Randomized algorithms

Inge Li Gørtz

Thank you to Kevin Wayne for inspiration to slides

Randomized algorithms

- Last week
  - Contention resolution
  - Global minimum cut
- Today
  - Expectation of random variables
    - Guessing cards
  - Three examples:
    - Median/Select.
    - Quick-sort

Random Variables and Expectation

Random variables

- A random variable is an entity that can assume different values.
- The values are selected "randomly"; i.e., the process is governed by a probability distribution.
- Examples: Let X be the random variable "number shown by dice".
  - X can take the values 1, 2, 3, 4, 5, 6.
  - If it is a fair dice then the probability that X = 1 is 1/6:
    - P[X=1] =1/6.
    - P[X=2] =1/6.
    - …

Expected values

- Let X be a random variable with values in $\{x_1,\ldots,x_n\}$, where x_i are numbers.
- The expected value (expectation) of X is defined as

$$E[X] = \sum_{j=1}^{n} x_j \cdot \Pr[X = x_j]$$

- The expectation is the theoretical average.
- Example:
  - X = random variable "number shown by dice"

$$E[X] = \sum_{j=1}^{6} j \cdot \Pr[X = j] = (1+2+3+4+5+6) \cdot \frac{1}{6} = 3.5$$

Waiting for a first succes

- Coin flips. Coin is heads with probability p and tails with probability $1-p$. How many independent flips X until first heads?
  - Probability of $X = j$? (first succes is in round j)

$$\Pr[X = j] = (1-p)^{j-1} \cdot p$$

  - Expected value of X:

$$E[X] = \sum_{j=1}^{\infty} j \cdot \Pr[X = j]$$
$$= \sum_{j=1}^{\infty} j \cdot (1-p)^{j-1} \cdot p$$
$$= \frac{p}{1-p} \sum_{j=1}^{\infty} j \cdot (1-p)^{j}$$
$$= \frac{p}{1-p} \cdot \frac{1-p}{p^2} = \frac{1}{p}$$

$$\sum_{k=0}^{\infty} k \cdot x^k = \frac{x}{(1-x)^2}$$

for $|x| < 1$.

Properties of expectation

- If we repeatedly perform independent trials of an experiment, each of which succeeds with probability $p > 0$, then the expected number of trials we need to perform until the first succes is $1/p$.
- If X is a 0/1 random variable, $E[X] = \Pr[X = 1]$.
- Linearity of expectation: For two random variables X and Y we have

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

Guessing cards

- Game. Shuffle a deck of n cards; turn them over one at a time; try to guess each card.
- Memoryless guessing. Can't remember what's been turned over already. Guess a card from full deck uniformly at random.
- Claim. *The expected number of correct guesses is 1.*
  - $X_i = 1$ if i^{th} guess correct and zero otherwise.
  - $X =$ the correct number of guesses $= X_1 + \ldots + X_n$.
  - $E[X_i] = \Pr[X_i = 1] = 1/n$.
  - $E[X] = E[X_1 + \cdots + X_n] = E[X_1] + \cdots + E[X_n] = 1/n + \cdots + 1/n = 1$.

Guessing cards

- Game. Shuffle a deck of n cards; turn them over one at a time; try to guess each card.
- Guessing with memory. Guess a card uniformly at random from cards not yet seen.
- Claim. *The expected number of correct guesses is* $\Theta(\log n)$.
  - $X_i = 1$ if i^{th} guess correct and zero otherwise.
  - $X =$ the correct number of guesses $= X_1 + \ldots + X_n$.
  - $E[X_i] = \Pr[X_i = 1] = 1/(n - i - 1)$.
  - $E[X] = E[X_1] + \cdots + E[X_n] = 1/n + \cdots + 1/2 + 1/1 = H_n$.

$\ln n < H(n) < \ln n + 1$

Coupon collector

- 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* $\Theta(n \log n)$.
  - Phase j = time between j and $j + 1$ distinct coupons.
  - X_j = number of steps you spend in phase j.
  - X = number of steps in total = $X_0 + X_1 + \cdots + X_{n-1}$.
  - $E[X_j] = n/(n - j)$.
  - The expected number of steps:

$$E[X] = E[\sum_{j=0}^{n-1} X_j] = \sum_{j=0}^{n-1} E[X_j] = \sum_{j=0}^{n-1} n/(n-j) = n \cdot \sum_{i=1}^{n} 1/i = n \cdot H_n.$$

Median/Select

Select

- Given n numbers S = {a_1, a_2, …, a_n}.
- Median: number that is in the middle position if in sorted order.
- Select(S,k): Return the kth smallest number in S.
  - Min(S) = Select(S,1), Max(S)= Select(S,n), Median = Select(S,n/2).
- Assume the numbers are distinct.

```
Select(S, k) {
   Choose a pivot s ∈ S uniformly at random.

   For each element e in S
      if e < s put e in S'
      if e > s put e in S''

   if |S'| = k-1 then return s

   if |S'| ≥ k then call Select(S', k)

   if |S'| < k then call Select(S'', k - |S'| - 1)
}
```

Select

```
Select(S, k) {

   Choose a pivot s ∈ S uniformly at random.

   For each element e in S
      if e < s put e in S'
      if e > s put e in S''

   if |S'| = k-1 then return s

   if |S'| ≥ k then call Select(S', k)

   if |S'| < k then call Select(S'', k - |S'| - 1)
 }
```

- Worst case running time: $T(n) = cn + c(n-1) + c(n-2) + \cdots = \Theta(n^2)$.
- If there is at least an ε fraction of elements both larger and smaller than s:

$$\begin{aligned} T(n) &= cn + (1-\varepsilon)cn + (1-\varepsilon)^2 cn + \cdots \\ &= \left(1 + (1-\varepsilon) + (1-\varepsilon)^2 + \cdots\right) cn \\ &\leq cn/\varepsilon. \end{aligned}$$

- Limit number of bad pivots.
- Intuition: A fairly large fraction of elements are "well-centered" => random pivot likely to be good.

Select

- Phase j: Size of set at most $n(3/4)^j$ and at least $n(3/4)^{j+1}$.
- Central element: at least a quarter of the elements in the current call are smaller and at least a quarter are larger.
- At least half the elements are central.
- Pivot central => size of set shrinks with by at least a factor 3/4 => current phase ends.
- Pr[s is central] = 1/2.
- Expected number of iterations before a central pivot is found = 2 =>
  expected number of iterations in phase j at most 2.
- X: random variable equal to number of steps taken by algorithm.
- X_j: expected number of steps in phase j.
- X = X_1+ X_2 + ….
- Number of steps in one iteration in phase j is at most $cn(3/4)^j$.
- $E[X_j] = 2cn(3/4)^j$.
- Expected running time: $E[X] = \sum_j E[X_j] \leq \sum_j 2cn\left(\frac{3}{4}\right)^j = 2cn \sum_j \left(\frac{3}{4}\right)^j \leq 8cn$

Quicksort

Quicksort

- Given n numbers S = {a_1, a_2, …, a_n} return the sorted list.
- Assume the numbers are distinct.

```
Quicksort(A,p,r) {

   if |S| ≤ 1 return S

   else

   Choose a pivot s ∈ S uniformly at random.

   For each element e in S
     if e < s put e in S'
     if e > s put e in S''

   L = Quicksort(S')
   R = Quicksort(S'')

   Return the sorted list L∘s∘R.
}
```

Quicksort: Analysis

- Worst case Quicksort requires Ω(n²) comparisons: if pivot is the smallest element in the list in each recursive call.
- If pivot always is the median then T(n) = O(n log n).
- for i < j: random variable

$$X_{ij} = \begin{cases} 1 & \text{if } a_i \text{ and } a_j \text{ compared by algorithm} \\ 0 & \text{otherwise} \end{cases}$$

- X total number of comparisons:

$$X = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}$$

- Expected number of comparisons:

$$E[X] = E[\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} E[X_{ij}]$$

Quicksort: Analysis

- Expected number of comparisons:

$$E[X] = E[\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} E[X_{ij}]$$

- Since X_{ij} only takes values 0 and 1: $E[X_{ij}] = \Pr[X_{ij} = 1]$
- a_i and a_j compared iff a_i or a_j is the first pivot chosen from $Z_{ij} = \{a_i, \ldots, a_j\}$.
- Pivot chosen independently uniformly at random ⇒ all elements from Z_{ij} equally likely to be chosen as first pivot from this set.
- We have $\Pr[X_{ij} = 1] = 2/(j - i + 1)$
- Thus

$$E[X] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} E[X_{ij}] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \Pr[X_{ij} = 1] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j - i + 1}$$

$$= \sum_{i=1}^{n-1} \sum_{k=2}^{n-i+1} \frac{2}{k} < \sum_{i=1}^{n-1} \sum_{k=1}^{n} \frac{2}{k} = \sum_{i=1}^{n-1} O(\log n) = O(n \log n)$$