Lecture 11

Approximate Counting

- Connection to uniform generation

Approximate Counting

Fully polynomial randomized approximation scheme (FPRAS)

Given formula ϕ, ε

s.t. $z = \#$ sat assignments to ϕ

Output y s.t.

$$\frac{z}{1+\varepsilon} \le y \le z \cdot (1+\varepsilon)$$

with prob $\ge 3/4$

Hope : runtime poly in $|\phi|, \frac{1}{\varepsilon}$

pset 1 problem 1:

algorithm that satisfies "hope"

$\Longrightarrow$ poly in $|\phi|, \underset{\substack{\uparrow \\ \text{approx error}}}{\frac{1}{\varepsilon}}, \log \frac{1}{\delta}$ $\longleftarrow$ prob of too much approx error

"Confidence"

FPRAS for SAT?

FPRAS for SAT $\Rightarrow$ randomized ptime algorithm for SAT:

<u>Algorithm for SAT:</u> Given formula ϕ

```
Call FPRAS on $\phi$ with $\varepsilon = 1/2$  ← any $\varepsilon > 0$ works
    if output $> 0$ output "Satisfiable"
    else output "unsatisfiable"
```

<u>Correctness</u> if ϕ satisfiable, $\#\phi \geq 1$ so

$y > \frac{1}{1+\varepsilon} > 0 \Rightarrow$ output "sat" ✓

if ϕ unsatisfiable, $\#\phi = 0$ so

$y = 0 \Rightarrow$ output "unsat" ✓

Exact vs. Approx Counting

Counting # SAT assignments to CNF is #P-complete
" " " " DNF " "
" " perfect matchings in graph " "
" " spanning trees in graph is in poly time

Is it hard to approx count?

CNF	hard
DNF	polytime ← today
Matching	polytime
Spanning trees	polytime

your favorite problem?

Approx counting for DNF:

Fully polynomial randomized approximation scheme (FPRAS)

Given ϕ, ε

s.t. $z = \#$ sat assignments to ϕ

Output y s.t.

$$\frac{z}{1+\varepsilon} \leq y \leq z \cdot (1+\varepsilon)$$

with prob $\geq 3/4$

Will use:

(1) uniform generation of DNF sat assignments

(2) "Downward self-reducibility" of DNF

Downward self-reducibility: (dsr)

Can compute problem by solving smaller subproblems & putting together answers via poly time computation.

Why is #-DNF dsr.?

$$\# \phi(x_1 \cdots x_n) = \# \phi(x_1 = T, x_2, \ldots x_n) +$$

$$\# \phi(x_1 = F, x_2, \ldots x_n)$$

both are still DNFs but in n-1 vars.

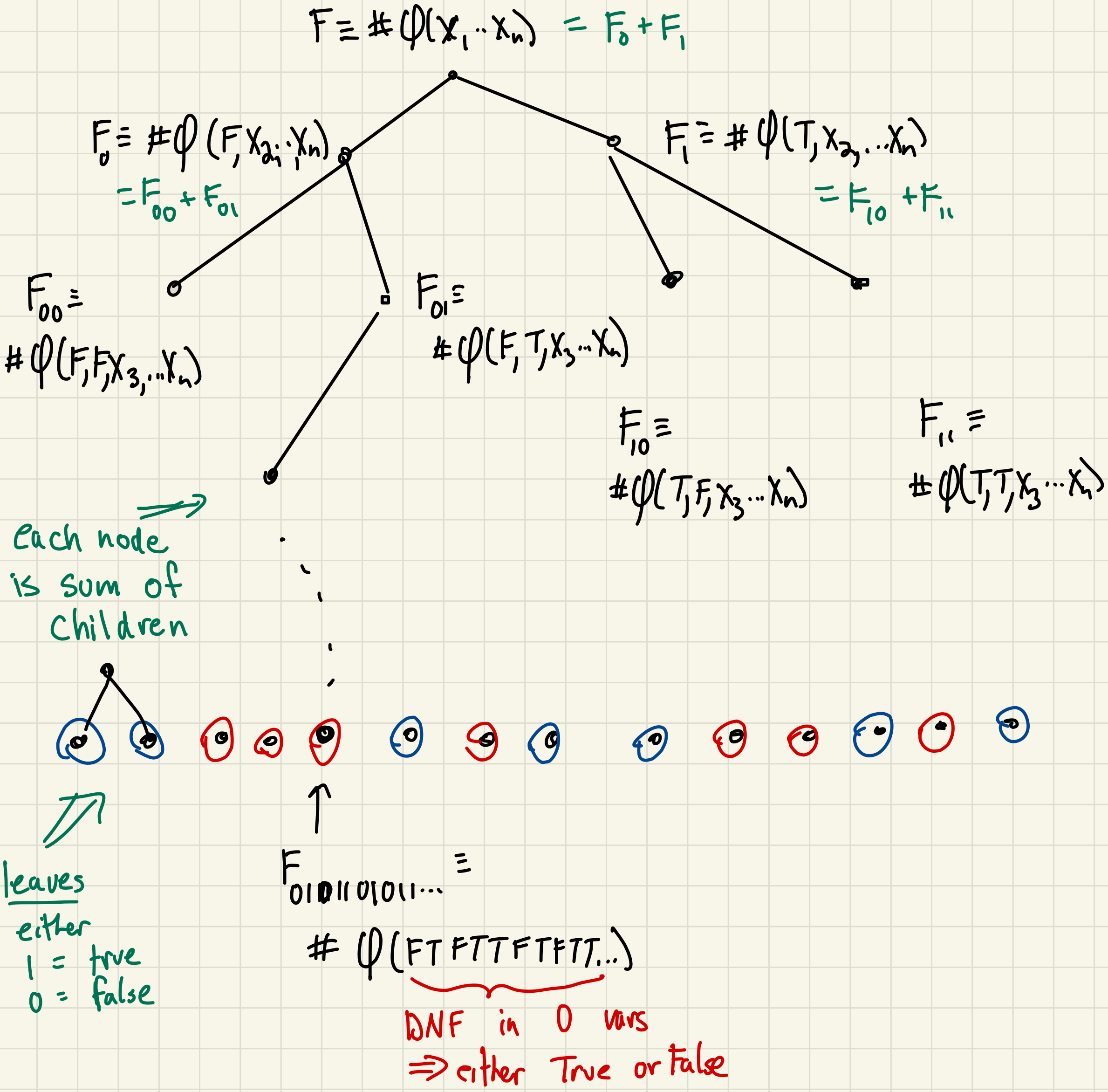
e.g. $\#(x_1 \bar{x}_2 \vee x_1 x_2 \vee \bar{x}_2)$

$= \#(\bar{x}_2)$ ← # settings where $x_1 = F$

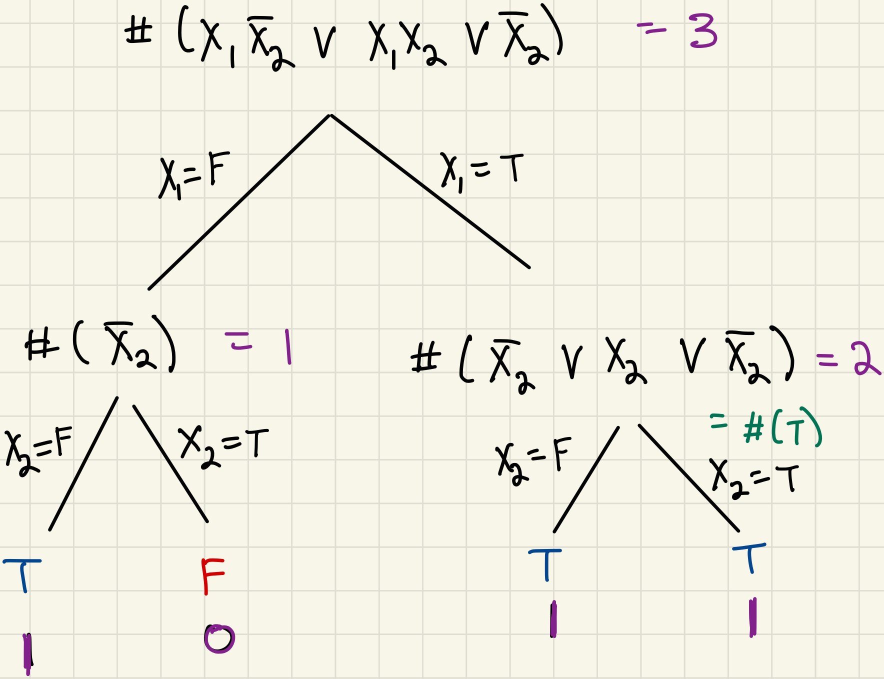
$+$

$\#(\bar{x}_2 \vee x_2 \vee \bar{x}_2)$ ← # settings where $x_1 = T$

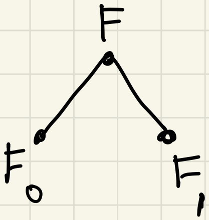
Downward Self-Reducibility Tree


$F \equiv \#\varphi(x_1 \cdots x_n) = F_0 + F_1$
$F_0 \equiv \#\varphi(F, x_2, \cdots, x_n)$
$= F_{00} + F_{01}$
$F_1 \equiv \#\varphi(T, x_2, \ldots x_n)$
$= F_{10} + F_{11}$
$F_{00} \equiv \#\varphi(F, F, x_3, \ldots x_n)$
$F_{01} \equiv \#\varphi(F, T, x_3 \cdots x_n)$
$F_{10} \equiv \#\varphi(T, F, x_3 \cdots x_n)$
$F_{11} \equiv \#\varphi(T, T, x_3 \cdots x_n)$
each node is sum of children
leaves either
1 = true
0 = false
$F_{0100110101 1\ldots} \equiv$
$\#\varphi(\underbrace{FTFTTFTFTT\ldots}_{})$
DNF in 0 vars
⇒ either True or False

example


$\#(x_1\bar{x}_2 \vee x_1x_2 \vee \bar{x}_2)$ = 3
$x_1 = F$
$x_1 = T$
$\#(\bar{x}_2)$ = 1
$\#(\bar{x}_2 \vee x_2 \vee \bar{x}_2) = 2$
$= \#(T)$
$x_2 = F$
$x_2 = T$
$x_2 = F$
$x_2 = T$
T
F
T
T
1
0
1
1

Approximate Counting Algorithm for #DNF



$$\text{Let } S_1 = \underbrace{\frac{F_1}{F}}_{\text{Fraction of sat assignments s.t. } x_1 = T} \Rightarrow F = \frac{F_1}{S_1}$$

main insight: for DNF, we can estimate S_1 via sampling!

- uniformly generate k sat assignments ← we know how to do this for DNF!!
- $\tilde{S}_1 \leftarrow \dfrac{\#\text{ with } x_1 = T}{K}$

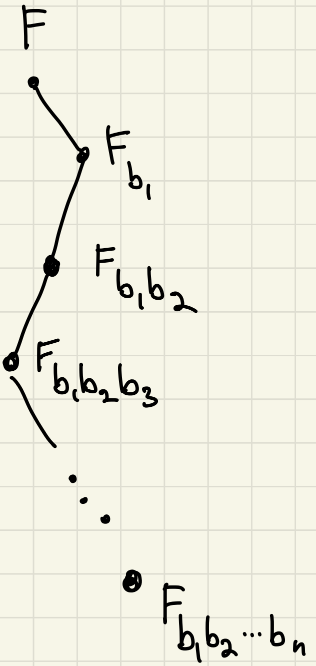
But how do we compute F_1?

recursively!

$$F_1 = \frac{F_{11}}{S_{11}}$$

← recurse on formula with one fewer variables

← estimate



So $F = \frac{F_{b_1}}{S_{b_1}} = \frac{F_{b_1b_2}}{S_{b_1}\cdot S_{b_1b_2}} = \frac{F_{b_1b_2b_3}}{S_{b_1}\cdot S_{b_1b_2}\cdot S_{b_1b_2b_3}}$

$\vdots$

$= \frac{1}{\prod_{i=1}^{n} S_{b_1\cdots b_i}}$

Potential Difficulties:

1. if $F_{b_1\cdots b_n}=0$ this doesn't work
2. Is approximation of $S_{b_1..b_i}$'s good enough? only get additive estimates ☹

Idea Always take path of "larger" child

↑ might guess wrong when both have lots of SAT assignments but soon will show that is ok

Claim if always pick b_i s.t. $F_{b_1..b_i} > F_{b_1..\bar{b}_i}$ then always reach SAT assignment leaf.
(So $F_{b_1..b_n}=1$)

Idea estimate each $S_{b_1 \cdots b_i}$ to within $\frac{\varepsilon}{8n}$ additive error (using Chernoff bnds, need only poly$\left(\frac{2n}{\varepsilon}, \log 8n\right)$ samples to get error $< \frac{1}{4n}$)

↑ union bnd over all i to get prob of error $< \frac{1}{4}$

$\Rightarrow$ if $1 \geq r \geq 1/2$

$$r + \frac{\varepsilon}{8n} \leq r\left(1 + \frac{\varepsilon}{8n \cdot r}\right) \leq r\left(1 + \frac{\varepsilon}{4n}\right)$$

* slight issue: might be estimating $1-r$ if pick wrong path. We will ignore this for now.

$$r - \frac{\varepsilon}{8n} \geq r\left(1 - \frac{\varepsilon}{8nr}\right) \geq r\left(1 - \frac{\varepsilon}{8n}\right)$$

Claim

$$\text{output} \leq \frac{F_{b_1}}{\tilde{S}_{b_1}} \leq \frac{F_{b_1 b_2}}{\tilde{S}_{b_1} \tilde{S}_{b_1 b_2}} \leq \cdots \leq \frac{1}{\prod \tilde{S}_{b_1 \cdots b_i}}$$

$$\leq \frac{\left(1 + \frac{\varepsilon}{4n}\right)^n}{\prod S_{b_1 \cdots b_n}} = F \cdot \underbrace{\left(1 + \frac{\varepsilon}{4n}\right)^n}_{} \leq F(1+\varepsilon)$$

$$\left. \begin{array}{l} 1 + \frac{\varepsilon}{4} + \frac{\left(\frac{\varepsilon}{4}\right)^2}{2!} + \cdots \\ \leq 1 + \varepsilon \end{array} \right\} \text{Taylor series}$$

similarly, $\text{output} \geq \frac{F}{1+\varepsilon}$

∎

Recursive Algorithm

- estimate S_0, S_1 from unif generated SAT assignments
- let $b_1 \leftarrow \operatorname{argmax}\{S_0, S_1\}$
- recurse on F_{b_1}

runtime?

$\le n \cdot$ #samples to get $\frac{\varepsilon}{4n}$ additive error $\cdot$ runtime of uniform generator

$\uparrow$ #recursions

$\uparrow$ poly $\underbrace{\left(\frac{\varepsilon}{4n}\right)^{-1}}_{\text{approx error}} \cdot \underbrace{\left(\frac{1}{4n}\right)^{-1}}_{\text{confidence error}}$

$\uparrow$ poly in n

total: poly $(n, \frac{1}{\varepsilon})$

$$\Pr[\text{algorithm fails}] \le \sum_{\substack{\text{recursion level}\\ i=1}}^{n} \Pr[\text{estimate bad}] \le n \cdot \frac{1}{4n} \le \frac{1}{4}.$$

Works for any d.s.r. problem!

poly time (almost)-uniform-generation of solutions

⇓ ⇑ what about this direction?

poly time approximate counting of # solns

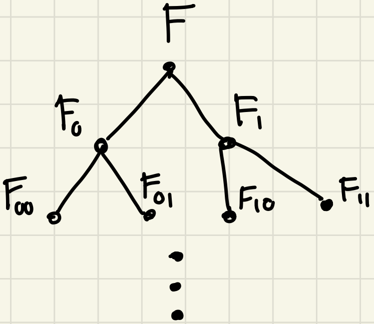
<u>Thm</u> [Jerrum Valiant Vazirani] for any problem in NP that is d.s.r.:

ptime approx counting of # solutions ⟺ ptime almost uniform generation

(easier case)

(Perfect) counting for # DNF ⇒

(perfect) Uniform generation



Recursive algorithm:

at $b_1 \ldots b_i$,

use (perfect) counter to compute

$$r_0 = F_{b_1 \ldots b_i 0}$$

$$r_1 = F_{b_1 \ldots b_i 1}$$

go left with prob $\frac{r_0}{r_0 + r_1}$

& right o.w.

<u>Claim</u> (1) always reach SAT assignment

since never take branch with 0 SAT assignments underneath

(2) $$\Pr[\text{output } \underbrace{b_1 \ldots b_n}_{\text{SAT assignment}}] = \frac{F_{b_1}}{F} \cdot \frac{F_{b_1 b_2}}{F_{b_1}} \cdot \frac{F_{b_1 b_2 b_3}}{F_{b_1 b_2}} \cdot \ldots \cdot \frac{1}{F_{b_1 b_2 \ldots b_n}}$$

$$= \frac{1}{F}$$ ← same for every sat assignment

Question what if only have approx counter?

Answer $\text{RHS} \le \frac{1}{F}\left(\frac{1+\varepsilon'}{1-\varepsilon'}\right)^n \le \frac{1}{F}\cdot\frac{1}{1-\varepsilon}$

if choose $\varepsilon' < \frac{\varepsilon}{2n}$

$\Longrightarrow$ close to uniform generation of sat assignments