$

$P = \{y \in S \mid a \leq y \leq b\}$.

Theorem

With probability $1 - O(n^{-1/4})$, $S_{(k)}$ is found in the first pass and thus only $2n + o(n)$ comparisons are made.

Randomized selection analysis (1)

If only one pass, only $2n + o(n)$ comparisons.
Failure modes:

- a too large: $a > S_{(k)}$.
- b too small: $b < S_{(k)}$.
- P too large: $|P| > 4n^{3/4} + 2$.

Failure mode 1: $a > S_{(k)}$

$a = R_{(l)}$.
$S_{(k)} \notin P$ iff not enough samples in R are $\leq S_{(k)}$.
Let $X_i = 1$ if the i-th random sample is $\leq S_{(k)}$, 0 otherwise.
Then $\Pr[X_i = 1] = k/n$. Let $X = \sum_i X_i$.
Now $\mathbf{E}X = \frac{k}{n^{1/4}}$ and
$\mathbf{Var}X = n^{3/4}(\frac{k}{n})(\frac{n-k}{n}) \leq \frac{n^{3/4}}{4}$.
Using Chebyshev's inequality:

$$\Pr[|X - \mathbf{E}X| \geq \sqrt{n}] = \Pr[|X - \mathbf{E}X| \geq (2n^{1/8})(n^{3/8}/2)] = O(\frac{1}{n^{1/4}}).$$

Failure mode 2: $b < S_{(k)}$

Symmetric to failure mode 1. $\Pr[b < S_{(k)}] = O(\frac{1}{n^{1/4}})$.
Now probability that we fail in either of the two ways is at most $O(\frac{1}{n^{1/4}}) + O(\frac{1}{n^{1/4}}) = O(\frac{1}{n^{1/4}})$.

Failure mode 3: $|P| > 4n^{3/4} + 2$

Similar to the other two cases.

Random select: remarks

- expected running time is $2n + o(n)$.
- best known deterministic algorithm: $3n$ worst case
- deterministic algorithms cannot do better than $2n$
- randomized algorithm can be improved to $n + \min\{k, n-k\} + o(n)$

Coupon collector's problem

Start with n empty bins.
Random process: in each step, a ball is placed randomly in one of the bins.
How long until all the bins are full?

Coupon collector: modeling

X = the number of steps untill all bins are full.
Define random variables properly:
X_0 = number of steps until 1 bin is full,
X_1 = number of steps after 1 bin is full, until 2 bins are full,
...
X_i = number of steps after i bins are full, until $i + 1$ bins are full.
(Epochs 1, 2, ..., n.)
Now,

$$X = X_0 + X_2 + \cdots + X_{n-1}.$$

Coupon collector: expectation

Let p_i = probability that the $(i+1)$-th bin is filled in any step in i-th epoch.
Then,

$$p_i = \frac{n-i}{n}.$$

$$\mathbf{E}X_i = \frac{1}{p_i} = \frac{n}{n-i}.$$

$$\mathbf{E}X = \sum_{i=0}^{n-1} \mathbf{E}X_i = \sum_{i=0}^{n-1} \frac{n}{n-i} = n\sum_{i=1}^{n} \frac{1}{i} = nH_n.$$