Randomized Algos

Fall 2025

Anupam Gupta anupam.g@nyu.edu
WWH 430.

Thursdays.

- Webpage
- Homeworks
- Topics
-

Why randomized algos?

- Amazing algorithmic resource. "superpower"
- simpler / better / faster algorithms
- elegant algos / analysis
- for some problems, we know randomized algos, but no deterministic algos (yet). In some models of computation, randomness gives extra power (e.g. streaming, online algos, distributed algos).
- working without randomness is like tying one hand behind back.

  Probably still OK, but why do it?

  Sometimes good as a challenge.... ☺

Full disclosure:

Some reasons to be cautious –

- getting true random bits can be challenging, so may want to remove dependence on randomness.
  - + though work on using weak random sources to amplify randomness (assuming complexity assumptions
  - + often times enough to work with limited randomness.
  - + sometimes: generic "derandomization" theorems.
- asking whether randomness is really needed is a fundamental question in the theory of computational complexity.

  is $P = RP$? $P = BPP$?

  whatever these are ... (randomized complexity classes)

Our goal is to understand!

Several Algorithmic Reasons to use Randomness

- Random Sampling

  e.g. Know many good solutions in some set but don't know how to find one (don't understand structure).

- Random Walk

  Set up a process that ~~gets us~~ lets us explore the structure of the sample space quickly (either explore all of it, or sample uniformly from it)
  (Markov Chain Monte Carlo)

- Sparsification

  A random subset preserves the structure of the object we're looking for. Then can focus on that in this smaller subset.

- Load Balancing / Symmetry Breaking

  Throwing balls into ~~m~~ bins should give us (approx) equal load on each bin. No bin has too many or too few.

- Making Algorithm Unpredictable (for Adversary)

  In online / streaming models, adversary can destroy us if it knows the algorithm's actions. Adding randomness (with "oblivious" adversary) can give us more power.

- Powerful Language of Probability theory

  E.g. Arguing about high dimensional geometry is difficult / cumbersome but often probabilistic arguments / methods give convenient tools

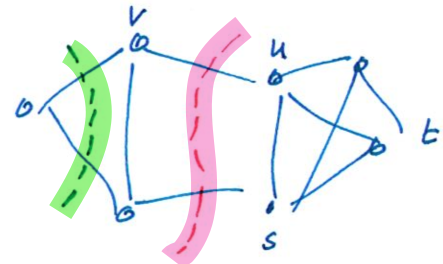
- and more...

- not to mention: Existence Proofs via $\boxed{\text{the Probabilistic Method}}$!!

Randomized Minimum Cut Algorithm

Given undirected graph $G=(V,E)$ say unweighted for now.

find the (global) minimum cut.



- Partition of vertices with fewest edges crossing it.

In the example: both red and green cuts are min-cuts.

"boundary of S" / "cut of S"

- Formally: ~~let~~ given $S \subseteq V$, let ∂S be edges with one end in S one end not in S.

want $S \subseteq V$ st $S \neq \emptyset$, $S \neq V$

that minimizes $|\partial S|$.

Strawman Algos:

- enumerate over all cuts. (duh!)
- since we can find minimum s-t cuts (using maximum flow techniques)

maybe larger than mincut

[~~these are cuts/ ... S containing s ∈ V and S ∌ t~~

Given $s,t \in V$ called "source" and "sink"

find ~~the ...~~ $S \subseteq V$ with $S \ni s$ and $S \not\ni t$ and min $|\partial S|$]

can use this to find (global) mincut in time

- $O(n^2) \times$ time to find s-t min cut

~~...~~ try all possible s,t pairs

- $O(n) \times$ (s-t min cut runtime)

if $|\partial S|$ is mincut then $|\partial(V \setminus S)|$ also.

so can assume mincut set S has vertex 1 (any fixed vtx) in it.

Now try all others, in $(n-1)$ tries find mincut.

Here's a lovely algorithm [Karger 199]

(Assume G connected, else problem is trivial)

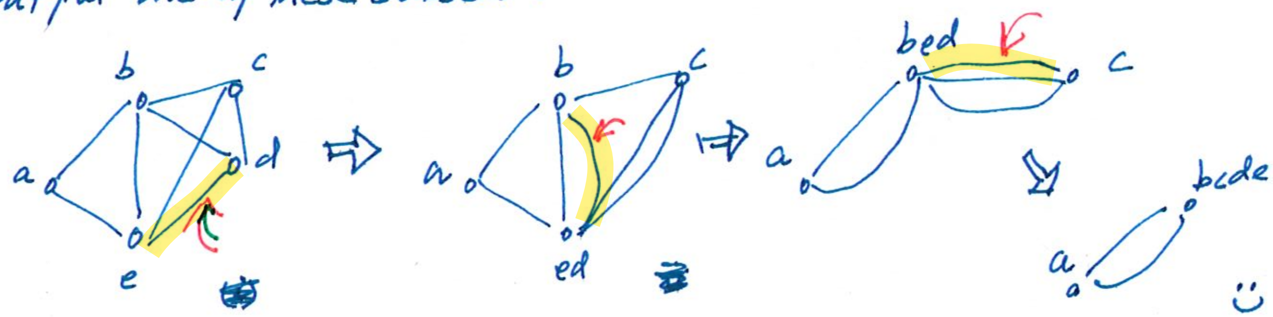
> While G has more than 2 vertices:
>
> - Pick a uniformly random edge of G
> - Contract it
>   - delete self loops
>   - keep parallel edges

"Contraction Algorithm"

When process ends (after $n-2$ rounds), graph has 2 nodes, each corresponds to some subset of nodes contracted into it.

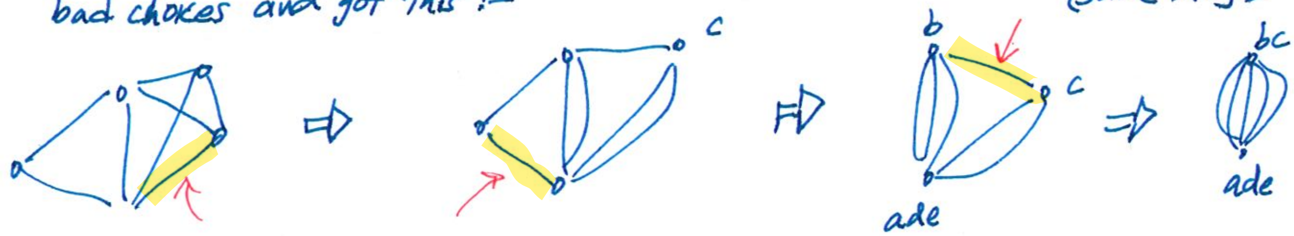
Output one of these subsets.

E.g.:



output $S = a$
or $S = bcde$
(same thing)

of course we got lucky: we may have picked bad choices and got this:-



in which case $|\partial\{bc\}| = 5$ ☹

Q: What is the probability that the algorithm outputs ~~the~~ a min-cut?

could be many mincuts!

Claim 1: Fix a mincut ∂S with λ edges, say

~~Suppose we have not:~~ Probability ~~that~~ [we contract edge in ∂S] $\leq 2/n$ in first step.

Pf: min cut ∂S has λ edges.

Note: any vertex v is a singleton cut $\partial\{v\}$, with degree(v) edges. must be no better than min cut.

$$\Rightarrow \text{degree}(v) \geq \lambda \quad \forall v. \quad \Rightarrow \underbrace{\sum_i \text{degree}(v)}_{=\,2m} \geq n\lambda$$

$2m$ (handshake lemma)

m ← # of edges.

$$\Rightarrow \Pr[\text{contract edge in } \partial S] = \frac{|\partial S|}{m} \leq \frac{\lambda}{(n\lambda/2)} = \frac{2}{n}.$$

$\blacksquare$

$$\Rightarrow \Pr[\text{mincut } \partial S \text{ survives at first step}] \geq 1 - 2/n = \frac{n-2}{n}$$

Important: min cut in G is also min cut in contracted graph!

if it survives

$$\Rightarrow \Pr[\text{mincut } \partial S \text{ survives 1st step, 2nd step, 3rd} \ldots, n-2^{th} \text{ step}]$$

$$\geq \frac{\cancel{n-2}}{n} \cdot \frac{\cancel{n-3}}{n-1} \cdot \frac{n-4}{\cancel{n-2}} \cdots \cdots \frac{2}{\cancel{4}} \cdot \frac{1}{\cancel{3}} = \frac{2}{n(n-1)} = \frac{1}{\binom{n}{2}}$$

Thm 2: In a single run of Contraction algo, $\Pr[\text{min cut } \partial S \text{ survives}] \geq 1/\binom{n}{2}$.

Good news: min-cut survives with (small) probability

Bad news: probability is small. ☹

What to do? Repeat the experiment!

Lemma 3: Suppose we succeed in some experiment with probability p. Then succeed in at least one of $\frac{1}{p} \cdot \log \frac{1}{\delta}$ runs with probability $1-\delta$.

Proof: $\Pr(\text{fail in all } T \text{ rounds}) = (1-p)^T \le e^{-pT}$

so if $T = \frac{1}{p} \ln \frac{1}{\delta}$

$\Rightarrow e^{-pT} = e^{-\ln 1/\delta} = e^{\ln \delta} = \delta.$

$\Rightarrow \Pr(\text{succeed in at least one round}) \ge 1-\delta.$ $\blacksquare$

Boosted Contraction Algorithm

Repeat $T = \binom{n}{2} \ln(n^{10})$ times

run Contraction Algo.

Return smallest cut found in those T runs

$1 - 1/\text{poly}(n)$

("with high probability")

Thm 4: Boosted CA returns the mincut w.p. $\ge 1 - 1/n^{10}$ (or whp)

"with probability"

— × —

Runtime:

Naive implementation of C.A. takes $O(m) = O(n^2)$ time

#edges (# vertices)^2

$\Rightarrow$ repeating $O(n^2 \log n)$ time gives $O(n^4 \log n)$.

Not amazing. ☹

Can do better using a nice idea, let me sketch it here.

Recursive Contraction Algo

Note: Pr(success of C.A.) $\geq \underbrace{\frac{n-2}{n} \cdot \frac{n-3}{n-1} \cdot \ldots}_{\text{good chance}} \frac{k}{k+2} \cdot \frac{k-1}{k+1} \underbrace{\cdots \frac{2}{4} \cdot \frac{1}{3}}_{\text{low prob.}}$

What if we stop at some $k = c.n$ where $c \in [0,1]$ constant.

$\Rightarrow$ Pr(min cut survives until this point) $\geq \frac{n-2}{n} \cdots \frac{k-1}{k+1} = \frac{k(k-1)}{n(n-1)} \approx \frac{k^2}{n^2}$

$= c^2$

So get from graph $G = G_n$ to G_k

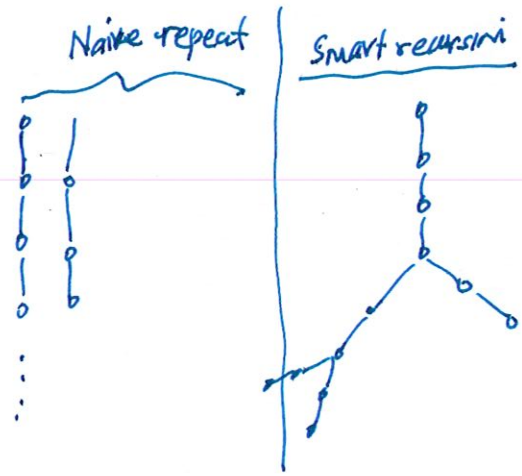
① succeed in maintaining min cut w.p. c^2

② G_k is smaller, of size cn.

Repeating on this costs less!

Smart recursion!

Recursive Contraction Algo



Contract until get to $k = cn$ vertices.

Let this contracted graph be G_k.

Repeat RCA twice on G_k.

Return the best.

[Karger-Stein]

Thm 5: Setting $c = 1/\sqrt{2}$ gives the following guarantee

(a) Pr[min cut returned] $\geq \frac{1}{\log n}$

(b) Runtime $= O(n^2 \log n)$.

Now can do naive repeats if desired.

Not proving in class / lecture. Plenty of proofs online.

Aside: The Probabilistic Method.

We picked a min cut ∂S.

Showed that Contraction Algo returns ∂S wp $\geq 1/\binom{n}{2}$.

Now suppose there is another <u>min cut</u> ∂T

C.A. will return ∂T w.p. $1/\binom{n}{2}$.

And these are <u>disjoint</u> events ($\partial S \neq \partial T$). !

<u>Thm</u>: Any graph G (connected, undirected) has $\leq \binom{n}{2}$ minimum cuts.

<u>Pf</u>: Each min cut is returned by C.A. wp $\geq 1/\binom{n}{2}$.

But these are disjoint events.

So $(\#\text{min cuts}) \cdot 1/\binom{n}{2} \leq 1$ (total probability)

$\Rightarrow \#\text{min cuts} \leq \binom{n}{2}$. $\blacksquare$

<u>Fact</u>: Tight.

Any 2 edges gives a min cut

$\Rightarrow \binom{n}{2}$ mincuts.

Theorem is fact about graphs.

No probability, no algorithms, no nothing.

But proof uses a randomized argument, a randomized algorithm.

~~Called~~ One example of $\boxed{\text{The Probabilistic Method}}$

more to come...