

Algorithms and Complexity

Lecture 9

Randomized Algorithms (and more)

Randomized Algorithms

- Algorithm has access to a (hypothetical) subroutine outputting ***a random bit***.
- Performance measures of algorithm (running time, output quality) now are **random variables**:
 - We want guarantees on distributions.
 - Still interested in the worst-case instance!
- **Why?!**
 - Often simpler, faster, provably better algo's.
 - Error probability often negligible.

Example 1 (a)

- **Zero-Polynomial:** Given a polynomial $p(x)$ of degree d in formula form, is it the zero polynomial?

$(x^2 - x)4 + 2(x^3(3 + x + x^2))$ is a **no instance**

$(x^2 - x)4 + 4(x - x^2)$ is a **yes instance**

- How to solve **Zero-Polynomial** quickly?
 - Expanding parentheses may require exponential time!
- **Algo: 1)** Pick $x \in_R \{1, \dots, 2d\}$ uniformly at random;
2) Return YES iff $p(x) = 0$.
- **Thm:** Let $p(x)$ be a (univariate) polynomial of degree at most d over any field \mathbb{F} . Then p has at most d roots.
- If yes instance, $\Pr[Yes] = 1$; if not, $\Pr[Yes] \leq 1/2$

Example 1 (b)

- **Thm:** Let $p(x)$ be a (univariate) polynomial of degree at most d over any field \mathbb{F} . Then p has at most d roots.
- Alice has string $a \in \{0,1\}^n$, Bob has $b \in \{0,1\}^n$.
- Goal: verify $a = b$, with little communication.
- In deterministic setting, n bits needed!
- **Algo:**
 - 1) A field \mathbb{F} of size $\sim 2n$ is fixed (a priori)
 - 2) Alice picks $x \in_R \mathbb{F}$; sends x and $a(x) := \sum_{i=1}^n a_i x^i$
 - 3) Bob claims equality if $\sum_{i=1}^n b_i x^i - a(x) = 0$
- Alice and Bob use $O(\log n)$ bits of communication
- If equal, always correct. If not, $\Pr[\text{corr}] \geq \sim 1/2$

Example 1 (b)

- **Thm:** Let $p(x)$ be a (univariate) polynomial of degree at most d over any field \mathbb{F} . Then p has at most d roots.
- Alice has string $a \in \{0,1\}^n$, Bob has $b \in \{0,1\}^n$.
- Goal: verify $a = b$, with little communication.
- In deterministic setting, n bits needed!
- **Algo:**
 - 1) A field \mathbb{F} of size $\sim 2n$ is fixed (a priori)
 - 2) Alice picks $x \in_R \mathbb{F}$; sends x and $a(x) := \sum_{i=1}^n a_i x^i$
 - 3) Bob claims equality if $\sum_{i=1}^n b_i x^i - a(x) = 0$
- Alice and Bob use $O(\log n)$ bits of communication
- If equal, always correct. If not, $\Pr[\text{corr}] \geq \sim 1/2$

Example 1 (c)

- **Thm:** Let $p(x)$ be a (univariate) polynomial of degree at most d over any field \mathbb{F} . Then p has at most d roots.
- **Perfect matching:** Let $G = (A \cup B, E)$ be a bipartite graph. Does it have a perfect matching?
- Let $A = \{a_1, \dots, a_n\}$, $B = \{b_1, \dots, b_n\}$ and define

$$M[i, j] = \begin{cases} 0, & \text{if } \{a_i, b_j\} \notin E \\ x_{i,j}, & \text{if } \{a_i, b_j\} \in E \end{cases}$$

Recall $\det(M) = \sum_{\pi \in S_n} (-1)^{\text{sgn}(\pi)} \prod_{i=1}^n M[i, \pi(i)]$

- $\det(M)$ is a polynomial of degree $\leq n$ in the $x_{i,j}$'s which is non-zero iff G has perfect matching.
- **Algo:** 1) Fix a field \mathbb{F} of size $\sim 2n$, and pick $x_{i,j} \in_R \mathbb{F}$
2) Use a fast algorithm to compute determinant

Example 2

- Given an n -vertex graph, find a max. size IS.
- A greedy algorithm finds an IS of size $n/(d + 1)$, if d is the maximum degree
- A randomized algorithm finds an IS of expected size $\sum_{v \in V} 1/(d_v + 1)$:
 - Pick a random permutation of the vertices
 - Iterate over permuted sequences
 - Include vertex when no neighbor has been included yet

Example 3

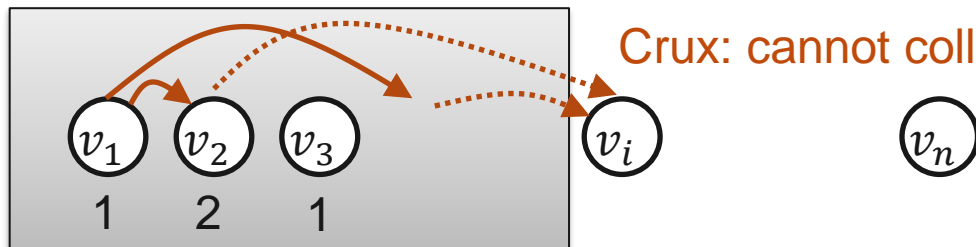
- Directed k -path problem: given directed graph G , is there a simple path on at least k vertices.
- In polynomial time if G is acyclic:

Algorithm $k\text{pathDAG}(G = (V, E), k)$

Assumes G is a directed acyclic graph

Output: Whether there exists a simple path on k vertices in G .

- 1: Find topological ordering v_1, \dots, v_n of G i.e. ordering such that for every $(v_i, v_j) \in E, i < j$
- 2: **for** $i = 1, \dots, n$ **do**
- 3: **if** $|N^-(v_i)| = 0$ **then**
- 4: Set $A[v_i] = 1$
- 5: **else**
- 6: Set $A[v_i] = \max_{u \in N^-(v_i)} A[u] + 1$
- 7: **return true** iff $\max_{v \in V} A[v] \geq k$



Example 3

- Idea: try to reduce the general case to acyclic case.

Algorithm `kpath1(G, k)`

G is directed

Output: Whether G has a simple path on k -vertices, with constant one-sided error probability.

- 1: **for** $i = 1 \dots k^k$ **do**
- 2: Pick for every $v \in V$ a number $c(v) \in \{1, \dots, k\}$ uniformly and independently at random
- 3: Let $G' = (V, E')$ where $E' = \{(u, v) \in E : c(v) = c(u) + 1\}$
- 4: **if** `kpathDAG(G', k)` **then return true**
- 5: **return false**

- If true is returned, G' has a k -path so G as well
- If G has a k -path, look at one iteration:

$$- \Pr_c[G' \text{ has } k\text{-path}] \geq \Pr_c[\forall i \in \{1, \dots, k\}: c(p_i) = i] = \left(\frac{1}{k}\right)^k$$

– So prob. we won't find a path in the whole loop is at most

$$\begin{array}{l} 1 + x \leq e^x \\ \text{for} \\ x = -1/k^k \end{array}$$

$$\left(1 - \frac{1}{k^k}\right)^{k^k} \leq 1/e, \text{ if } k\text{-path exists, return true with prob } 1 - \frac{1}{e}$$

- $O^*(k^k)$ time algorithm with constant one-sided err.

Example 4

- Determine whether a given k -CNF formula on n vars is satisfiable in $O^* \left(\left(\frac{2k}{k+1} \right)^n \right)$ time, for constant k .
 - for $k = 100$, this is about $O^*(1.9802^n)$
- relies on local search subroutine
- Given $x, y \in \{0,1\}^n$, $H(x, y)$ is *Hamming distance* (i.e., nr. of coordinates where x and y differ)

Example 4

Algorithm `localSearch(φ, x, d)` φ is a k -CNF-formula on n variables, $x \in \{0, 1\}^n$, $d \in \mathbb{N}_{\geq 0}$.

Output: Whether there exists $y \in \{0, 1\}^n$ that satisfies φ and $H(x, y) \leq d$

- 1: **if** $d = 0$ and $\varphi(x) = \text{false}$ **then return false**
- 2: **if** \exists clause C_j not satisfied by x **then**
- 3: For $i = 1, \dots, \ell = |C_j|$ let $z^i \in \{0, 1\}^n$ be the assignment obtained from x by flipping the
- 4: variable in the i 'th literal of C_j
- 5: **return** $\bigvee_{i=1}^{\ell} \text{localSearch}(\varphi, z^i, d - 1)$
- 6: **else**
- 7: **return true**

- Run time: recursion tree of depth $\leq d$, $\ell \leq k$ fanout
 - Polynomial time per recursive call so $O^*(k^d)$ time.
- Only considers assignments of HD $\leq d$:
 - decreases d if it flips a coordinate
 - if true is returned, found a solution of HD $\leq d$
- By induction on d : if sol y of HD $\leq d$, it returns true
 - True for $d = 0$, since condition on L2 fails
 - For $d > 0$, x, y must differ on variable occurring in C_j
 - If variable occurs in i 'th literal $H(z^i, y) \leq d - 1$

Example 4

Algorithm $\text{kSAT}(\varphi)$

φ is a k -CNF-formula on n variables

Output: Whether $x \in \{0, 1\}^n$ is satisfiable, with constant one-sided error probability.

1: $d = n/(k + 1)$

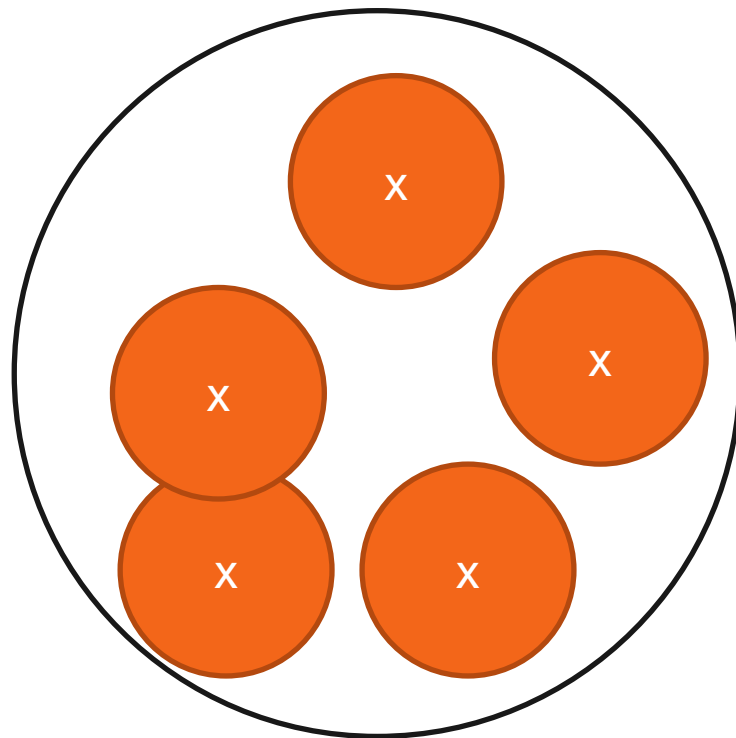
Assume d is integer (otherwise, add at most k dummy variables)

2: **for** $i = 1 \dots \lceil 2^n / \binom{n}{d} \rceil$ **do**

3: Pick $x \in \{0, 1\}^n$ uniformly at random

4: **if** $\text{localSearch}(\varphi, x, d)$ **then return true**

5: **return false**



Example 4

Algorithm $kSAT(\varphi)$

φ is a k -CNF-formula on n variables

Output: Whether $x \in \{0, 1\}^n$ is satisfiable, with constant one-sided error probability.

- 1: $d = n/(k + 1)$ Assume d is integer (otherwise, add at most k dummy variables)
- 2: **for** $i = 1 \dots \lceil 2^n / \binom{n}{d} \rceil$ **do**
- 3: Pick $x \in \{0, 1\}^n$ uniformly at random
- 4: **if** $localSearch(\varphi, x, d)$ **then return true**
- 5: **return false**

- If returns true it is correct since so is **localSearch**
- If sol y exists, prob true is returned in one iteration is

$$\Pr_x[H(x, y) \leq d] = \frac{\sum_{i=0}^d \binom{n}{i}}{2^n} \geq \frac{\binom{n}{d}}{2^n}, \text{ so in some iteration } \geq 1 - \frac{1}{e}$$

$$\binom{n}{d} = \frac{n!}{d!(n-d)!} \geq \frac{(n/e)^n}{3n(d/e)^d 3n((n-d)/e)^{n-d}} = \frac{n^n}{d^d (n-d)^{n-d} 9n^2}$$

- And the running time boils down to:

$$k^d \frac{2^n}{\binom{n}{d}} = \frac{2^n (kd)^d (n-d)^{n-d} 9n^2}{n^n} = \frac{2^n \left(n \frac{k}{k+1}\right)^d \left(n \frac{k}{k+1}\right)^{n-d} 9n^2}{n^n} = O^* \left(\left(\frac{2k}{k+1} \right)^n \right)$$

Conditional expectations and derandomization

- Consider the problem MAX-3-SAT: given a 3-CNF formula ϕ , find an assignment satisfying the maximum number of clauses
- A random assignment will in expectation satisfy $7/8$ clauses

Conditional expectations and derandomization

- Consider the problem MAX-3-SAT: given a 3-CNF formula ϕ , find an assignment satisfying the maximum number of clauses
- A random assignment will in expectation satisfy $7/8$ clauses
- Can deterministically find such a solution, by computing
 - $E[\text{\#of clauses satisfied} \mid x_1 = \textit{true}]$
 - $E[\text{\#of clauses satisfied} \mid x_1 = \textit{false}]$

More on randomized algo's

- Randomized rounding
 - Important approach to turn LP-solutions into ILP solutions for obtaining good apx algo's
- Many more cool examples

- LNMB course `Randomized Algorithms`.

Algorithms & Complexity

- We saw:
 - NP, co-NP, NP-completeness, undecidability
 - Approximation algorithms
 - Exponential time (and FPT) algorithms
 - Treewidth
 - Randomized algorithms
- What did we skip?

Algorithms

- Many `basic' algorithms for classic problems on
 - Sequences: Sorting, Pattern Matching,...
 - Graphs: Shortest Path, Matching, Max-Flow, ..
 - Algebra: Matrix multiplication, FFT, number-theoretic
 - Computational Geometry: Convex hull, closest pair
- We've seen many of the important techniques
 - Greedy, DP, Divide&Conquer, LP-rounding, randomization
- Online, quantum, parallel, game-theoretic algorithms
- **Important one we skipped:** Heuristic algorithms

Complexity

- Models of Computation
 - Turing machine, circuits, branching programs...
- Diagonalization
- Polynomial hierarchy, NP-intermediate problems
- Proof systems
 - PSPACE = IP
 - PCP-theorem
- Derandomization
 - L=NL, Adleman's theorem
- Communication Complexity
- 'Fine-grained complexity'
 - (S)ETH, W-hardness, conditional hardness results

Further reading

