Probabilistic Computation

Lecture 12

Flipping coins, taking chances

PP, BPP

Probabilistic Computation

Probabilistic Computation

- Output depends not only on x, but also on random "coin flips"

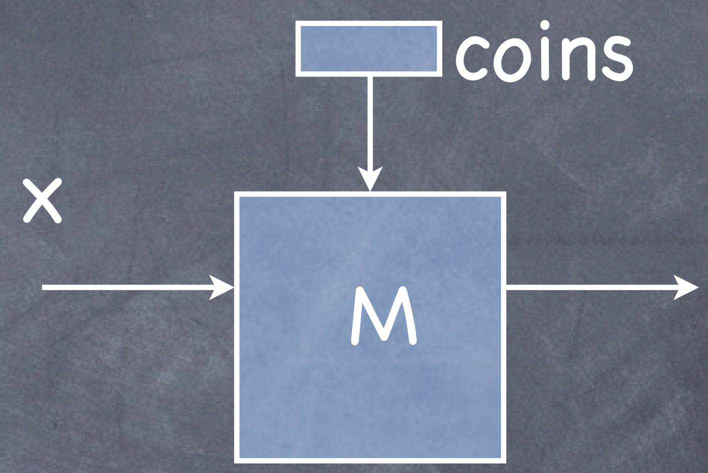
Probabilistic Computation

- Output depends not only on x, but also on random "coin flips"
- M,x define a probability distribution over outcomes

Probabilistic Computation

- Output depends not only on x, but also on random "coin flips"
- M,x define a probability distribution over outcomes
- If for all x, M(x) equals f(x) with very high probability, could be used as f(x)

Probabilistic Computation



- Output depends not only on x, but also on random "coin flips"
- M,x define a probability distribution over outcomes
- If for all x, M(x) equals f(x) with very high probability, could be used as f(x)

Language Decided by a Probabilistic Computation

Language Decided by a Probabilistic Computation

- Different possible definitions of a prob. TM accepting input

Language Decided by a Probabilistic Computation

- Different possible definitions of a prob. TM accepting input
  - M accepts x if pr[M(x)=yes] > 0; rejects if pr[M(x)=yes] = 0

Language Decided by a Probabilistic Computation

- Different possible definitions of a prob. TM accepting input
  - M accepts x if pr[M(x)=yes] > 0; rejects if pr[M(x)=yes] = 0
  - M accepts x if pr[M(x)=yes] > 1/2; rejects if pr[M(x)=yes] ≤ 1/2

Language Decided by a Probabilistic Computation

- Different possible definitions of a prob. TM accepting input
  - M accepts x if pr[M(x)=yes] > 0; rejects if pr[M(x)=yes] = 0
  - M accepts x if pr[M(x)=yes] > 1/2; rejects if pr[M(x)=yes] ≤ 1/2
  - M accepts x if pr[M(x)=yes] > 2/3; rejects if pr[M(x)=yes] < 1/3

Language Decided by a Probabilistic Computation

- Different possible definitions of a prob. TM accepting input
  - M accepts x if pr[M(x)=yes] > 0; rejects if pr[M(x)=yes] = 0
  - M accepts x if pr[M(x)=yes] > 1/2; rejects if pr[M(x)=yes] ≤ 1/2
  - M accepts x if pr[M(x)=yes] > 2/3; rejects if pr[M(x)=yes] < 1/3
  - M accepts x if pr[M(x)=yes] > 2/3; rejects if pr[M(x)=yes] = 0

Language Decided by a Probabilistic Computation

- Different possible definitions of a prob. TM accepting input
  - M accepts x if pr[M(x)=yes] > 0; rejects if pr[M(x)=yes] = 0
  - M accepts x if pr[M(x)=yes] > 1/2; rejects if pr[M(x)=yes] ≤ 1/2
  - M accepts x if pr[M(x)=yes] > 2/3; rejects if pr[M(x)=yes] < 1/3
  - M accepts x if pr[M(x)=yes] > 2/3; rejects if pr[M(x)=yes] = 0
    - Last two: If on any x neither, M doesn't decide a language!

Language Decided by a Probabilistic Computation

- Different possible definitions of a prob. TM accepting input
  - M accepts x if pr[M(x)=yes] > 0; rejects if pr[M(x)=yes] = 0
  - M accepts x if pr[M(x)=yes] > 1/2; rejects if pr[M(x)=yes] ≤ 1/2
  - M accepts x if pr[M(x)=yes] > 2/3; rejects if pr[M(x)=yes] < 1/3
  - M accepts x if pr[M(x)=yes] > 2/3; rejects if pr[M(x)=yes] = 0
    - Last two: If on any x neither, M doesn't decide a language!
      - When M does decide, much better than random guess

Probabilistic TM

Probabilistic TM

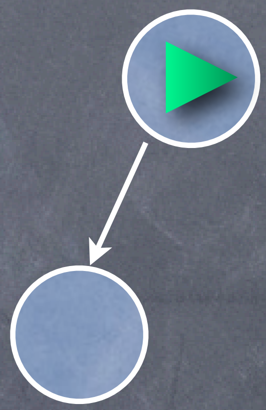
- Like an NTM, but the two possible transitions are considered to be taken with equal probability

Probabilistic TM

- Like an NTM, but the two possible transitions are considered to be taken with equal probability

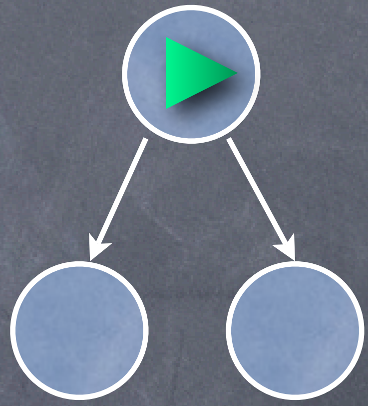
Probabilistic TM

- Like an NTM, but the two possible transitions are considered to be taken with equal probability



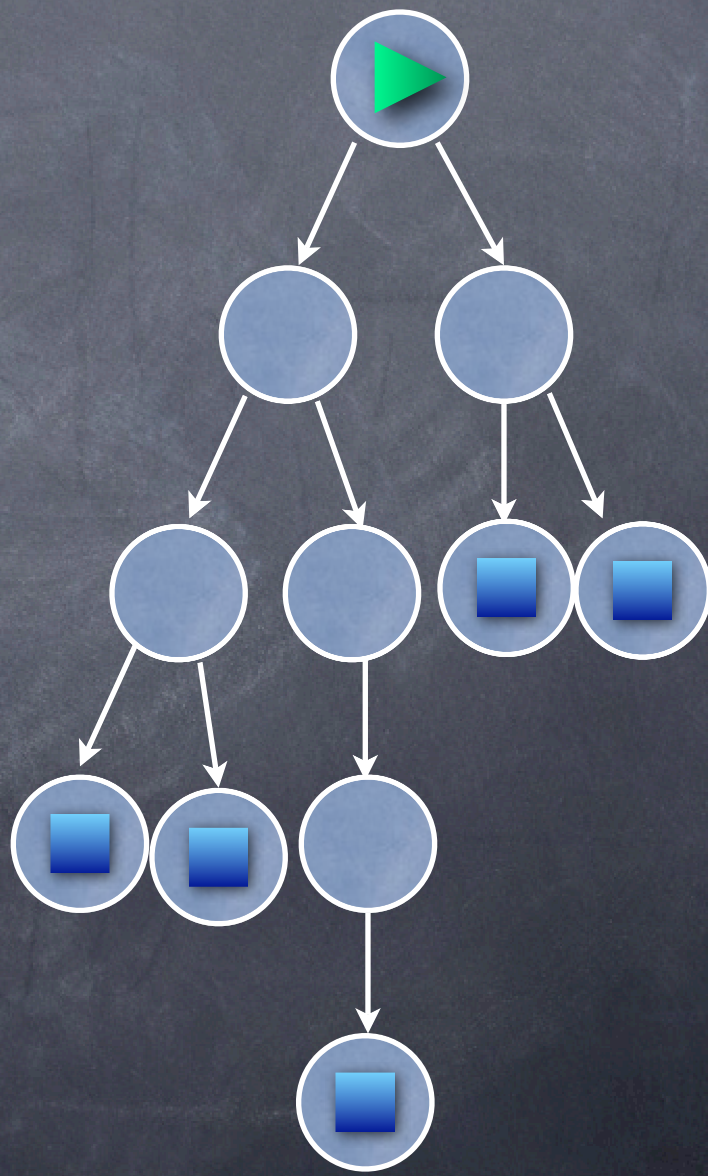
Probabilistic TM

- Like an NTM, but the two possible transitions are considered to be taken with equal probability



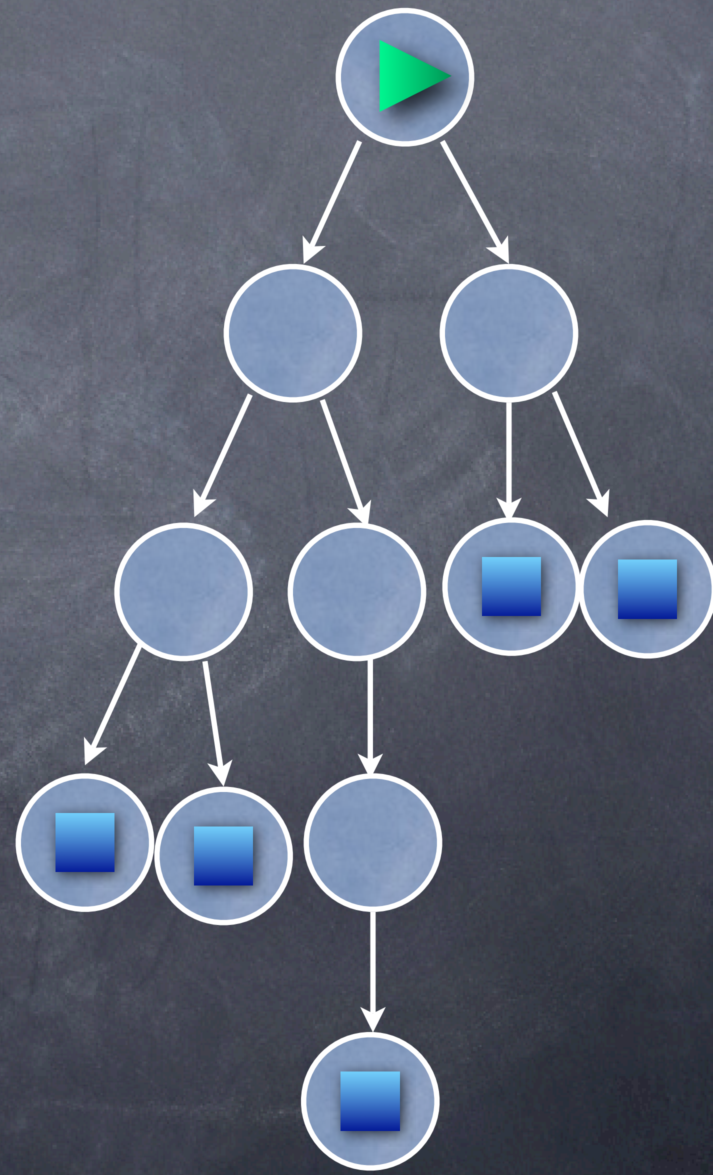
Probabilistic TM

- Like an NTM, but the two possible transitions are considered to be taken with equal probability



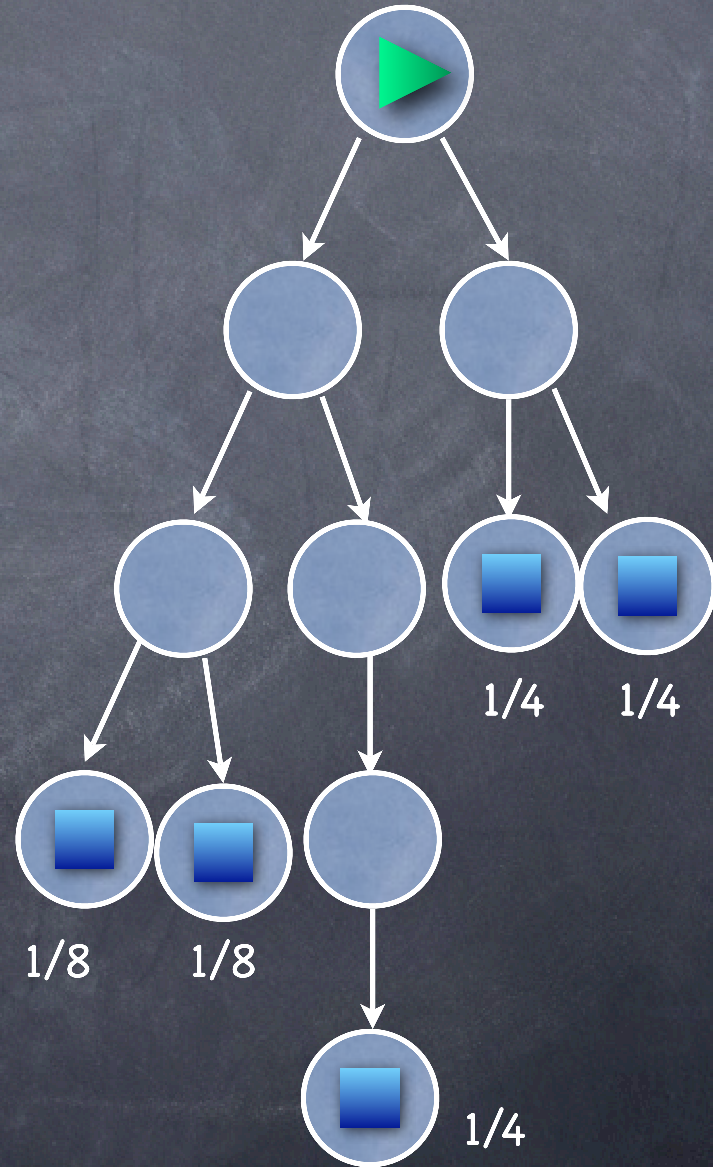
Probabilistic TM

- Like an NTM, but the two possible transitions are considered to be taken with equal probability
  - Defines a probability with which an input is accepted or rejected

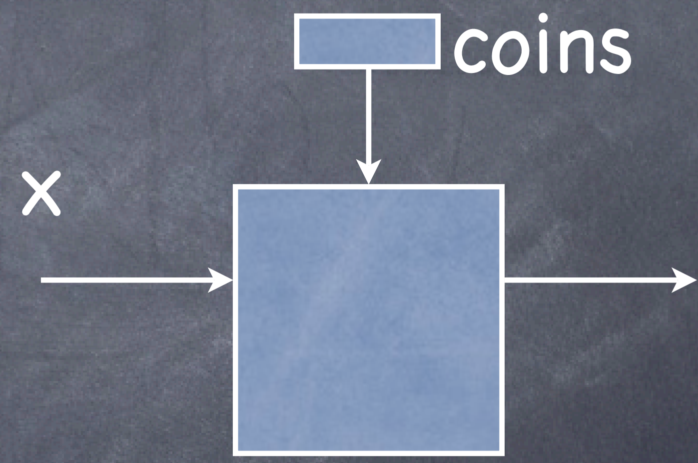


Probabilistic TM

- Like an NTM, but the two possible transitions are considered to be taken with equal probability
  - Defines a probability with which an input is accepted or rejected

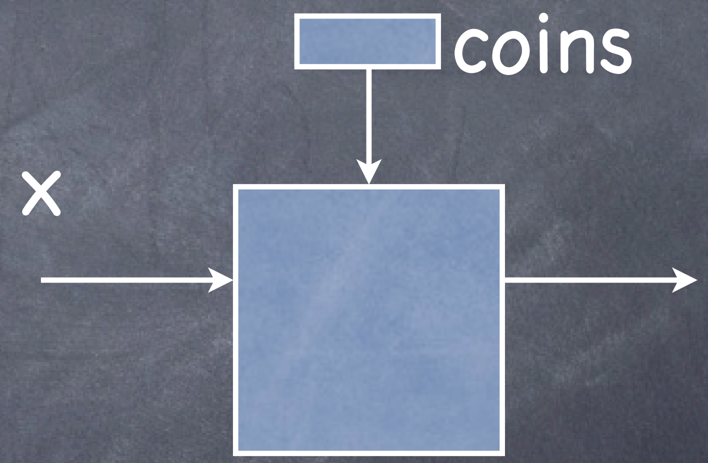


Random Tape



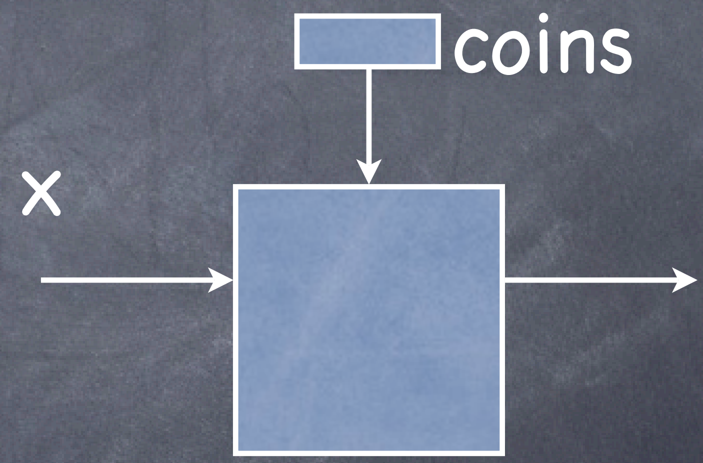
Random Tape

- Random choice: flipping a fair coin



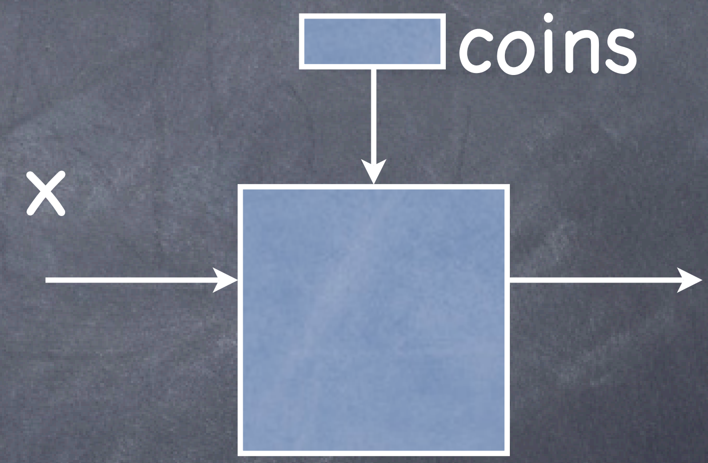
Random Tape

- Random choice: flipping a fair coin
  - Coin flip is written on a read-once "random tape"



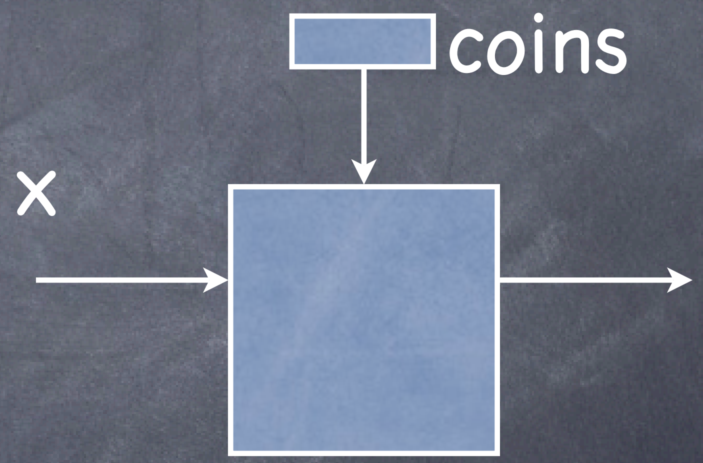
Random Tape

- Random choice: flipping a fair coin
  - Coin flip is written on a read-once "random tape"
  - Enough coin flips made and written on the tape first, then start execution

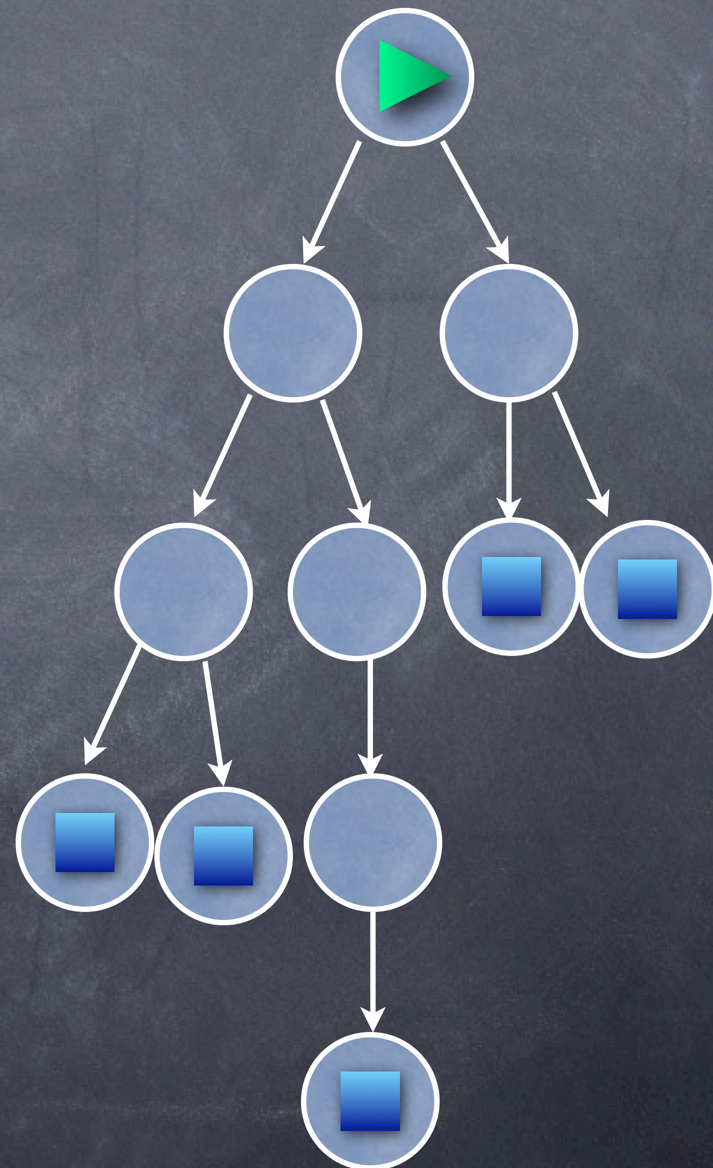


Random Tape

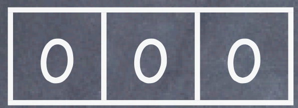
- Random choice: flipping a fair coin
  - Coin flip is written on a read-once "random tape"
  - Enough coin flips made and written on the tape first, then start execution
    - When considering bounded time TMs length of random tape (max coins used) also bounded

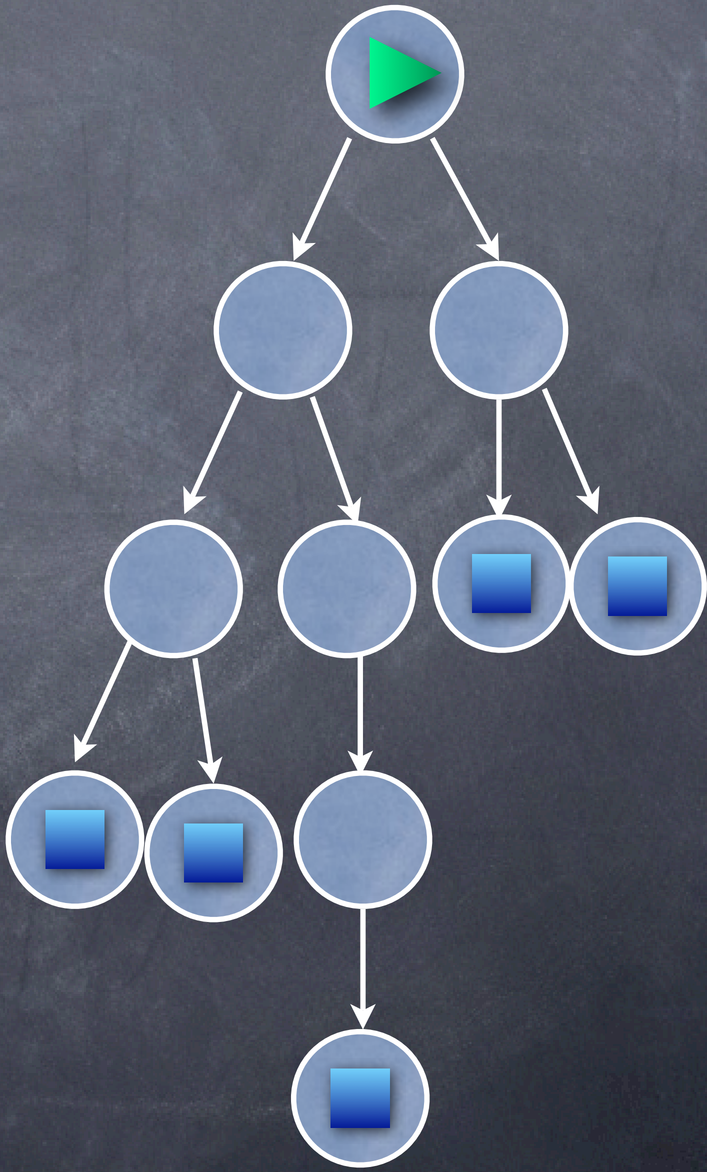


Random Tape



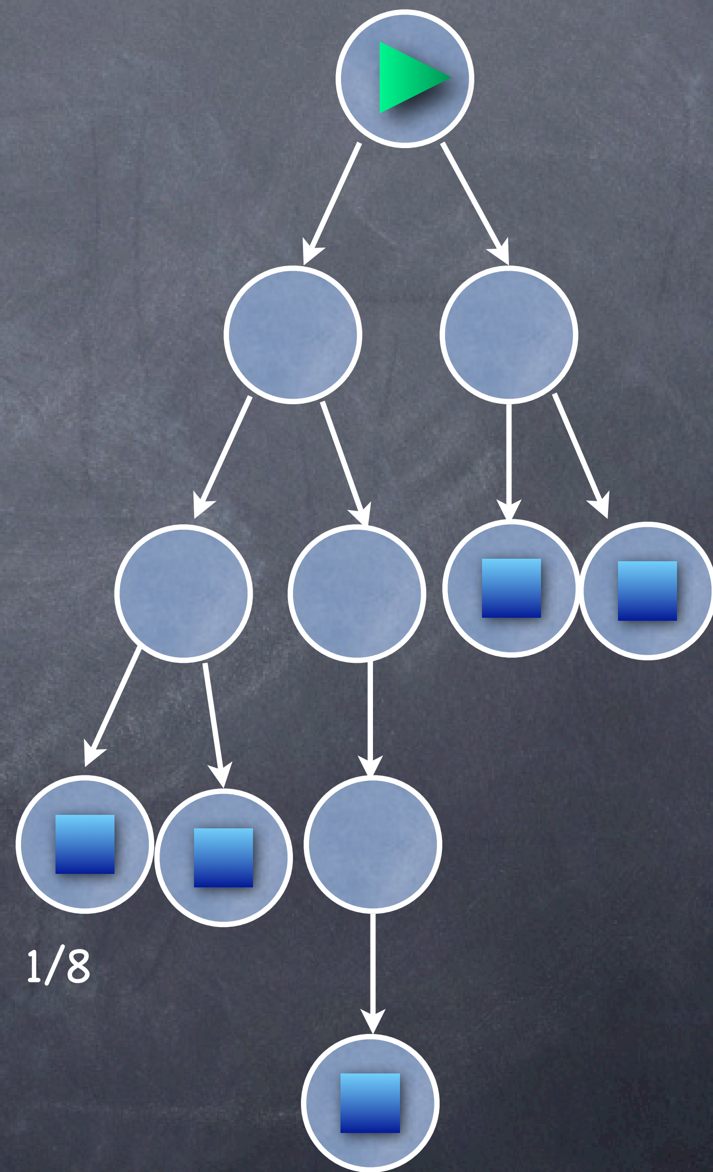
Random Tape





Random Tape

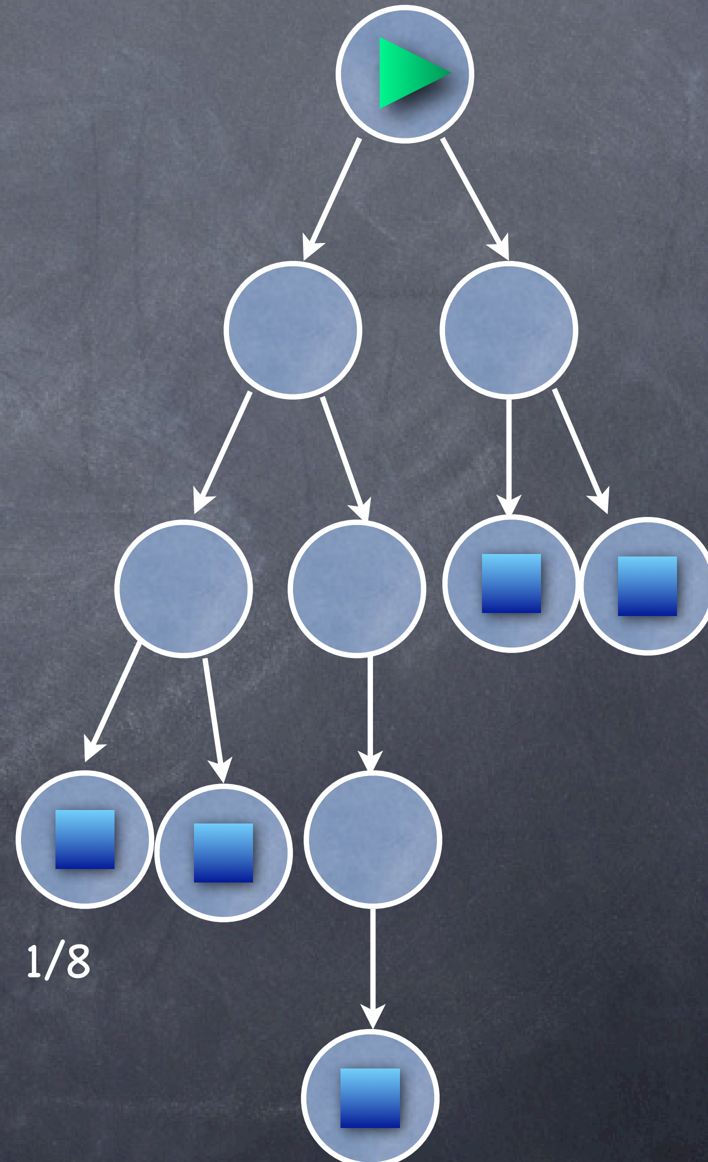
0	0	0



Random Tape

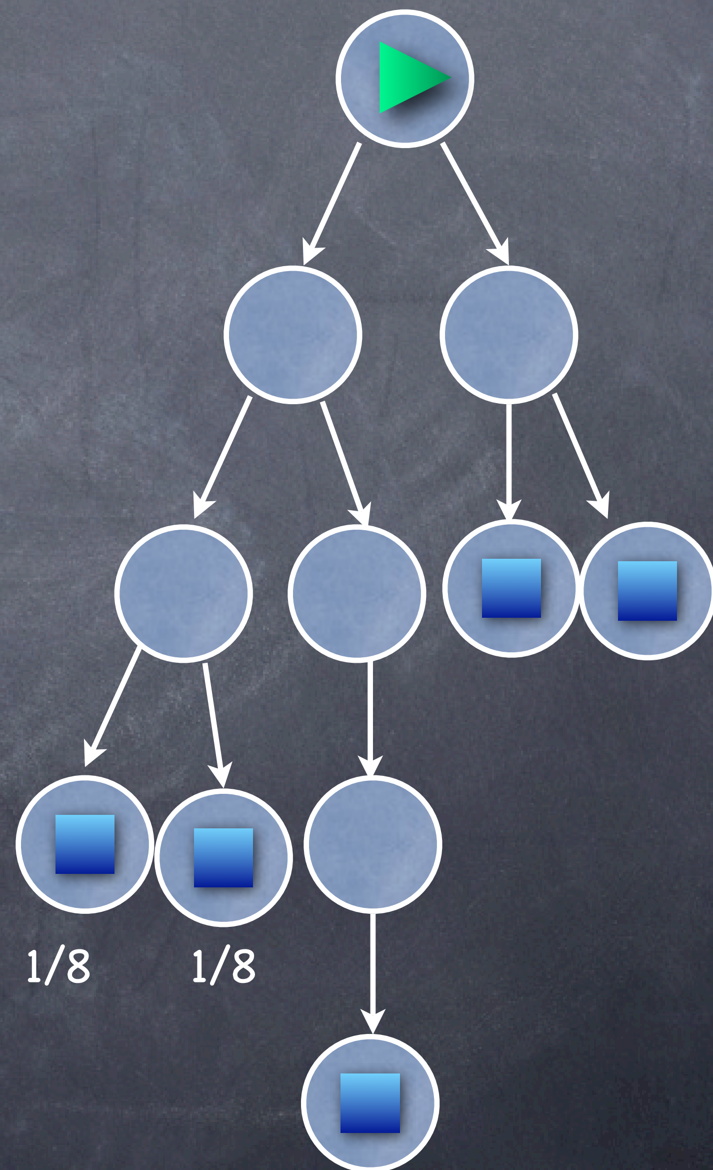
0	0	0

0	0	1



Random Tape

0	0	0
0	0	1

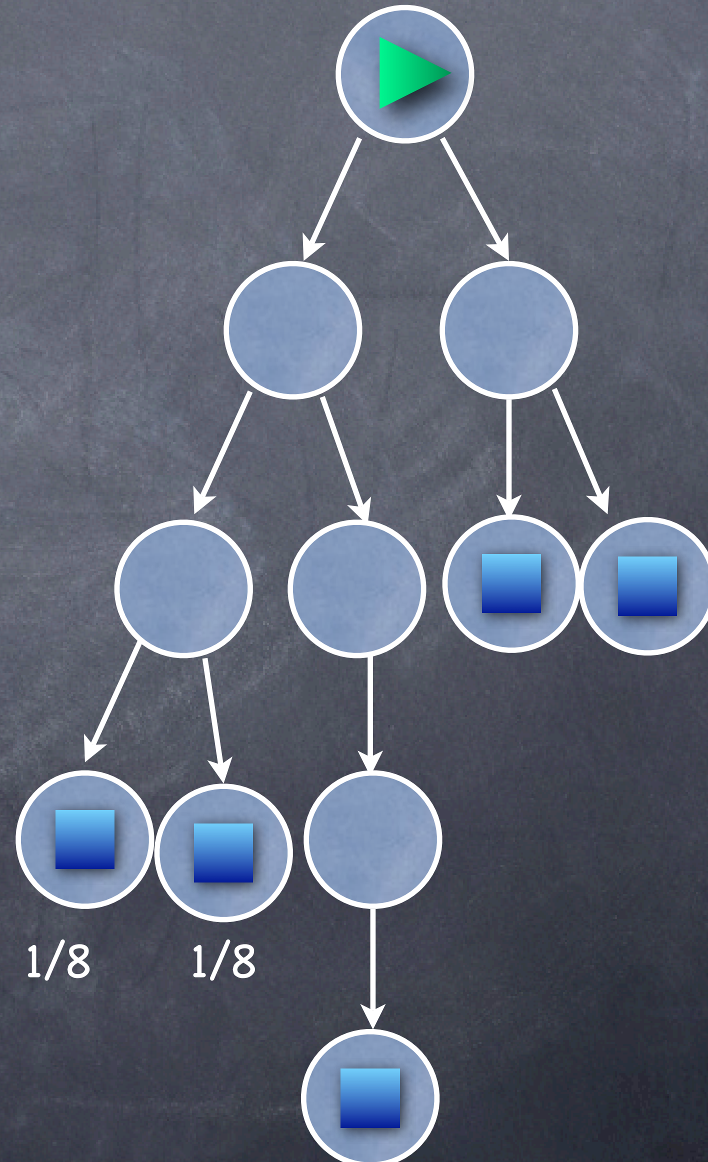


Random Tape

0	0	0

0	0	1

0	1

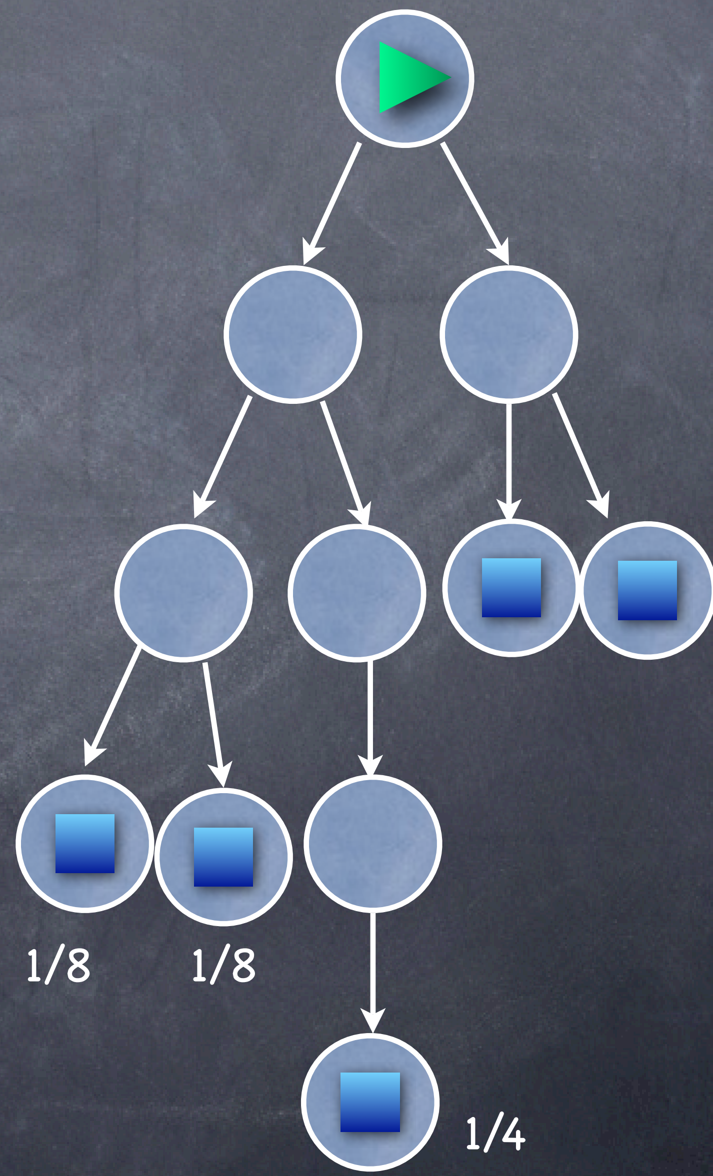


Random Tape

0	0	0

0	0	1

0	1



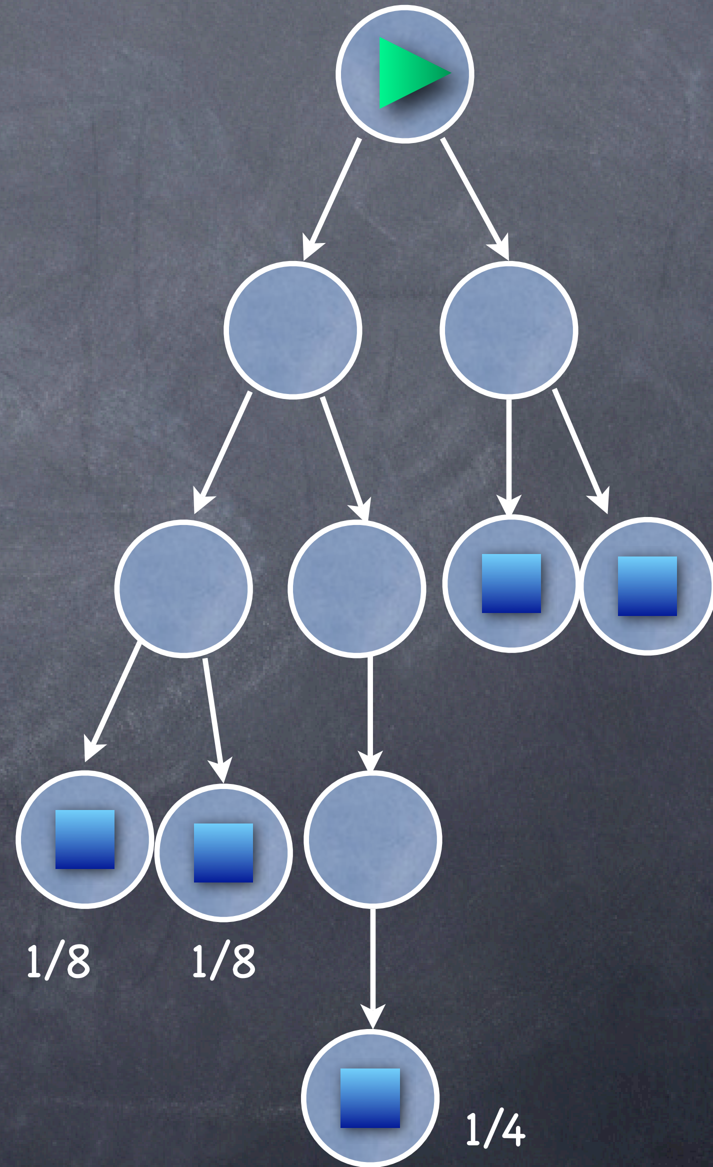
Random Tape

0	0	0

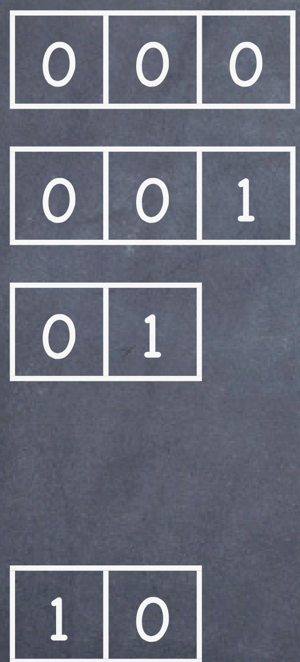
0	0	1

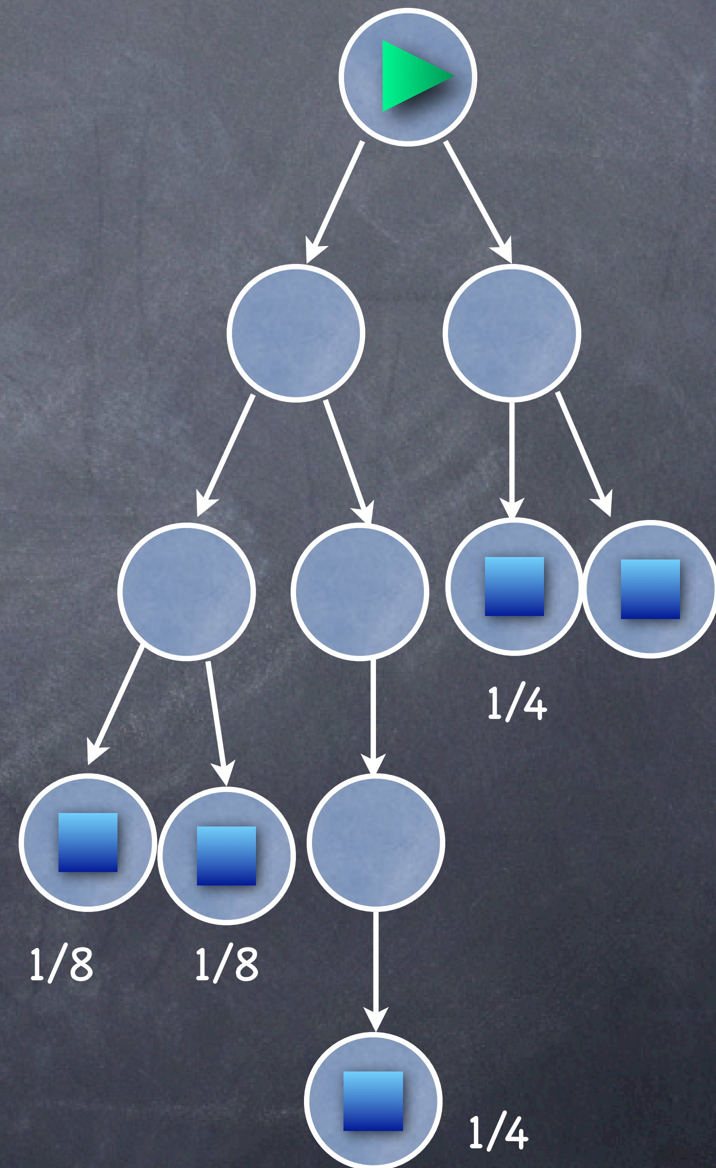
0	1

1	0



Random Tape





Random Tape

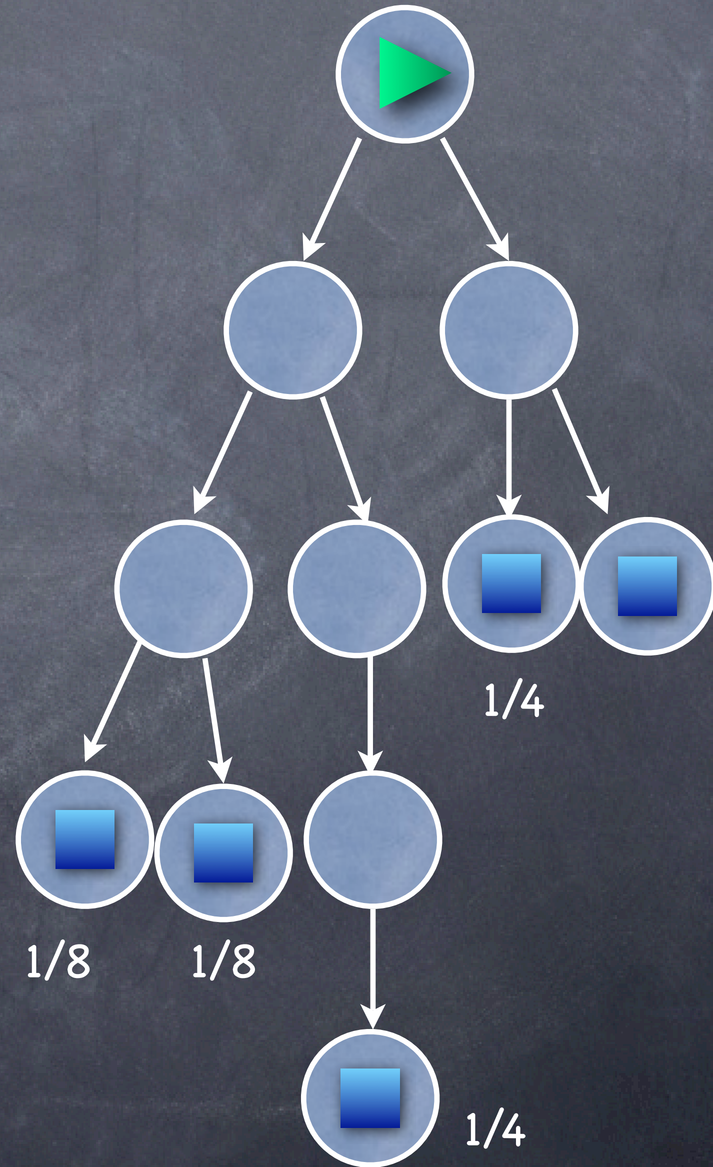
0	0	0

0	0	1

0	1

1	0

1	1



Random Tape

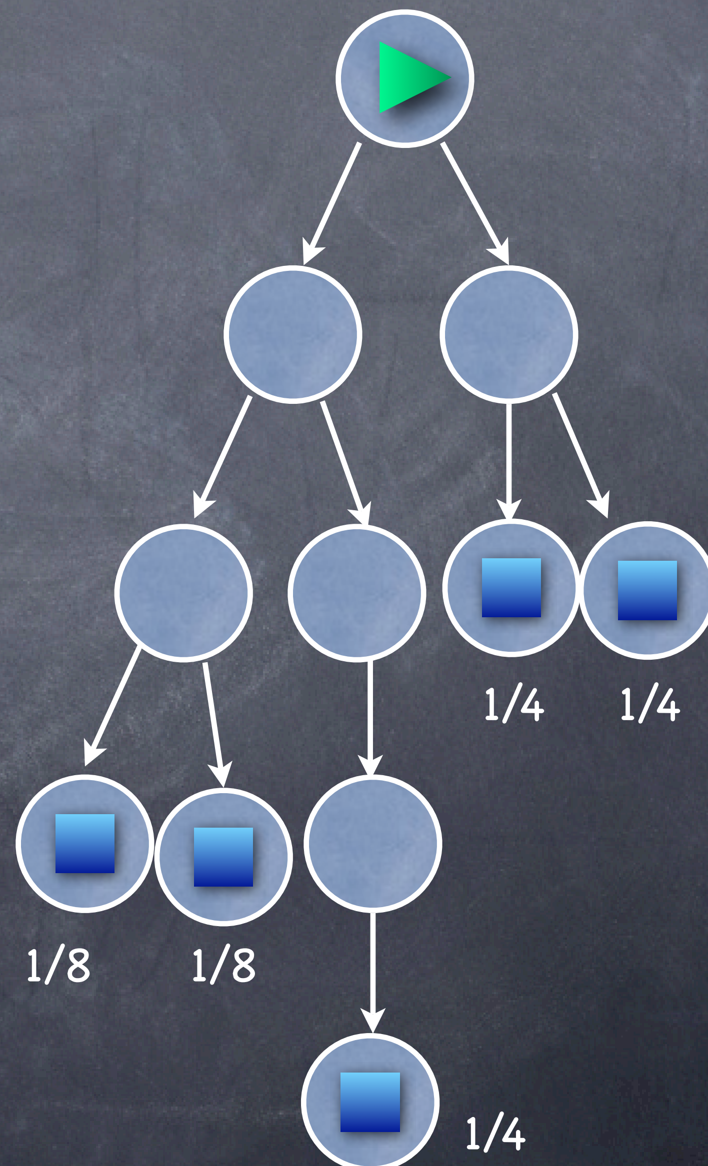
0	0	0

0	0	1

0	1

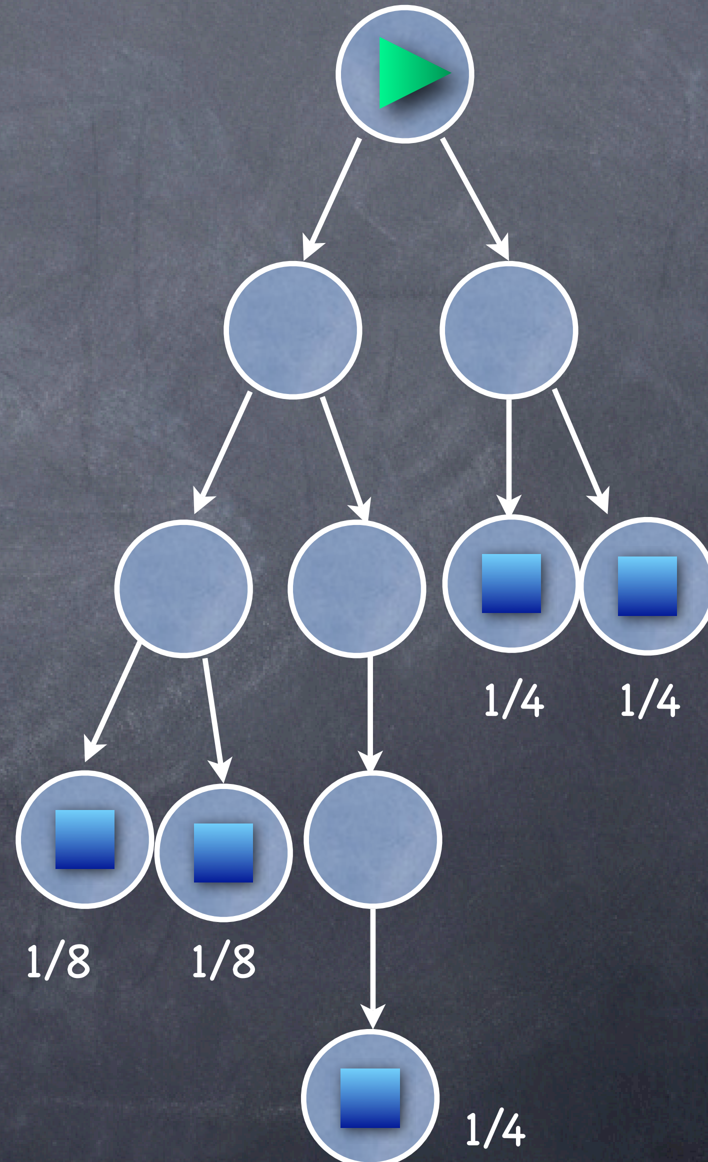
1	0

1	1



Random Tape

0 0 0	0 0 0
0 0 1	0 0 1
0 1	0 1 0
	0 1 1
1 0	1 0 0
	1 0 1
1 1	1 1 0
	1 1 1



Language Decided by a Probabilistic Computation

Language Decided by a Probabilistic Computation

- Different possible definitions of accepting input

Language Decided by a Probabilistic Computation

- Different possible definitions of accepting input
  - Accept if pr[yes] > 0; reject if pr[yes] = 0

Language Decided by a Probabilistic Computation

- Different possible definitions of accepting input
  - Accept if pr[yes] > 0; reject if pr[yes] = 0 NTM

Language Decided by a Probabilistic Computation

- Different possible definitions of accepting input
  - Accept if pr[yes] > 0; reject if pr[yes] = 0 NTM
  - Accept if pr[yes] > 1/2; reject if pr[yes] ≤ 1/2

Language Decided by a Probabilistic Computation

- Different possible definitions of accepting input
  - Accept if pr[yes] > 0; reject if pr[yes] = 0 NTM
  - Accept if pr[yes] > 1/2; reject if pr[yes] ≤ 1/2 PTM

Language Decided by a Probabilistic Computation

- Different possible definitions of accepting input
  - Accept if pr[yes] > 0; reject if pr[yes] = 0 — NTM
  - Accept if pr[yes] > 1/2; reject if pr[yes] ≤ 1/2 — PTM
  - Accept if pr[yes] > 2/3; reject if pr[yes] < 1/3

Language Decided by a Probabilistic Computation

- Different possible definitions of accepting input
  - Accept if pr[yes] > 0; reject if pr[yes] = 0 NTM
  - Accept if pr[yes] > 1/2; reject if pr[yes] ≤ 1/2 PTM
  - Accept if pr[yes] > 2/3; reject if pr[yes] < 1/3 BPTM

Language Decided by a Probabilistic Computation

- Different possible definitions of accepting input
  - Accept if pr[yes] > 0; reject if pr[yes] = 0 — NTM
  - Accept if pr[yes] > 1/2; reject if pr[yes] ≤ 1/2 — PTM
  - Accept if pr[yes] > 2/3; reject if pr[yes] < 1/3 — BPTM
  - Accept if pr[yes] > 2/3; reject if pr[yes] = 0

Language Decided by a Probabilistic Computation

- Different possible definitions of accepting input
  - Accept if pr[yes] > 0; reject if pr[yes] = 0 — NTM
  - Accept if pr[yes] > 1/2; reject if pr[yes] ≤ 1/2 — PTM
  - Accept if pr[yes] > 2/3; reject if pr[yes] < 1/3 — BPTM
  - Accept if pr[yes] > 2/3; reject if pr[yes] = 0 — RTM

Language Decided by a Probabilistic Computation

- Different possible definitions of accepting input
  - Accept if pr[yes] > 0; reject if pr[yes] = 0 — NTM
  - Accept if pr[yes] > 1/2; reject if pr[yes] ≤ 1/2 — PTM
  - Accept if pr[yes] > 2/3; reject if pr[yes] < 1/3 — BPTM
  - Accept if pr[yes] > 2/3; reject if pr[yes] = 0 — RTM
- (Not standard nomenclature!)

PTIME, BPTIME and RTIME

PTIME, BPTIME and RTIME

- T-time probabilistic TM

PTIME, BPTIME and RTIME

- T-time probabilistic TM
  - On all inputs x, on any random tape, terminates in T(|x|) time and outputs "yes" or "no."

PTIME, BPTIME and RTIME

- T-time probabilistic TM
  - On all inputs x, on any random tape, terminates in T(|x|) time and outputs "yes" or "no."
  - Just like NTIME(T)

PTIME, BPTIME and RTIME

- T-time probabilistic TM
  - On all inputs x, on any random tape, terminates in T(|x|) time and outputs "yes" or "no."
  - Just like NTIME(T)
- BPTIME(T) = class of languages decided by BPTMs in time T

PTIME, BPTIME and RTIME

- T-time probabilistic TM
  - On all inputs x, on any random tape, terminates in T(|x|) time and outputs "yes" or "no."
  - Just like NTIME(T)
- BPTIME(T) = class of languages decided by BPTMs in time T
  - Similarly PTIME(T) and RTIME(T)

PP, BPP and RP

PP, BPP and RP

- PP = $\cup_{c>0}$ PTIME($O(n^c)$)

PP, BPP and RP

- $PP = \cup_{c>0} PTIME(O(n^c))$
- $BPP = \cup_{c>0} BPTIME(O(n^c))$

PP, BPP and RP

- PP = $\cup_{c>0}$ PTIME($O(n^c)$)
- BPP = $\cup_{c>0}$ BPTIME($O(n^c)$)
- RP = $\cup_{c>0}$ RTIME($O(n^c)$)

PP, BPP and RP

- PP = $\bigcup_{c>0}$ PTIME($O(n^c)$)
- BPP = $\bigcup_{c>0}$ BPTIME($O(n^c)$)
- RP = $\bigcup_{c>0}$ RTIME($O(n^c)$)
  - co-RP

co-RTM

co-RTM

- Accept if pr[yes] > 0; reject if pr[yes] = 0 NTM

co-RTM

- Accept if pr[yes] > 0; reject if pr[yes] = 0 NTM
    - Accept if pr[yes] = 1; reject if pr[yes] < 1 co-NTM

co-RTM

- Accept if pr[yes] > 0; reject if pr[yes] = 0 — NTM
  - Accept if pr[yes] = 1; reject if pr[yes] < 1 — co-NTM
- Accept if pr[yes] > 1/2; reject if pr[yes] ≤ 1/2 — PTM

co-RTM

- Accept if pr[yes] > 0; reject if pr[yes] = 0 NTM
    - Accept if pr[yes] = 1; reject if pr[yes] < 1 co-NTM
- Accept if pr[yes] > 1/2; reject if pr[yes] ≤ 1/2 PTM
    - Accept if pr[yes] ≥ 1/2; reject if pr[yes] < 1/2 co-PTM

co-RTM

- Accept if pr[yes] > 0; reject if pr[yes] = 0 NTM
    - Accept if pr[yes] = 1; reject if pr[yes] < 1 co-NTM
- Accept if pr[yes] > 1/2; reject if pr[yes] ≤ 1/2 PTM
    - Accept if pr[yes] ≥ 1/2; reject if pr[yes] < 1/2 co-PTM
- Accept if pr[yes] > 2/3; reject if pr[yes] < 1/3 BPTM

co-RTM

- Accept if pr[yes] > 0; reject if pr[yes] = 0 — NTM
  - Accept if pr[yes] = 1; reject if pr[yes] < 1 — co-NTM
- Accept if pr[yes] > 1/2; reject if pr[yes] ≤ 1/2 — PTM
  - Accept if pr[yes] ≥ 1/2; reject if pr[yes] < 1/2 — co-PTM
- Accept if pr[yes] > 2/3; reject if pr[yes] < 1/3 — BPTM
  - Accept if pr[yes] > 2/3; reject if pr[yes] < 1/3 — co-BPTM

co-RTM

- Accept if pr[yes] > 0; reject if pr[yes] = 0 — NTM
    - Accept if pr[yes] = 1; reject if pr[yes] < 1 — co-NTM
- Accept if pr[yes] > 1/2; reject if pr[yes] ≤ 1/2 — PTM
    - Accept if pr[yes] ≥ 1/2; reject if pr[yes] < 1/2 — co-PTM
- Accept if pr[yes] > 2/3; reject if pr[yes] < 1/3 — BPTM
    - Accept if pr[yes] > 2/3; reject if pr[yes] < 1/3 — co-BPTM
- Accept if pr[yes] > 2/3; reject if pr[yes] = 0 — RTM

co-RTM

- Accept if pr[yes] > 0; reject if pr[yes] = 0 NTM
    - Accept if pr[yes] = 1; reject if pr[yes] < 1 co-NTM
- Accept if pr[yes] > 1/2; reject if pr[yes] ≤ 1/2 PTM
    - Accept if pr[yes] ≥ 1/2; reject if pr[yes] < 1/2 co-PTM
- Accept if pr[yes] > 2/3; reject if pr[yes] < 1/3 BPTM
    - Accept if pr[yes] > 2/3; reject if pr[yes] < 1/3 co-BPTM
- Accept if pr[yes] > 2/3; reject if pr[yes] = 0 RTM
    - Accept if pr[yes] = 1; reject if pr[yes] < 1/3 co-RTM

RP and co-RP

RP and co-RP

- One sided error ("bounded error" versions of NP and co-NP)

RP and co-RP

- One sided error (“bounded error” versions of NP and co-NP)
  - RP: if yes, may still say no w/p at most 1/3

RP and co-RP

- One sided error ("bounded error" versions of NP and co-NP)
  - RP: if yes, may still say no w/p at most 1/3
    - i.e., if RTM says no, can be wrong

RP and co-RP

- One sided error ("bounded error" versions of NP and co-NP)
  - RP: if yes, may still say no w/p at most 1/3
    - i.e., if RTM says no, can be wrong
  - co-RP: if no, may still say yes w/p at most 1/3

RP and co-RP

- One sided error ("bounded error" versions of NP and co-NP)
  - RP: if yes, may still say no w/p at most 1/3
    - i.e., if RTM says no, can be wrong
  - co-RP: if no, may still say yes w/p at most 1/3
    - i.e., if co-RTM says yes, can be wrong

co-BPP, co-PP

co-BPP, co-PP

- BPP = co-BPP

co-BPP, co-PP

- BPP = co-BPP
  - co-BPTMs are same as BPTMs

co-BPP, co-PP

- BPP = co-BPP
  - co-BPTMs are same as BPTMs
- In fact PP = co-PP

co-BPP, co-PP

- BPP = co-BPP
  - co-BPTMs are same as BPTMs
- In fact PP = co-PP
  - PTMs and co-PTMs differ on accepting inputs with Pr[yes]=1/2

co-BPP, co-PP

- BPP = co-BPP
  - co-BPTMs are same as BPTMs
- In fact PP = co-PP
  - PTMs and co-PTMs differ on accepting inputs with Pr[yes]=1/2
  - But can modify a PTM so that Pr[M(x)=yes] ≠ 1/2 for all x, without changing language accepted

PP = co-PP

PP = co-PP

- Modifying a PTM M to an equivalent PTM M′, so that for all x $\Pr[M'(x)=\text{yes}] \neq 1/2$

PP = co-PP

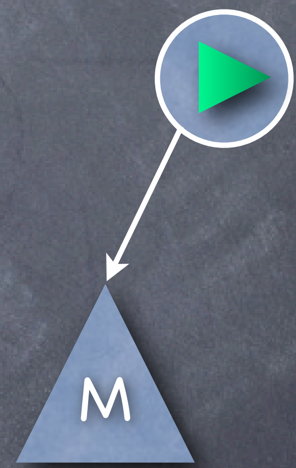
- Modifying a PTM M to an equivalent PTM M′, so that for all x Pr[M′(x)=yes] ≠ 1/2
  - Consider M′(x): w.p. 1/2 run M(x); w.p. 1/2, ignore input and say yes w.p. 1/2 - ϵ, and say no w.p. 1/2 + ϵ

PP = co-PP

- Modifying a PTM M to an equivalent PTM M′, so that for all x Pr[M′(x)=yes] ≠ 1/2
    - Consider M′(x): w.p. 1/2 run M(x); w.p. 1/2, ignore input and say yes w.p. 1/2 - ϵ, and say no w.p. 1/2 + ϵ

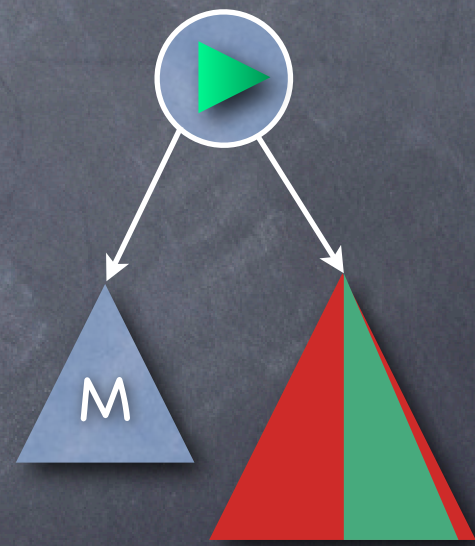
PP = co-PP

- Modifying a PTM M to an equivalent PTM M′, so that for all x Pr[M′(x)=yes] ≠ 1/2
  - Consider M′(x): w.p. 1/2 run M(x); w.p. 1/2, ignore input and say yes w.p. 1/2 - ϵ, and say no w.p. 1/2 + ϵ



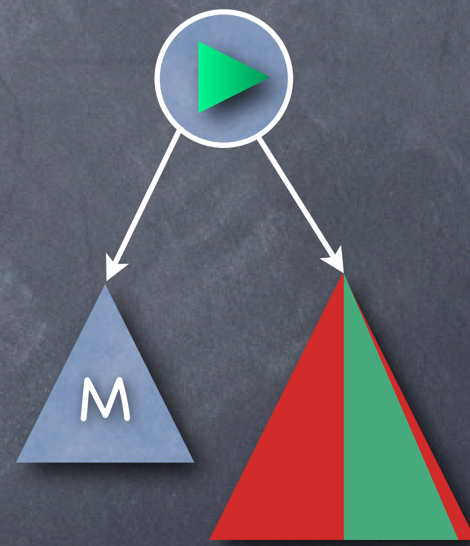
PP = co-PP

- Modifying a PTM M to an equivalent PTM M′, so that for all x Pr[M′(x)=yes] ≠ 1/2
  - Consider M′(x): w.p. 1/2 run M(x); w.p. 1/2, ignore input and say yes w.p. 1/2 - ε, and say no w.p. 1/2 + ε



PP = co-PP

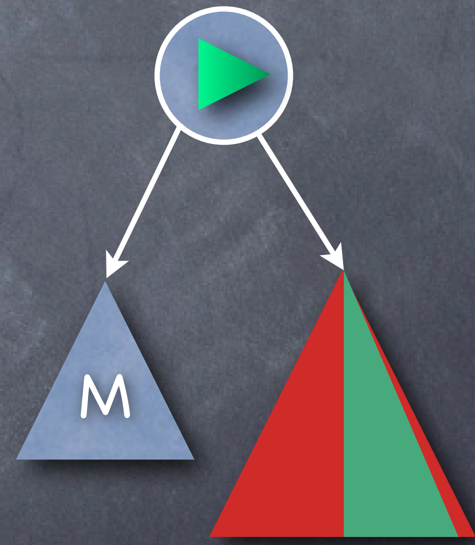
- Modifying a PTM M to an equivalent PTM M′, so that for all x Pr[M′(x)=yes] ≠ 1/2
  - Consider M′(x): w.p. 1/2 run M(x); w.p. 1/2, ignore input and say yes w.p. 1/2 - ϵ, and say no w.p. 1/2 + ϵ
    - pr[M′(x)=yes] = pr[M(x)=yes]/2 + (1/2 - ϵ)/2



PP = co-PP

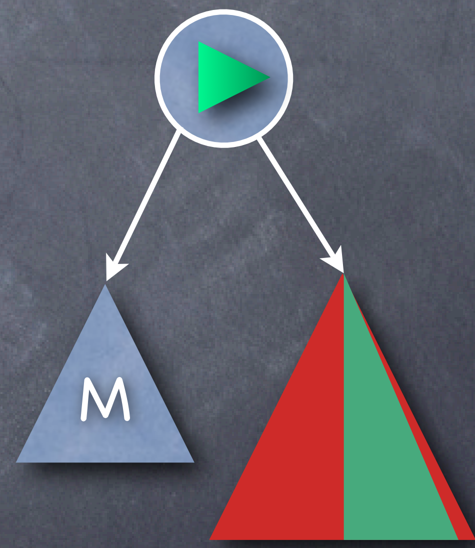
- Modifying a PTM M to an equivalent PTM M′, so that for all x Pr[M′(x)=yes] ≠ 1/2
  - Consider M′(x): w.p. 1/2 run M(x); w.p. 1/2, ignore input and say yes w.p. 1/2 - ϵ, and say no w.p. 1/2 + ϵ
    - pr[M′(x)=yes] = pr[M(x)=yes]/2 + (1/2 - ϵ)/2
  - If pr[M(x)=yes] > 1/2 ⇒ pr[M(x)=yes] > 1/2 + ϵ

    then M and M′ equivalent



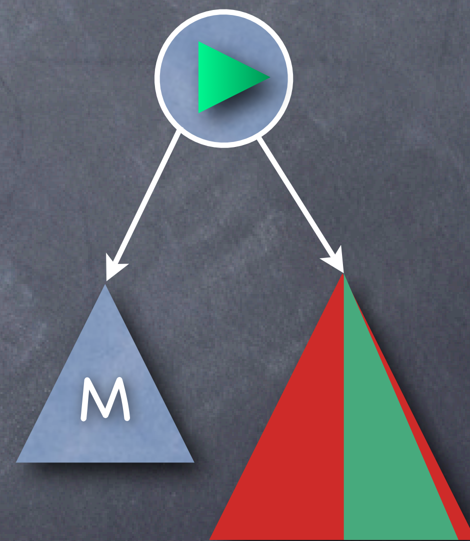
PP = co-PP

- Modifying a PTM M to an equivalent PTM M′, so that for all x Pr[M′(x)=yes] ≠ 1/2
  - Consider M′(x): w.p. 1/2 run M(x); w.p. 1/2, ignore input and say yes w.p. 1/2 - ϵ, and say no w.p. 1/2 + ϵ
    - pr[M′(x)=yes] = pr[M(x)=yes]/2 + (1/2 - ϵ)/2
  - If pr[M(x)=yes] > 1/2 ⇒ pr[M(x)=yes] > 1/2 + ϵ then M and M′ equivalent
  - What is such an ϵ?



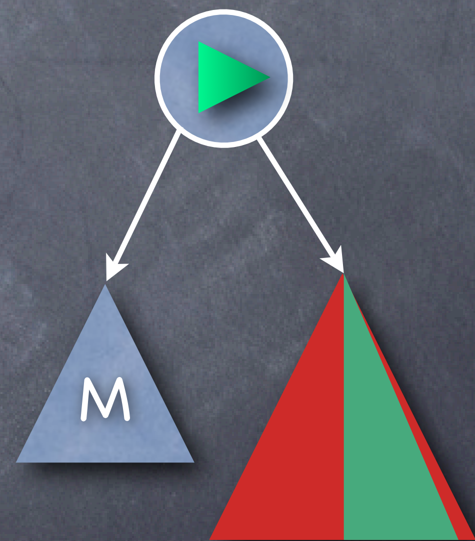
PP = co-PP

- Modifying a PTM M to an equivalent PTM M′, so that for all x Pr[M′(x)=yes] ≠ 1/2
  - Consider M′(x): w.p. 1/2 run M(x); w.p. 1/2, ignore input and say yes w.p. 1/2 - ε, and say no w.p. 1/2 + ε
    - pr[M′(x)=yes] = pr[M(x)=yes]/2 + (1/2 - ε)/2
  - If pr[M(x)=yes] > 1/2 ⇒ pr[M(x)=yes] > 1/2 + ε then M and M′ equivalent
  - What is such an ε?
    - $2^{-(m+1)}$ where no. of coins used by M is at most m



PP = co-PP

- Modifying a PTM M to an equivalent PTM M′, so that for all x Pr[M′(x)=yes] ≠ 1/2
  - Consider M′(x): w.p. 1/2 run M(x); w.p. 1/2, ignore input and say yes w.p. 1/2 - ϵ, and say no w.p. 1/2 + ϵ
    - pr[M′(x)=yes] = pr[M(x)=yes]/2 + (1/2 - ϵ)/2
  - If pr[M(x)=yes] > 1/2 ⇒ pr[M(x)=yes] > 1/2 + ϵ then M and M′ equivalent
  - What is such an ϵ?
    - $2^{-(m+1)}$ where no. of coins used by M is at most m
      - M′ tosses at most m+2 coins



PP and NP

PP and NP

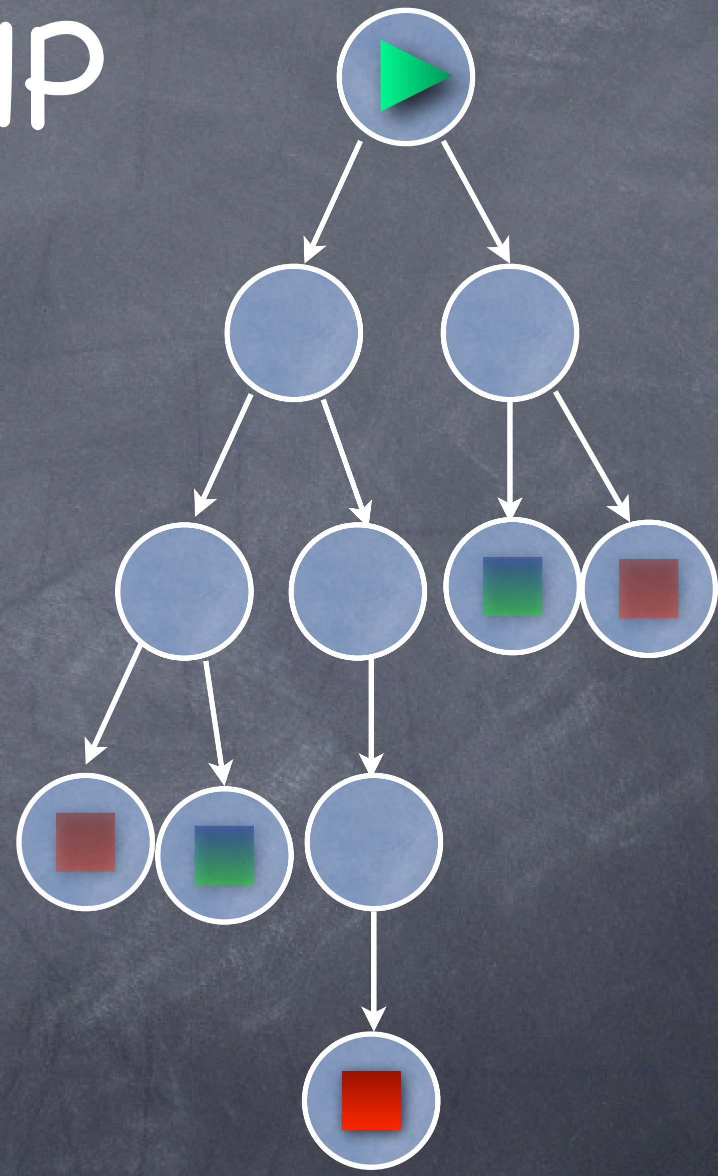
- $NP \subseteq PP$

PP and NP

- NP $\subseteq$ PP
  - Use random-tape as non-deterministic choices of NTM M

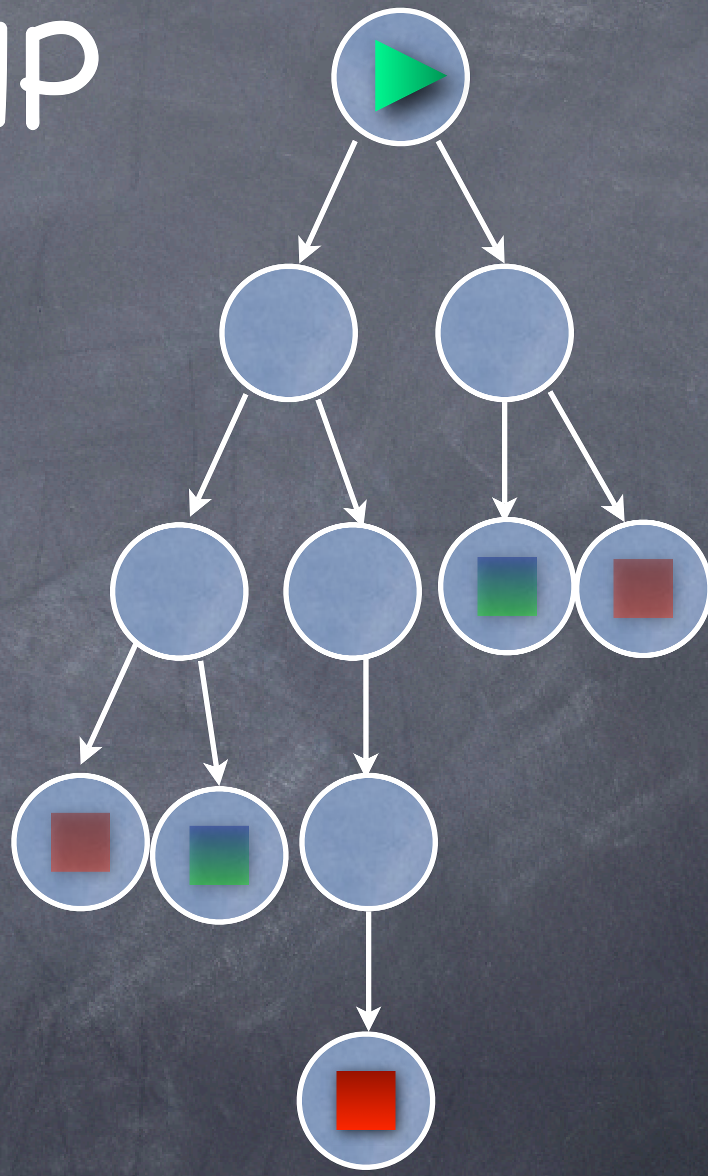
PP and NP

- NP ⊆ PP
  - Use random-tape as non-deterministic choices of NTM M



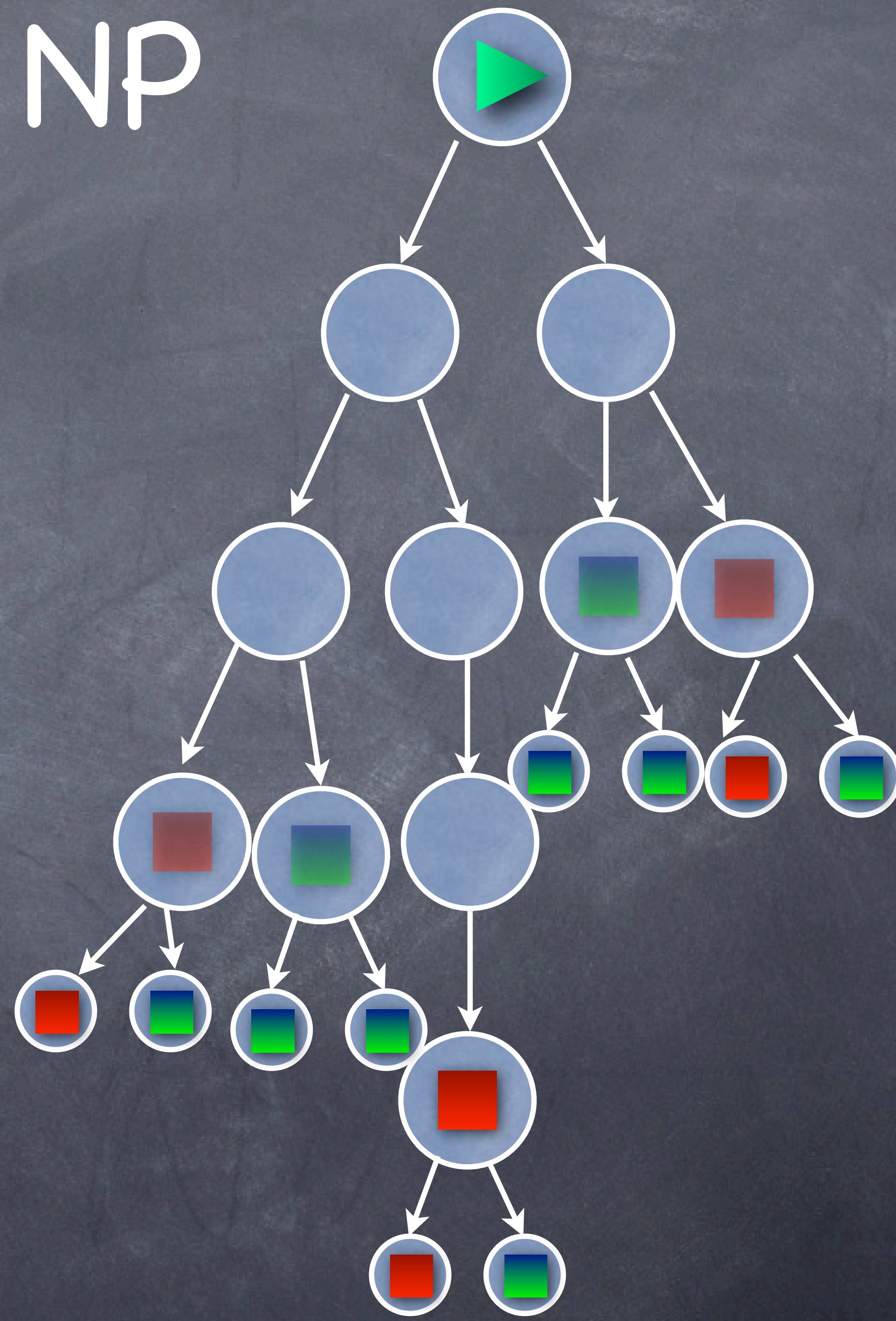
PP and NP

- NP ⊆ PP
  - Use random-tape as non-deterministic choices of NTM M
  - If M rejects, accept with 1/2 prob., else accept



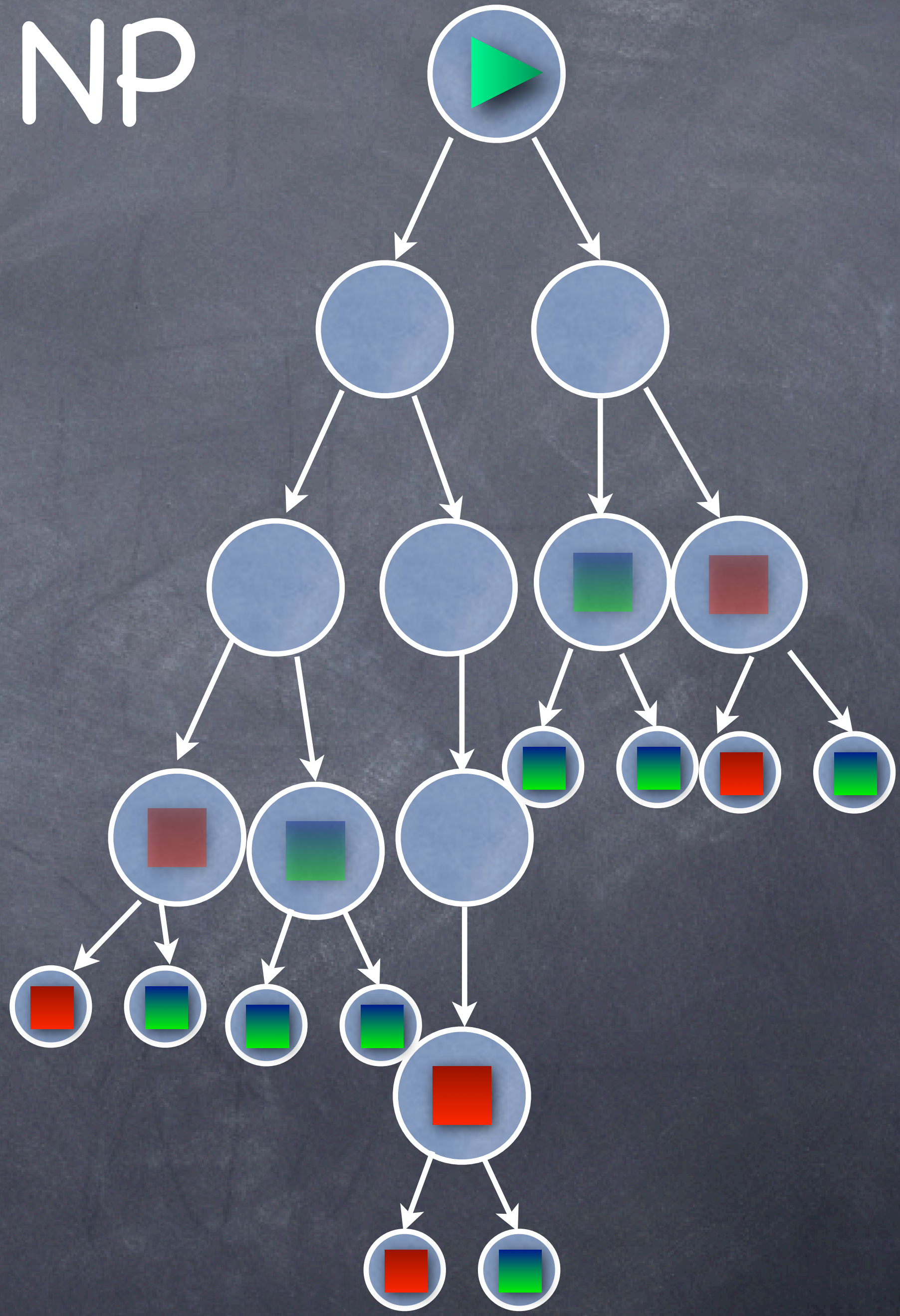
PP and NP

- NP ⊆ PP
  - Use random-tape as non-deterministic choices of NTM M
  - If M rejects, accept with 1/2 prob., else accept



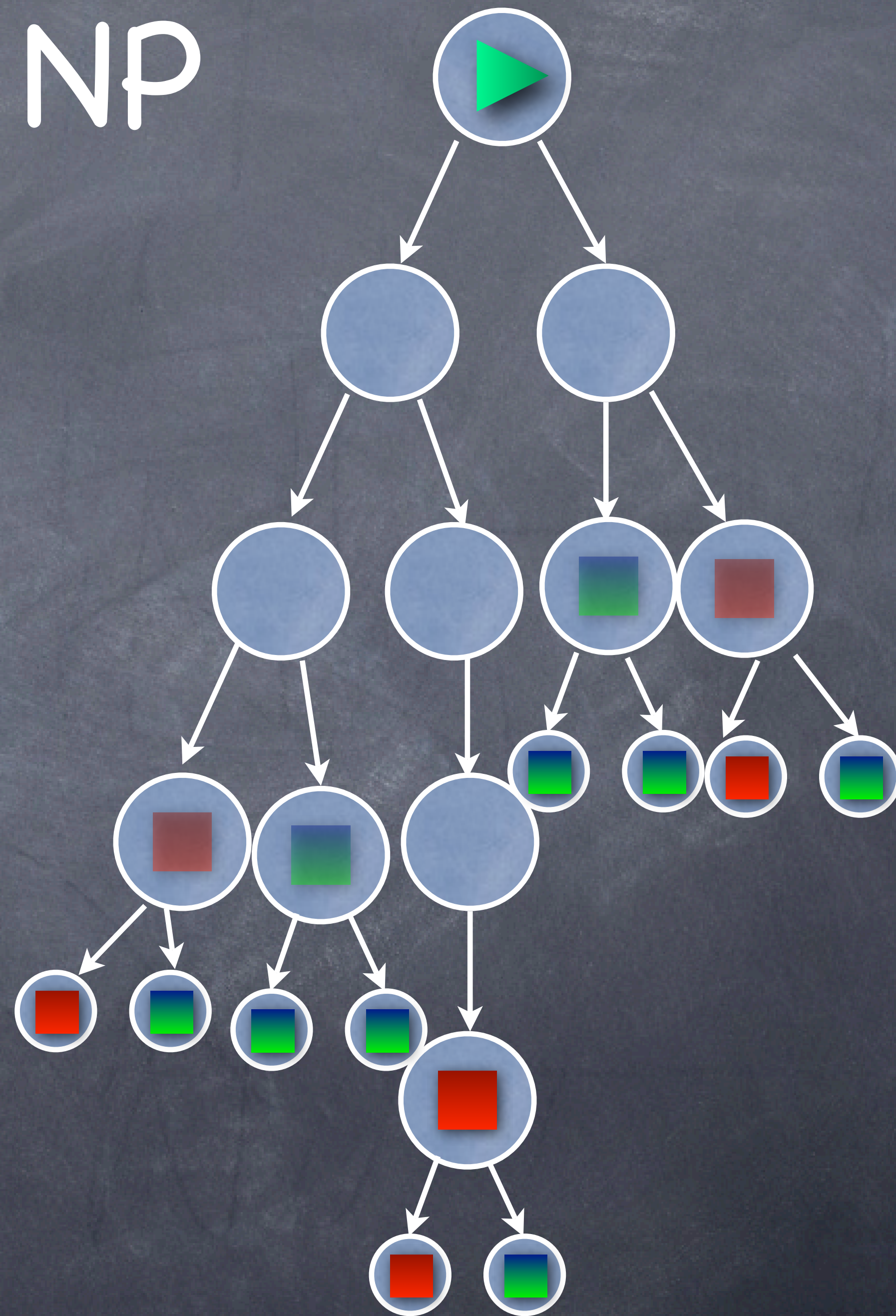
PP and NP

- NP ⊆ PP
  - Use random-tape as non-deterministic choices of NTM M
  - If M rejects, accept with 1/2 prob., else accept
  - If even one thread of M(x) accepts, pr[M(x)=yes] > 1/2



PP and NP

- NP ⊆ PP
  - Use random-tape as non-deterministic choices of NTM M
  - If M rejects, accept with 1/2 prob., else accept
  - If even one thread of M(x) accepts, pr[M(x)=yes] > 1/2
- Accepting gap can be exponentially small



Bounding Probability Gap

Bounding Probability Gap

- Gap

Bounding Probability Gap

- Gap
  - $\text{Min}_{x \in L}$ Pr[M(x)=yes] - $\text{Max}_{x \notin L}$ Pr[M(x)=yes]

Bounding Probability Gap

- Gap
  - $\text{Min}_{x \in L} \Pr[M(x)=\text{yes}] - \text{Max}_{x \notin L} \Pr[M(x)=\text{yes}]$
  - BPP, RP, coRP require M to have gap some constant (1/3, 2/3)

Bounding Probability Gap

- Gap
  - $\text{Min}_{x \in L}$ Pr[M(x)=yes] - $\text{Max}_{x \notin L}$ Pr[M(x)=yes]
  - BPP, RP, coRP require M to have gap some constant (1/3, 2/3)
  - Setting gap = $1/n^c$ is enough

Bounding Probability Gap

- Gap
  - $\text{Min}_{x \in L}$ Pr[M(x)=yes] - $\text{Max}_{x \notin L}$ Pr[M(x)=yes]
  - BPP, RP, coRP require M to have gap some constant (1/3, 2/3)
  - Setting gap = $1/n^c$ is enough
    - Can be boosted to gap = $1 - 1/2^{n^d}$ in polynomial time

Soundness Amplification for RP

Soundness Amplification for RP

- M'(x): Repeat M(x) t times and if any yes, say yes

Soundness Amplification for RP

- M'(x): Repeat M(x) t times and if any yes, say yes
  - If $x \notin L$: Pr[M(x)=no] = 1. So Pr[M'(x)=no] = 1

Soundness Amplification for RP

- M′(x): Repeat M(x) t times and if any yes, say yes
  - If $x \notin L$: Pr[M(x)=no] = 1. So Pr[M′(x)=no] = 1
  - If $x \in L$: Pr[M(x)=no] $\leq 1-\delta$ (when gap = δ). Then Pr[M′(x)=no] $\leq (1-\delta)^t$

Soundness Amplification for RP

- M′(x): Repeat M(x) t times and if any yes, say yes
  - If $x \notin L$: Pr[M(x)=no] = 1. So Pr[M′(x)=no] = 1
  - If $x \in L$: Pr[M(x)=no] $\leq 1-\delta$ (when gap = δ). Then Pr[M′(x)=no] $\leq (1-\delta)^t$
  - With $t = n^d/\delta$, Pr[M′(x)=no] $< e^{-(n^d)}$ $\quad (1-\delta < e^{-\delta})$

Soundness Amplification for RP

- M′(x): Repeat M(x) t times and if any yes, say yes
  - If $x \notin L$: Pr[M(x)=no] = 1. So Pr[M′(x)=no] = 1
  - If $x \in L$: Pr[M(x)=no] $\leq 1-\delta$ (when gap = δ). Then Pr[M′(x)=no] $\leq (1-\delta)^t$
  - With $t = n^d/\delta$, Pr[M′(x)=no] $< e^{-(n\hat{}d)}$ $(1-\delta < e^{-\delta})$
    - For $\delta = n^{-c}$, $t=n^{d+c}$ is polynomial

Soundness Amplification for BPP

Soundness Amplification for BPP

- Repeat M(x) t times and take majority

Soundness Amplification for BPP

- Repeat M(x) t times and take majority
  - i.e. estimate Pr[M(x)=yes] and check if it is > 1/2

Soundness Amplification for BPP

- Repeat M(x) t times and take majority
  - i.e. estimate Pr[M(x)=yes] and check if it is > 1/2
  - Error only if |estimate - real| ≥ gap/2

Soundness Amplification for BPP

- Repeat M(x) t times and take majority
  - i.e. estimate Pr[M(x)=yes] and check if it is > 1/2
  - Error only if |estimate - real| ≥ gap/2
  - Estimation error goes down exponentially with t: Chernoff bound

Soundness Amplification for BPP

- Repeat M(x) t times and take majority
  - i.e. estimate Pr[M(x)=yes] and check if it is > 1/2
  - Error only if |estimate - real| ≥ gap/2
  - Estimation error goes down exponentially with t: Chernoff bound
    - Pr[|estimate - real| ≥ $\delta/2$] ≤ $2^{-\Omega(t.\delta\hat{}2)}$

Soundness Amplification for BPP

- Repeat M(x) t times and take majority
  - i.e. estimate Pr[M(x)=yes] and check if it is > 1/2
  - Error only if |estimate - real| ≥ gap/2
  - Estimation error goes down exponentially with t: Chernoff bound
    - Pr[|estimate - real| ≥ δ/2] ≤ $2^{-\Omega(t.\delta^2)}$
      - t = $O(n^d/\delta^2)$ enough for Pr[error] ≤ 2^{-n^d}

Today

Today

- Probabilistic computation

Today

- Probabilistic computation
- PP, RP, co-RP, BPP

Today

- Probabilistic computation
- PP, RP, co-RP, BPP
  - PP too powerful: NP $\subseteq$ PP

Today

- Probabilistic computation
- PP, RP, co-RP, BPP
    - PP too powerful: NP ⊆ PP
    - Constant gap: BPP, RP, co-RP

Today

- Probabilistic computation
- PP, RP, co-RP, BPP
  - PP too powerful: NP $\subseteq$ PP
  - Constant gap: BPP, RP, co-RP
    - RP, co-RP one-sided error

Today

- Probabilistic computation
- PP, RP, co-RP, BPP
    - PP too powerful: NP $\subseteq$ PP
    - Constant gap: BPP, RP, co-RP
        - RP, co-RP one-sided error
    - Soundness Amplification: for RP, for BPP

Today

- Probabilistic computation
- PP, RP, co-RP, BPP
  - PP too powerful: NP $\subseteq$ PP
  - Constant gap: BPP, RP, co-RP
    - RP, co-RP one-sided error
  - Soundness Amplification: for RP, for BPP
    - From gap 1/poly to 1-1/exp

Today

- Probabilistic computation
- PP, RP, co-RP, BPP
  - PP too powerful: NP $\subseteq$ PP
  - Constant gap: BPP, RP, co-RP
    - RP, co-RP one-sided error
  - Soundness Amplification: for RP, for BPP
    - From gap 1/poly to 1-1/exp
- Next: more on BPP and relatives