EECS 495: Randomized Algorithms **Lecture 1**
Logistics, Overview, Application: Sorting

Reading: Text: chapter 1, Paper: *An introduction to randomized algorithms*, Karp 1991.

Logistics

The essentials:

- Website: linked to from my homepage
- Office Hours: by appointment
- Project: due at end of quarter
- Problem Sets: two problem sets, due in class
- Readings: from Randomized Algorithms, current research papers

The prereqs:

- discrete math: graph theory, big-oh notation, basic probability (random variables, expectation, variance, basic bounds)
- algorithms: run-time analysis, dynamic programming, LP-based algorithms, basic NP-hardness

Overview

A *randomized* algorithm takes:

- input
- string of random bits [[*For now, assume endless supply of truly random bits*]]

Note: same input may produce different outputs.

Advantages:

- reduced execution time/space requirements, simple to analyze/implement
- reduces det alg with bad worst-case behavior to alg that performs well on every input with high probability

Techniques:

- **foiling the adversary**:

  Example: Playing the lottery:

  - always pick the same number → adversary can guarantee you always lose
  - pick a random number → you win sometimes

  Idea:

  - An algorithm is a zero-sum game between
    - an adversary providing the inputs and
    - the algorithm designer providing alg

    * with adversary payoff being running time of alg
  - a randomized alg is a mixed strategy in the game
  - mixed strategies can guarantee higher payoffs.

- **random sampling**:

  Example: Sensitivity analysis:

  - given complicated system $f(x_1, \ldots, x_n)$ (boolean function), how sensitive system is to failure of i'th component x_i
  - sample inputs $x \in \{0,1\}^n$ with $x_i = 0$ and test for what fraction does $f = 0$

  Idea: random sample from population is representative of population as a whole

- **abundance of witnesses**:

  Example: Primality testing:

  - a factor p of a number n is a *witness* that n is composite
  - test random numbers to see if they are factors

  Idea: Witnesses

  - hard to find deterministically
  - if abundant enough, can sample and get one whp.

- **fingerprinting and hasing**:

  Example: Testing membership:

  - want to maintain set of objects
  - to implement "add", must test if object is already in set
  - map objects to small set of buckets based on "fingerprints"
  - compare new object to those in its bucket

  Idea:

  - represent a long string by a short fingerprint
  - use fingerprint to reduce search space/input size, or to test equality

- **random re-ordering**:

  Example: Binary search trees:

  - arrange input data into binary search tree
  - given order may create very unbalanced tree with naive alg
  - a random re-ordering is likely to give balanced tree with naive alg

  Idea: after re-ordering, input is unlikely to be pathological for naive algorithm

- **load balancing**

  Example: Machine scheduling:

  - send n printing jobs to n printers
  - pick arbitrary printer – worst-case load n
  - pick random printer – balls and bins, expected load $\log n$

  Idea:

  - randomization spreads load among resources
  - good for distributed environments

- **markov chains**

  Example: PageRank:

  - want to calculate probability random surfer lands at given page

- simulate random walk for "long enough" and count fraction of time spent at given page

Idea:

- many walks are rapidly mixing
- can use walks to efficiently samply from subspace
- useful for counting problems

Application: Sorting

Problem: Given

- a set S of n numbers

Output

- a list of members of S in ascending order

Algorithm:

- find *pivot* element $y \in S$ s.t. half of S is smaller than y
- partition $S \setminus \{y\}$ into S_1 and S_2 s.t.
  - S_1 is elts in S smaller than y
  - S_2 is elts in S larger than y
- recursively sort S_1 and S_2

Analysis: $T(n)$ is running time on input size n

- time to find y: cn for constant c
- time to partition: $(n-1)$ (compare each elt to y)

so

$$T(n) \leq 2T(n/2) + (c+1)n$$

which has solution $T(n) = c'n \log n$.

Question: How to find y?

Idea:

- running time good so long as S_1 and S_2 are *approximately* same size

  Example: if aim for partition such that $|S_1| \leq 3n/4$ and $|S_2| \leq 3n/4$, then

  - $T(n) \leq 2T(3n/4) + (c+1)n$ has solution $T(n) = O(n \log n)$
  - there are $n/2$ pivots y whose partitions are like this [[*To see this, imagine sorted array of elts and observe that middle half have this property.*]]
- choose a random pivot element y uniformly from S and hope to get lucky often enough

Analysis: (randomized alg):

Question: How many comparisons *in expectation*?

Def: For $1 \leq i \leq n$, let $S_{(i)}$ denote the element of *rank* i (the i'th smallest element) in S.

- $S_{(1)}$ is smallest elt of S
- $S_{(n)}$ is largest elt of S

Def: Let X_{ij} be indicator random variable that $S_{(i)}$ and $S_{(j)}$ are compared by alg.

- $X_{ij} = 1$ means $S_{(i)}$ and $S_{(j)}$ were compared, sorted directly
- $X_{ij} = 0$ means $S_{(i)}$ and $S_{(j)}$ were not compared, sorted implicitly

Then total number of comparisons is

$$\sum_{i=1}^{n} \sum_{j>i} X_{ij}$$

and expectation number is

$$E[\sum_{i=1}^{n} \sum_{j>i} X_{ij}] = \sum_{i=1}^{n} \sum_{j>i} E[X_{ij}].$$

Def: Let p_{ij} be probability $S_{(i)}$ and $S_{(j)}$ are compared by alg. Then

$$E[X_{ij}] = 1 \times p_{ij} + 0 \times (1 - p_{ij}) = p_{ij}.$$

Think of steps performed by alg. in a tree of pivots (DRAW TREE) and consider level-order traversal.

- there is a comparison between $S_{(i)}$ and $S_{(j)}$ iff $S_{(i)}$ or $S_{(j)}$ is chosen as pivot *before* any elt of rank between i and j

- consider permutation defined by order in which elts were chosen as pivots and note any elt in $\{S_{(i)}, \ldots, S_{(j)}\}$ equally likely to be first one chosen as pivot

Thus,

$$p_{ij} = \frac{2}{j - i + 1}.$$

[[*Is permutation defined above a uniformly random one? No, e.g., perm $S_{(3)}S_{(1)}S_{(2)}S_{(4)}S_{(5)}$ can't happen. Prob only uniform over 1st elt.*]]

To conclude,

$$\begin{aligned}
E &= \sum_{i=1}^{n} \sum_{j=i+1}^{n} p_{ij} \\
&= \sum_{i=1}^{n} \sum_{j=i+1}^{n} \frac{2}{j-i+1} \\
&= \sum_{i=1}^{n} \sum_{k=2}^{n-i+1} \frac{2}{k} \\
&\leq 2 \sum_{i=1}^{n} \sum_{k=1}^{n} \frac{1}{k} \\
&= O(n \log n)
\end{aligned}$$

Note:

- expected running time holds *for every input* [[*realization of running time depends only on random choices made by alg, not on input itself*]]

- we bounded expected running time, but can make stronger statement that running time is close to expectation *with very high probability* (for every input)

- choosing random pivot required $\log |S|$ random bits; sometimes not so easy
  - picking random real number from $[0, 1]$
  - simulating coin flip with bias $p \neq 2^{-k}$