上海科技大学
ShanghaiTech University

CS240 Algorithm Design and Analysis

Lecture 20

Randomized Algorithms

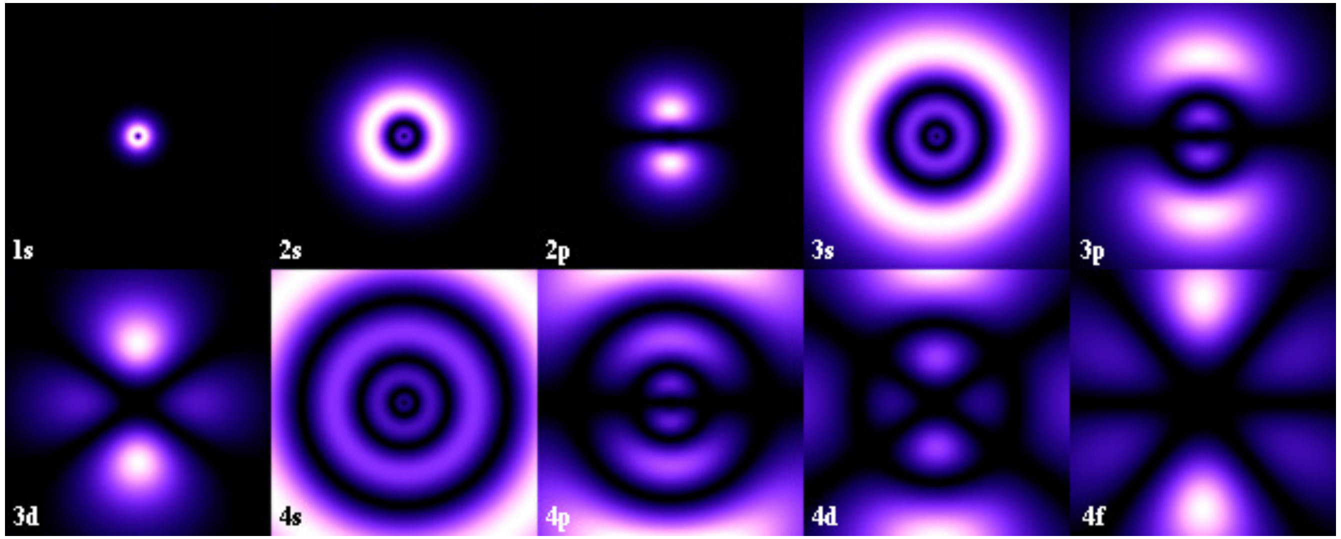
Quan Li
Fall 2024
2024.12.05

Randomized Algorithms

- Till now, all of our algorithms have been deterministic.
  - Given an input, the algorithm always does the same thing.
- It turns out it's very useful to allow algorithms to be nondeterministic.
  - As the algorithm operates, it's allowed to make some random choices.
  - Running the algorithm multiple times on same input can produce different behaviors.
- Randomization. Allow fair coin flip in unit time.

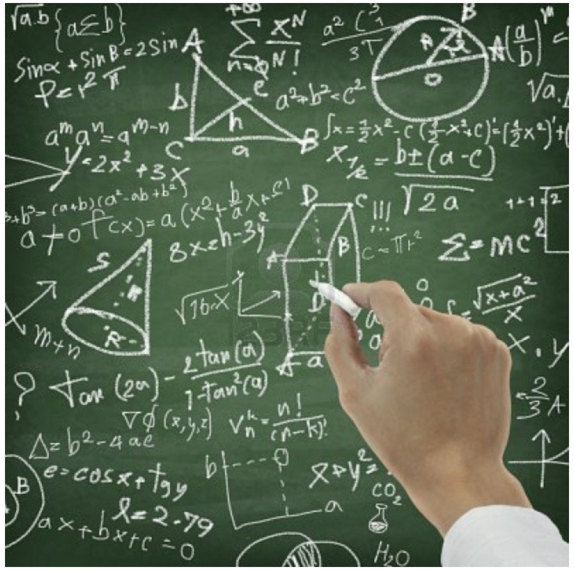
Why Randomized Algorithms?

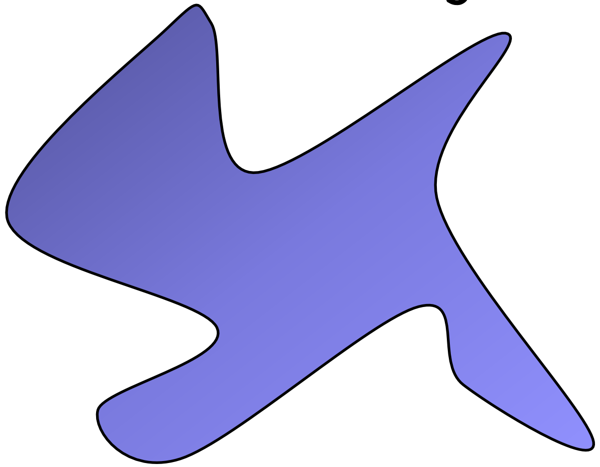
- For many problems, randomized algorithms work better than deterministic ones.
  - Faster / uses less memory
  - Simpler, easier to understand.
  - Some problems that provably can't be solved (or solved efficiently) by deterministic algorithms can be solved by randomized ones.
  - According to quantum mechanics, the world is inherently probabilistic, so nature is randomized!



How can randomness help?

- Say you have a string of length n that's half A's and half B's.
- We want to find a location in the string with an A.
- Any deterministic algorithm takes n/2+1 steps in the worst case.
- But by checking random locations, a randomized algorithm finds an A in 2 steps in expectation.
- Measure the area of this
- Method 1





- Method 2
  - Print the shape out on a piece of paper.
  - Throw 100 darts at it.
  - See what percent land in the shape.
  - Multiply by area of your paper.

Las Vegas VS. Monte Carlo

- A Las Vegas randomized algorithm always produces the right answer. But it's running time can vary depending on its random choices.
  - We want to minimize the expected running time of a Las Vegas algorithm.
- A Monte Carlo algorithm always has the same running time. But it sometimes produces the wrong answer, depending on its random choices.
  - We want to minimize the error probability of a Monte Carlo algorithm.

Probability Review

- Discrete probability theory is based on events and their probabilities.
  - Events can be composed of more basic events.
  - Ex Event of rolling a 2 on a fair dice, with probability 1/6.
  - Ex Event of rolling an even number, with probability ½. Composed of basic events of rolling a 2, 4 or 6.
  - If A is event, write Pr[A]=y. E.g., Pr[roll a 2]=1/6
- Two events A, B are independent if $Pr[A \wedge B]=Pr[A]*Pr[B]$.
  - Ex Events A="2 on first roll" and B="3 on second roll" are independent, because $Pr[A \wedge B]=1/36=Pr[A]*Pr[B]=1/6*1/6$.
  - Ex Events A="2 on first roll" and B="the two rolls sum to 5" are not independent, because $Pr[A \wedge B]=1/36 \neq Pr[A]*Pr[B]=1/6*4/36$.

Probability Review

- Random variables
  - A variable which takes values with certain probabilities. The probabilities sum to 1.
  - Ex X = value from roll of dice. Values are {1,2,3,4,5,6}, each with probability 1/6.
  - Ex Y = number of heads in 4 flips of fair coin. Values are {0,1,2,3,4}, with probabilities {1/16,4/16,6/16,4/16,1/16}.
  - Ex Z = number of flips of fair coin till first head. Values are {1,2,3,...}, with probabilities {1/2,1/4,1/8,...}.
  - We write Pr[X=x]=y, e.g. Pr[Z=3]=1/8.

Probability Review

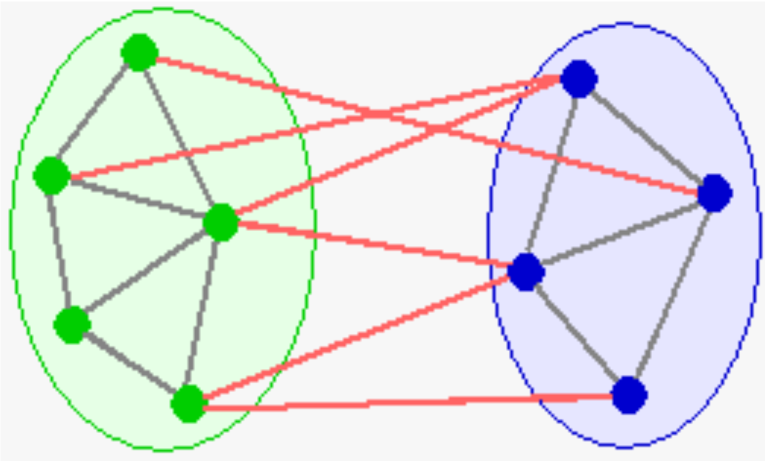
- Expectation of random variable X
  - $E[X]=\sum_x X * Pr[X=x]$.
  - The average value of X, over many trials.
  - Ex X=number of heads in 4 flips. E[X]=0*1/16+1*4/16+2*6/16+3*4/16+4*1/16=2.
    - If you flip a coin 4 times, for 1000 times, on average you see 2 heads per 4 flips.
- An indicator variable X for an event E is a random variable that's 1 of E occurs, and 0 otherwise.
- If event E has probability p of occurring, and X is E's indicator variable, then E[X]=p.
  - Because E[X]=Pr[E occurs]*1+Pr[E doesn't occur]*0=p.
  - This is a convenient fact we'll frequently use.

Probability Review

- Linearity of expectations
  - Given random variables X, Y, E[X+Y]=E[X]+E[Y].
  - Extends to any number of random variables, e.g. E[X+Y+Z]=E[X]+E[Y]+E[Z].
  - The random variables do not have to be independent.
  - Very useful property!
  - Ex Let X=number of heads in 100 coin flips. Calculate E[X].
    - Direct method: 1*Pr[1 head]+2*Pr[2 heads]+…+100*Pr[100 heads], a very complicated sum.
    - Linearity method: $X=X_1+X_2+\ldots+X_{100}$, where X_i=number of heads on i'th flip.
    - $E[X_i]=0*Pr[0 \text{ heads}]+1*Pr[1 \text{ head}]=1/2$.
    - $E[X]=E[X_1]+\ldots+E[X_{100}]=100/2=50$.

Max Cut

- We studied the Min-Cut problem, which is closely related to finding the max flow in a network.
- Max-Cut is the opposite of Min-Cut.
- Given a graph G, split vertices into two sides to maximize the number of edges between the sides.



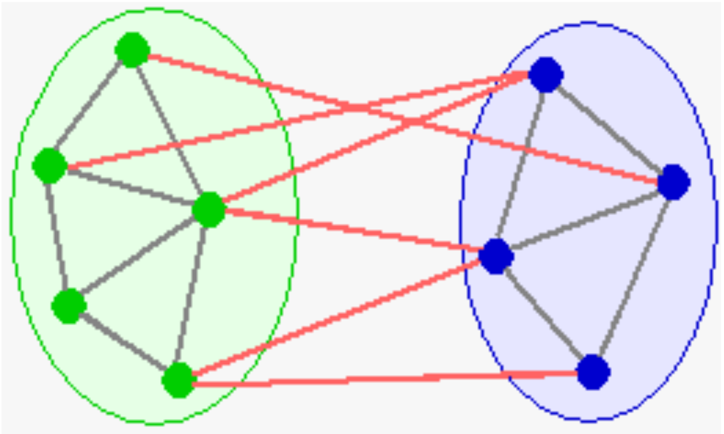
Max Cut

- Unlike Min-Cut, Max-Cut is NP-complete.
- We'll give a very simple randomized Monte Carlo 2-approximation algorithm.
  - Monte Carlo means the algorithm sometimes returns the wrong answer, i.e., a cut that's not a 2-approximation.
  - Monte Carlo also means the algorithm always runs in a fixed amount of time.

    - Put each node in a random side with probability ½.

Correctness

- Lemma In a graph with e edges, the algorithm produces a cut with expected size e/2.
- Proof Let X be a random variable equal to the size of the cut. We want to bound E[X].
  - For each edge e, let X_e be the indicator variable of whether e is in the cut.
    - i.e., X_e=1 if e is in the cut and 0 otherwise.
  - X= $\sum_e X_e$.
  - Given an edge e=(i,j), e is in the cut if i and j are on different sides.
  - Pr[e in cut]=Pr[(i in L) ∧ (j in R)] + Pr[(j in L) ∧ (i in R)]=1/4+1/4=1/2.
  - E[X_e]=1/2.
  - E[X]=e/2 by linearity of expectations.



Correctness

- Since a cut can have at most e edges, the e/2 edges the algorithm outputs in expectation is a 2 approximation.
- Note that we only bounded expected size of the algorithm's cut.
  - In any particular execution, the algorithm can output a cut that's smaller or larger than e/2.
    - On average, the cut has size e/2.

Contention Resolution

An example in distributed computing where randomization is necessary

Distributed Computing

- Distributed system
  - Set of autonomous nodes, working independently of each other
  - Nodes may be able to communicate, at a cost
  - Ex: Internet, computer cluster, sensor network
- Nodes need to coordinate to solve some problem
- Coordination can be done using communication. But communication is expensive
- By making nodes randomized, they can coordinate with minimal communication
- Randomization also simplifies symmetry breaking between nodes

Contention Resolution in a Distributed System

上海科技大学 ShanghaiTech University

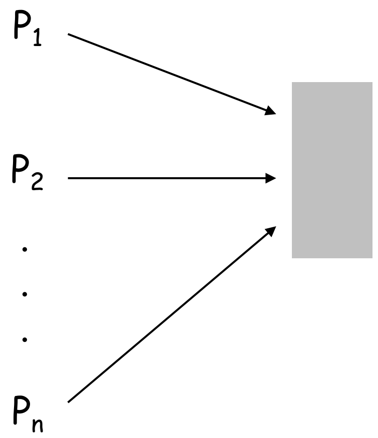
Contention resolution. Given n processes P_1, …, P_n, each competing for access to a shared channel. If two or more processes access the channel simultaneously, all processes are locked out. Devise protocol to ensure all processes get through on a regular basis.

Assumption. Time is divided into rounds.

Restriction. Processes can't communicate, and they don't have id's.

Challenge. Need symmetry-breaking paradigm.

No deterministic protocol can solve the problem.



立志成才报国裕民

Contention Resolution: Randomized Protocol

Protocol. Each process requests access to the channel at time t with probability $p = 1/n$.

Claim. Let S[i, t] = event that process i succeeds in accessing the database at time t. Then $1/(e \cdot n) \le \Pr[S(i, t)] \le 1/(2n)$.

Pf. By independence, $\Pr[S(i, t)] = p\,(1-p)^{n-1}$.

process i requests access

none of remaining n-1 processes request access

- Setting p = 1/n, we have $\Pr[S(i, t)] = 1/n\ \underbrace{(1 - 1/n)^{n-1}}_{\text{between } 1/e \text{ and } 1/2}$. ▪

Useful facts from calculus.

- $1/4 < (1 - 1/n)^n < 1/e < (1 - 1/n)^{n-1} < 1/2$
- $9/4 < (1 + 1/n)^n < e < (1 + 1/n)^{n+1} < 27/8$

Contention Resolution: Randomized Protocol

Claim. The probability that process i fails to access the channel in e·n rounds is at most 1/e. After e·n·c ln n rounds, the probability is at most n^{-c}.

Pf. Let F[i, t] = event that process i fails to access database in rounds 1 through t. By independence and previous claim, we have
Pr[F(i, t)] $\leq$ $(1 - 1/(en))^t$.

- Choose t = e n: $$\Pr[F(i,t)] \leq \left(1 - \tfrac{1}{en}\right)^{en} \leq \tfrac{1}{e}$$
- Choose t = (e n) (c ln n): $$\Pr[F(i,t)] \leq \left(\tfrac{1}{e}\right)^{c \ln n} = n^{-c}$$

Contention Resolution: Randomized Protocol

Claim. The probability that all processes succeed within $2e \cdot n \ln n$ rounds is at least $1 - 1/n$.

Pf. Let $F[t]$ = event that at least one of the n processes fails to access database in any of the rounds 1 through t.

$$\Pr\big[F[t]\big] = \Pr\left[\bigcup_{i=1}^{n} F[i,t]\right] \le \sum_{i=1}^{n} \Pr[F[i,t]] \le n \cdot n^{-c}$$

union bound previous slide

- Choosing $t = 2 e n \ln n$ yields $\Pr[F[t]] \le n \cdot n^{-2} = 1/n$. ▪

Union bound. Given events $E_1, \ldots, E_n$, independent or not,

$$\Pr\left[\bigcup_{i=1}^{n} E_i\right] \le \sum_{i=1}^{n} \Pr[E_i]$$

Global Minimum Cut

A problem for which the best-known randomized algorithm is faster than the best-known deterministic algorithm

Global Minimum Cut

Global min cut. Given a connected, undirected graph G = (V, E), find a cut (A, B) of minimum cardinality.

Applications. Partitioning items in a database, identify clusters of related documents, network reliability, network design, circuit design, TSP solvers.

Network flow solution.

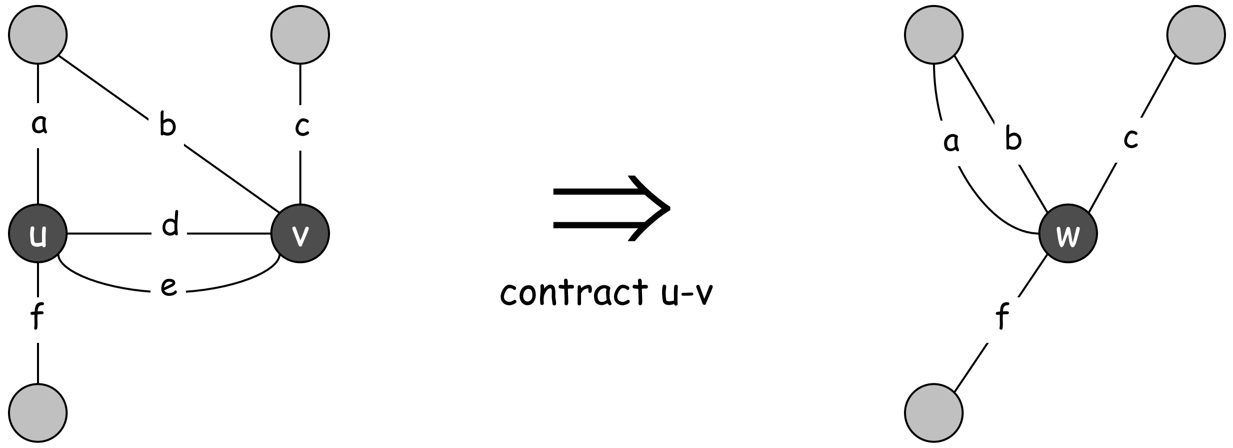
- Replace every edge (u, v) with two antiparallel edges (u, v) and (v, u).
- Pick some vertex s and compute min s-v cut separating s from each other vertex $v \in V$.

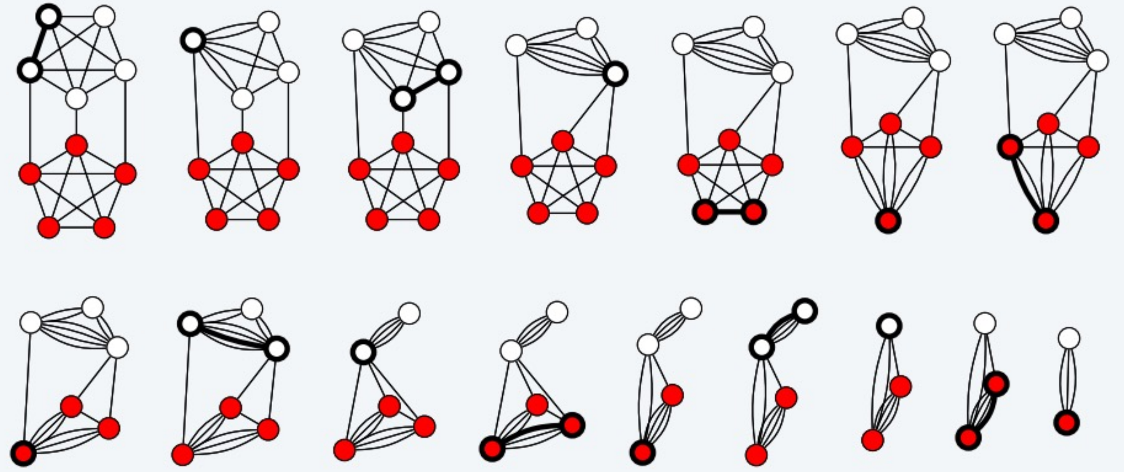
False intuition. Global min-cut is harder than min s-t cut.

Contraction Algorithm

Contraction algorithm. [Karger 1995]

- Pick an edge e = (u, v) uniformly at random.
- Contract edge e.
  - replace u and v by a single new supernode w
  - preserve edges, updating endpoints of u and v to w
  - keep parallel edges, but delete self-loops
- Repeat until graph has just two supernodes v_1 and v_2.
- Return the cut (between the two supernodes).





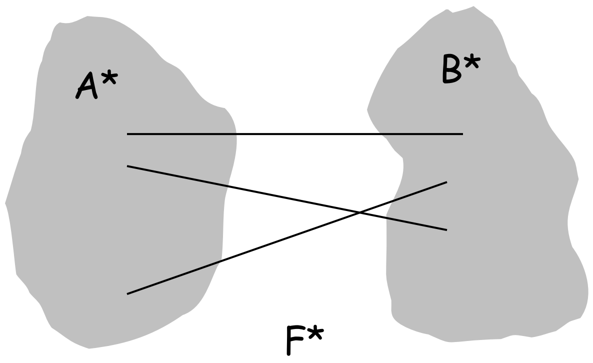
Reference: Thore Husfeldt

Contraction Algorithm

Claim. The contraction algorithm returns a min cut with prob $\geq 2/n^2$.

Pf. Consider a global min-cut (A^*, B^*) of G. Let F^* be edges with one endpoint in A^* and the other in B^*. Let $k = |F^*|$ = size of min cut.

- In first step, algorithm contracts an edge in F^* with prob $k / |E|$.
- Every node has degree $\geq k$ since otherwise (A^*, B^*) would not be min-cut. $\Rightarrow$ $|E| \geq \frac{1}{2}kn$.
- Thus, algorithm contracts an edge in F^* with probability $\leq 2/n$.



Contraction Algorithm

Claim. The contraction algorithm returns a min cut with prob $\geq 2/n^2$.

Pf. Consider a global min-cut (A*, B*) of G. Let F* be edges with one endpoint in A* and the other in B*. Let k = |F*| = size of min cut.

- Let G' be graph after j iterations. There are n' = n-j supernodes.
- Suppose no edge in F* has been contracted. The min-cut in G' is still k.
- Since value of min-cut is k, $|E'| \geq \frac{1}{2}kn'$ -> $k/|E'|$ <= $2/n'$
- Thus, algorithm contracts an edge in F* with probability $\leq 2/n'$.

- Let E_j = event that an edge in F* is not contracted in iteration j.

$$
\begin{aligned}
\Pr[E_1 \cap E_2 \cdots \cap E_{n-2}] &= \Pr[E_1] \times \Pr[E_2 \mid E_1] \times \cdots \times \Pr[E_{n-2} \mid E_1 \cap E_2 \cdots \cap E_{n-3}] \\
&\geq \left(1-\tfrac{2}{n}\right)\left(1-\tfrac{2}{n-1}\right)\cdots\left(1-\tfrac{2}{4}\right)\left(1-\tfrac{2}{3}\right) \\
&= \left(\tfrac{n-2}{n}\right)\left(\tfrac{n-3}{n-1}\right)\cdots\left(\tfrac{2}{4}\right)\left(\tfrac{1}{3}\right) \\
&= \frac{2}{n(n-1)} \\
&\geq \frac{2}{n^2}
\end{aligned}
$$

Contraction Algorithm

Amplification. To amplify the probability of success, run the contraction algorithm many times.

Claim. If we repeat the contraction algorithm n^2 times with independent random choices and return the best cut found, then the algorithm finds the min-cut with constant probability.

Pf. By independence, the probability of failure is at most

$$\left(1-\frac{2}{n^2}\right)^{n^2} = \left[\left(1-\frac{2}{n^2}\right)^{\frac{1}{2}n^2}\right]^2 \le \left(e^{-1}\right)^2 = \frac{1}{e^2}$$

Global Min Cut: Context

Remark. Overall running time is slow since we perform $O(V^2)$ iterations, and each takes $O(E \log V)$ time (we always merge the vertex with smaller degree into the other).

Best known. [Karger 2000] $O(E \log^3 V)$.

faster than best known deterministic global min cut algorithm

立志成才 报国裕民