Lecture 16

- **Last time:**
  - $\#P = \{ f : \Sigma^* \to \mathbb{N} \mid \exists$ poly-time NDTM s.t. $\forall x : f(x) =$ #accepting computations on input $x \}$

    e.g. #SAT, #MATCHINGS, etc.
  - f.p.r.a.s for function f:
    - randomized algorithm, which on input x, outputs Z s.t.

      w.pr. $\geq \frac{3}{4}$: $\frac{f(x)}{1+\varepsilon} \leq Z \leq f(x) \cdot (1+\varepsilon)$. (*)
    - running time $\text{poly}(|x|, \frac{1}{\varepsilon})$
    - can boost $3/4$ to $1-\delta$, by running algorithm $O(\log \frac{1}{\delta})$ times and outputting the median of the results.
  - shouldn't hope to get f.p.r.a.s. for problems in #P whose underlying decision problem is NP-complete, as we can always tell the difference between 0 or ≥ 1 # of solutions from the estimate output from the algorithm if that satisfies (*) (which it does w pr. $\geq 3/4$)

- **Today:**
  - i> Random sampling is equivalent to appx counting
  - ii> all or nothing: a counting problem either has an f.p.r.a.s. or has no poly-time randomized approximation to within any polynomial factor.

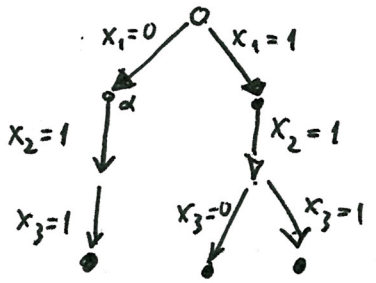
  → for "self-reducible problems"

def: An NP search problem is self-reducible if the set of solutions can be partitioned efficiently into a polynomial number of sets, each of which is in 1-1 correspondence w/ the set of solutions of a smaller instance.

- The polynomial-size set of smaller instances should be efficiently computable.

- e.g. SAT

$$\varphi = (\neg x_1 \vee x_2) \wedge (x_1 \vee x_2) \wedge (x_1 \vee x_3)$$



leaves : satisfying assignments

internal nodes : correspond to smaller formulas

e.g. node α corresponds to formula: $\varphi_\alpha = x_2 \wedge x_3$

- can represent instance x of f self-reducible problem by a tree $T(x)$:

  root = original instance x

  leaves = solutions (# leaves = $f(x)$)

  internal nodes = smaller instances

  branching = corresponds to the partition in the definition of self-reducible problems

  (degree is poly in $|x|$)

  depth : poly($|x|$) as the instance becomes progressively smaller.

Theorem 1: For all self-reducible NP problems, there exists a f.p.r.a.s. for counting iff there exists a polynomial time algorithm for almost uniform sampling.

Proof: ($\Leftarrow$ sampling to counting)

- simplifying assumptions: i) can sample exactly uniformly

  (the error coming from appx sampling can be easily absorbed into the error of the f.p.r.a.s; we skip this analysis here for clarity)

ii> the tree has branching 2

- Idea: work w/ self-reducibility tree

- First stage of the algorithm: • use sampling algorithm to sample t leaves u.a.r.

• using these samples get an estimate $\hat{r}$ of the ratio:

$$r = \frac{\#\text{leaves in left subtree}}{\text{total number of leaves}}$$

(assume w.l.o.g. that the left subtree has $\geq \frac{1}{4}$ of leaves)

[can always pick confidently a side w/ this property by looking at the samples]

• now left subtree corresponds to a smaller instance w/ a total # of solutions N_1; recursively produce an estimate $\hat{N}_1$ of N_1 (if left subtree is just a leaf then output 1)

• output $\frac{\hat{N}_1}{\hat{r}}$ as an estimate of the total #leaves

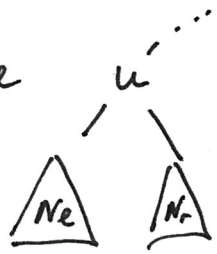
- if m is depth of tree, a Chernoff bound shows that to get the right answer to within factor $(1+\varepsilon)$ w. prob. $\geq 1-\delta$, need to take $t = O(\frac{m^2}{\varepsilon^2} \log \frac{m}{\delta})$ samples per level of the tree,

i.e. a total of $O(\frac{m^3}{\varepsilon^2} \log \frac{m}{\delta})$ samples.

- can be improved to $O(\frac{m^2}{\varepsilon^2} \log \frac{1}{\delta})$ using second moment arguments and repetition, outputting the median estimate of a bunch of estimates (ex 1pt)

(⇒ counting to sampling)

- Idea: using counting estimates, incrementally sample a path going down the tree

- at node u  use f.p.r.a.s. to get estimates $\hat{N}_\ell$ and $\hat{N}_r$

- then branch left w.p.r. $\frac{\hat{N}_\ell}{\hat{N}_\ell + \hat{N}_r}$, o.w. branch right and repeat.
- keep error of every estimate to within a factor of $(1+\frac{\varepsilon}{4m})$

⇒ probability that each leaf is sampled is within $(1+\varepsilon)$ factor from uniform, since bias is

$$\frac{1}{(1+\frac{\varepsilon}{4m})^2}\frac{N_\ell}{N_\ell+N_r} \le \frac{\hat{N}_\ell}{\hat{N}_\ell+\hat{N}_r} \le \frac{N_\ell}{N_\ell+N_r}\left(1+\frac{\varepsilon}{4m}\right)^2 \quad \text{per level of tree} \quad (*)$$

$$\text{factor of } \left(1+\frac{\varepsilon}{4m}\right)^{2m} \le 1+\varepsilon \text{ overall.}$$

⇒ hence total variation distance from uniform distn' is ε.

- $2m$ invocations of fpras, each w/ accuracy $\frac{\varepsilon}{2m}$, probability of success $1-\frac{\delta}{2m}$

Hence:

$$O\left(m \times \text{poly}\left(\tfrac{2m}{\varepsilon}, |x|\right) \times \log\tfrac{1}{\delta}\right) \text{ time overall.}$$

Remark: (*) only holds w/ probability $\ge 1-\delta$, for chosen δ at each level; so the output sample is within $\le\varepsilon$ from uniform w/ probability $\ge 1-\delta\cdot 2m$; the probability $\delta\cdot 2m$ of failure of (*) at some level introduces an extra bias in the output distribution that is $\delta\cdot 2m$; chose $\delta=\frac{\varepsilon}{2m}$.

Theorem 2 (all or nothing)

For self-reducible problem, if there exists a polynomial-time randomized algorithm for counting within a factor of $(1+\text{poly}(1/\varepsilon))^{\pm 1}$ with prob. $\geq 3/4$, then there exists a f.p.r.a.s.

e.g. it can count colorings (say) to within a factor of n^{100}, then I can also get a fpras!

this is rather striking as in typical optimization problems there are several levels of approximability fptas, ptas, constant factor, etc.

Proof of thm 2:

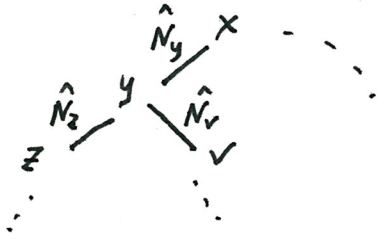
- work w/ self-reducibility tree; let N_z = # leaves in subtree rooted at z
- assume w.l.o.g. that we have a randomized algorithm that provides counting estimates that are within factor $\alpha = \text{poly}(n)$ w/ prob. 1, i.e. for any queried z the algorithm outputs $\hat{N}_z$ s.t. w.pr. 1:

$$\frac{N_z}{\alpha} \leq \hat{N}_z \leq \alpha \cdot N_z \quad (*)$$

our approach still works if (*) holds w.h.p. (ex 1pt).

- Define MC on self-reducibility tree, viewed as a weighted undirected graph whose edge weights are the estimates $w_{zy} = \hat{N}_z$, where z is lower node of the edge.

e.g.



Remarks: 1. assume weights of leaf edges are 1, since these are trivial instances of counting problem

2. "read once": if weight of edge has been estimated in a previous step of MC, don't estimate it again.

3. non-explicit: I won't write the tree down before the execution of the RW; construct the tree as I need it.

We prove:

1. stationary distn π of MC is uniform over leaves;
2. total weight of leaves in π is at least $\frac{\text{constant}}{\Delta \cdot m}$;
3. mixing time is $\tau_{mix} = O(m^2 \Delta^2 \log(\Delta m))$.

note: 1, 2, 3 suffice to prove theorem:

- expected # of repetitions of MC to see a leaf is $O(\Delta m)$
- if see a leaf, it is sampled uniformly;
- i.e. to sample a leaf uniformly at random, expected time is $O(m^3 \Delta^3 \log(\Delta m))$.

→ use these observations for approximately sampling (left as exercise)

Proof of 1: • easy to see that RW is reversible with respect to distribution $\pi(x) = \frac{D(x)}{D}$ where $D(x) = \sum_{e=(x,y)} w_e$, $D = \sum_x D(x)$

• since all leaf-edges have weight 1, $D(x) = 1$ for all leaves x ☒

Proof of 2:

- $\sum_{\text{leaves } x} \frac{D(x)}{D} = \frac{\#\text{leaves}}{D} = \frac{N}{D}$ (#leaves = N)
  - ↑ b.c. $D(x)=1$ for all leaves x
- $D = \sum_x D(x)$ = 2 × sum of all edge weights $\le 2 \times \alpha m N$
  - ↳ since the edge weights of each level α-approximate N
- Hence $\sum_{\text{leaves } x} \pi(x) = \frac{N}{D} \ge \frac{1}{2\alpha m}$ ☒

Proof of 3: ∘ use flow argument

- γ_{xy} : the unique path connecting x and y (so choice of flow is forced)
- consider edge e_z whose lower vertex is z 

$$f(e) = \sum_{\substack{x \in T_z \\ y \notin T_z}} \pi(x)\cdot\pi(y) = \pi(T_z)\cdot\pi(\bar{T}_z) \le \pi(T_z)$$

(this is flow $z \to w$; T_z ← tree rooted at z)

but $\pi(T_z) = \sum_{x \in T_z} \frac{D(x)}{D} \le \frac{1}{D} 2\alpha m \times N_z$ (because at most m levels in subtree rooted at z, and weights of each level α-approximate N_z)

hence $f(e) \le \frac{1}{D} 2\alpha m N_z$.

- on the other hand: $C(e) = \pi(z)\cdot\frac{\hat{N}_z}{D(z)} = \frac{D(z)}{D}\cdot\frac{\hat{N}_z}{D(z)} = \frac{\hat{N}_z}{D}$

hence $\frac{f(e)}{C(e)} \le 2\alpha m \frac{N_z}{\hat{N}_z} \le 2\alpha^2 m$

- So $\rho(t) \le 2\alpha^2 m$
- $\ell(t) \le 2m$

} mixing time from the root is $O(\alpha^2 m^2 \log \pi(\text{root})^{-1})$

$$\pi(\text{root}) = \frac{D(\text{root})}{D} \geq \frac{\frac{1}{\alpha} N}{2\alpha m N} \geq \frac{1}{2\alpha^2 m}$$

hence $\tau_{min} = O(\alpha^2 m^2 \log(\alpha m))$

□