Random variables and expectation

RA-MIRI QT Curs 2020-2021

Most if the material included here is based on Chapter 13 of Kleinberg & Tardos Algorithm Design book.

Waiting for a first success

- A coin is heads with probability p and tails with probability $1{-}p$.
- How many independent flips we expect to get heads for the first time?
- Let X the random variable that gives the number of flips. Observe that
$$Pr[X = j] = (1-p)^{j-1}p$$
and
$$E[X] = \sum_{j=1}^{\infty} jPr[X = j] = \sum_{j=1}^{\infty} (1-p)^{j-1}p = \frac{p}{1-p}\sum_{j=1}^{\infty} j(1-p)^j$$
as $\sum_{j=1}^{\infty} jx^j = \frac{x}{(1-x)^2}$, we have
$$E[X] = \frac{p}{1-p}\frac{1-p}{p^2} = \frac{1}{p}$$

Bernoulli process

- A Bernoulli process denotes a sequence of experiments, each of them a with binary output: success (1) with probability p, and failure (0) with prob. $q = 1 - p$.
- A nice thing about Bernoulli distributions: it is natural to define a indicator r.v.

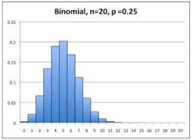
  $X = 1$ if the output is 1, otherwise $X = 0$.

  Clearly, $E[X] = p$

The binomial distribution

A r.v. X has a Binomial distribution with parameter p ($B(n,p)$) if X counts the number of successes during n trials of a Bernoulli experiments having probability of success p.

$$\mathbf{Pr}[X = k] = \binom{n}{k} p^k (1-p)^{n-k}.$$



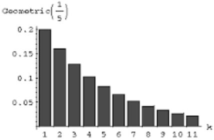
Let $X \in B(n,p)$, to compute $\mathbf{E}[X]$, we define indicator r.v. $\{X_i\}_{i=1}^n$, where $X_i = 1$ iff the i-th output is 1, otherwise $X_i = 0$.

Then $X = \sum_{i=1}^n X_i \Rightarrow \mathbf{E}[X] = \mathbf{E}[\sum_{i=1}^n X_i] = \sum_{i=1}^n \underbrace{\mathbf{E}[X_i]}_{=p} = np.$

The Geometric distribution

A r.v. X has a Geometric distribution with parameter p ($X \sim G(p)$) if X counts the number of trials until the first success.

If $X \in G(p)$ then
$\mathbf{Pr}[X = k] = (1-p)^{k-1}p$,
$\mathbf{E}[X] = \frac{1}{p}$.



Random generators

Consider a sequential random generator of n bits, so that the probability that a bit is 1 is p.

- If $X = \#$ number of 1's in the generated n bit number, $X \in B(n, p)$.
- If $Y = \#$ bits in the generated number until the first 1, $Y \in G(p)$.

Coupon collector

Each box of cereal contains a coupon. There are n different types of coupons. Assuming all boxes are equally likely to contain each coupon, how many boxes before you have at least 1 coupon of each type?

Claim

The expected number of steps is $\Theta(nlogn)$.

Proof.

- Phase j = time between j and $j+1$ distinct coupons.
- Let X_j = number of steps you spend in phase j.
- Let X = total number of steps,
  of course, $X = X_0 + X_1 + \cdots + X_{n-1}$.

Coupon collector

X_j = number of steps you spend in phase j.

- We can consider a Bernoulli process that succeeds when we hit one of the still not collected coupons.
- The probability of success is $\frac{n-j}{n}$.
- X_j counts the time until the Bernoulli process reaches a success, therefore

$$E[X_j] = \frac{n}{n-j}$$

Coupon collector

X = total number of steps
Using the decomposition in sums of indicator r.v. we have

$$
\begin{aligned}
E[X] &= E[X_0] + E[X_1] + \cdots + E[X_{n-1}] \\
&= \sum_{j=0}^{n-1} \frac{n}{n-j} = n \sum_{i=1}^{n} \frac{1}{n} = nH(n) \approx n \log n
\end{aligned}
$$

A randomized approximation algorithm for MAX 3-SAT

A 3-SAT formula is a Boolean formula in CNF such that each clause has exactly 3 literals and each literal corresponds to a different variable.

$$(x_2 \vee \overline{x}_3 \vee \overline{x}_4) \wedge (x_2 \vee x_3 \vee \overline{x}_4) \wedge (\overline{x}_1 \vee x_2 \vee x_4) \wedge (\overline{x}_1 \vee \overline{x}_2 \vee x_3) \wedge (x_1 \vee \overline{x}_2 \vee \overline{x}_4)$$

MAXIMUM 3-SAT. Given a 3-SAT formula, find a truth assignment that satisfies as many clauses as possible.

The problem is NP-hard. We can try to design a randomized algorithm that produces a good assignment, even if it is not optimal.

A randomized approximation algorithm for MAX 3-SAT

Algorithm. For each variable, flip a fair coin, and the variable to True (1) if it is heads, to False (0) otherwise.

Note that a variable gets 1 with probability $\frac{1}{2}$, and this assignment is made independently of the other variables.

What is the expected number of satisfied clauses?

Assume that the 3-SAT formula has n variables and m clauses.

- Let $Z =$ number of clauses satisfied by the random assignment
- For $1 \leq j \leq m$, define the random variables
  $Z_j = 1$ if clause j is satisfied, 0 otherwise.
- By definition, $Z = \sum_{j=1}^{m} Z_j$.
- $Pr[Z_j = 1] = 1 - (1/2)^3 = 7/8$, so $E[Z_j] = 7/8$. Therefore ,

$$E[Z] = \sum_{j=1}^{m} E[Z_j] = \frac{7}{8}m$$

A randomized approximation algorithm for MAX 3-SAT

How good is the solution computed by the random algorithm?

- For a 3-CNF formula let $opt(F)$ be the maximum number of clauses than can be satisfied by an assignment.
- As for any assignment x the number of satisfied clauses is always $\leq opt(F)$, we have that $E[Z] \leq opt(F)$.
- Of course $opt(F) \leq m$, that is $\frac{7}{8}opt(F) \leq \frac{7}{8}m = E[Z]$, then

$$\frac{opt(F)}{E[Z]} \leq \frac{8}{7}$$

We have a $\frac{8}{7}$-approximation algorithm for $\textsc{Max 3-Sat}$.

The probabilistic method

Claim

For any instance of 3-SAT, there exists a truth assignment that satisfies at least a 7/8 fraction of all clauses.

Proof. Random variable must have one event on which the measured value is at least its expectation.

Probabilistic method. [Paul Erdös] Prove the existence of a non-obvious property by showing that a random construction produces it with positive probability

Random-Quicksort

Input: An array A holding n keys. For simplicity we assumed that all keys are different.
Output: A sorted in increasing order.

I'm assuming that all of you known:

- The Quick sort algorithm which has $O(n^2)$ cost
- and $O(n \log n)$ average cost.
- One randomized version randomly sorts the input and then applies the deterministic algorithm, having average running time $O(n \log n)$
- Here we consider another randomized version of Quick sort.

Random-Quicksort

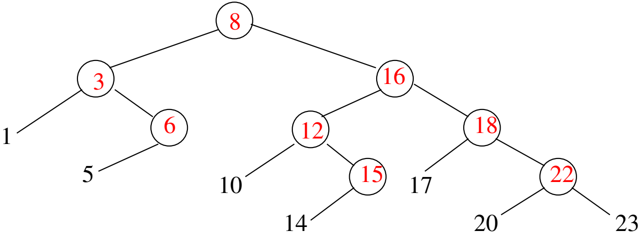
Ran-Quicksort (A)
if $A.size() \leq 3$ **then**
  Sort A using insertion sort
  return A
Choose an element $a \in A$ uniformly at random
Put in B all elements $< a$ and in C all elements $> a$
$B =$ **Ran-Quicksort** (B)
$C =$ **Ran-Quicksort** (C)
return B followed by a followed by C

The main difference is that we perform a random partition in each call around the random pivot a.

Example

A={1,3,5,6,8,10,12,14,15,16,17,18,20,22,23}

Ran–Partition of input



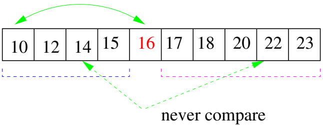
Expected Complexity of Ran-Partition

Taken from CMU course 15451-07
`https://www.cs.cmu.edu/afs/cs/academic/class/15451-s07/www/lecture_notes/lect0123.pdf`

- The expected running time $T(n)$ of Rand-Quicksort is dominated by the number of comparisons.
- Every Rand-Partition has cost
  $\Theta(1) + \Theta(\underbrace{\text{number of comparisons}}_{A.size()})$
- If we can count the number of comparisons, we can bound the the total time of Quicksort.
- Let X be the number of comparisons made in all calls of Ran-Quicksort
- X is a r.v. as it depends of the random choices of the element used to do a Ran-Partition

Expected Complexity of Ran-Partition

- Note: In the first application of Ran-Partition the selected a compares with all $n-1$ elements.
- Key observation: Any two keys are compared iff one of them is selected as pivot, and they are compared at most one time.



Denote the i-th smallest element in the array by z_i and define the indicator r.v.:

$$X_{ij} = \begin{cases} 1 & \text{if } z_i \text{ is compared to } z_j, \\ 0 & \text{otherwise.} \end{cases}$$

Then, $X = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{i,j}$
(this is true because we never compare a pair more than once)

$$\mathbf{E}[X] = \mathbf{E}\left[\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{i,j}\right] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \mathbf{E}[X_{i,j}]$$

As $\mathbf{E}[X_{i,j}] = 0\mathbf{Pr}[X_{i,j} = 0] + 1\mathbf{Pr}[X_{i,j} = 1]$

$\therefore \mathbf{E}[X_{i,j}] = \mathbf{Pr}[X_{i,j} = 1] = \mathbf{Pr}[z_i \text{ is compared to } z_j]$

- If the pivot we choose is between z_i and z_j then we never compare them to each other.
- If the pivot we choose is either z_i or z_j then we do compare them.
- If the pivot is less than z_i or greater than z_j then both z_i and z_j end up in the same partition and we have to pick another pivot.
- So, we can think of this like a dart game: we throw a dart at random into the array: if we hit z_i or z_j then X_{ij} becomes 1, if we hit between z_i and z_j then X_{ij} becomes 0, and otherwise we throw another dart.
- At each step, the probability that $X_{ij} = 1$ conditioned on the event that the game ends in that step is exactly $2/(j - i + 1)$. Therefore, overall, the probability that $X_{ij} = 1$ is $2/(j - i + 1)$.

End of the computation

$$
\begin{aligned}
\mathbf{E}[X] &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \mathbf{E}[X_{i,j}] \\
&= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} \\
&= 2 \cdot \sum_{i=1}^{n} \left(\frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n-i+1}\right) \\
&< 2 \cdot \sum_{i=1}^{n} \left(\frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}\right) \\
&= 2 \cdot \sum_{i=1}^{n} H_n = 2 \cdot n \cdot H_n = O(n \lg n).
\end{aligned}
$$

Therefore, $\mathbf{E}[X] = 2n \ln n + \Theta(n)$.

Main theorem

Theorem

The expected complexity of Ran-Quicksort is $\mathbf{E}[T_n] = O(n \lg n)$.

Selection and order statistics

Problem: Given a list A of n of unordered distinct keys, and a $i \in \mathbb{Z}, 1 \leq i \leq n$, select the element $x \in A$ that is larger than exactly $i - 1$ other elements in A.

Notice if:

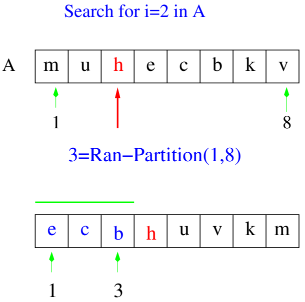
1. $i = 1 \Rightarrow$ MINIMUM element
2. $i = n \Rightarrow$ MAXIMUM element
3. $i = \lfloor \frac{n+1}{2} \rfloor \Rightarrow$ the MEDIAN
4. $i = \lfloor 0.9 \cdot n \rfloor \Rightarrow$ *order statistics*

Sort A ($O(n \lg n)$) and search for $A[i]$ ($\Theta(n)$).
Can we do it in linear time?
Yes, we saw it in the Algorismia class a deterministic linear time algorithm for selection with a bad constant.

Quick-Select

Given unordered $A[1, \ldots, n]$ return the i-th. element

- Quick-Select $(A[p, \ldots, q], i)$
- $r =$ Ran-Partition (p, q) to find position of pivot and partition the array
- if $i = r$ return $A[r]$
- if $i < r$ Quick-Select $(A[p, \ldots, r-1], i)$
- else Quick-Select $(A[r+1, \ldots, q], i)$



Analysis

Theorem

Given $A[1,\ldots,n]$ and i, the expected number of steps for Quick-Select to find the i-th. element in A is $O(n)$

- The algorithm is in phase j when the size of the set under consideration is at most $n(3/4)^j$ but greater than $n(3/4)^{j-1}$
- We bound the expected number of iterations spent in phase j.
- An element is central if at least a quarter of the elements are smaller and at least a quarter of the elements are larger.
- If a central element is chosen as pivot, at least a quarter of the elements are dropped. So, the set shrinks by a $3/4$ factor or better.
- As, half of the elements are central, the probability of choosing as pivot a central element is $1/2$.
- So, the expected number of iterations in phase j is 2.

Analysis

- Let X = number of steps taken by the algorithm.
- Let X_j = number of steps in phase j. We have $X = X_0 + X_1 + X_2 + \ldots$
- An iteration in phase j requires at most $cn(3/4)^j$ steps, for some constant c.
- Therefore, $E[X_j] = 2cn(3/4)^j$ and by linearity of expectation.

$$E[X] = \sum_j E[X_j] \leq \sum_j 2cn \left(\frac{3}{4}\right)^j = 2cn \sum_j \left(\frac{3}{4}\right)^j \leq 8cn$$