COMS21400 : Space Complexity and Beyond

G. Fuchsbauer

Dept of Computer Science
University of Bristol,
Room 3.53, Merchant Venturers Building

04–11 Dec 2012

Outline

Space Complexity

Logarithmic Space

Separations

Randomised Computation

Space complexity for TMs

Definition. Let M be a TM which halts on every input. The **space complexity** of M is $f : \mathbb{N} \to \mathbb{N}$, where $f(n)$ is the maximum number of tape cells that M scans for any input of length n.

Space complexity for TMs

Definition. Let M be a TM which halts on every input. The **space complexity** of M is $f : \mathbb{N} \to \mathbb{N}$, where $f(n)$ is the maximum number of tape cells that M scans for any input of length n.

Definition. Let N be a **N**TM which halts on every input. The space complexity of N is $f : \mathbb{N} \to \mathbb{N}$, where $f(n)$ is the maximum number of tape cells that N scans **on any computation branch** for any input of length n.

Space complexity for TMs

Definition. Let M be a TM which halts on every input. The **space complexity** of M is $f : \mathbb{N} \to \mathbb{N}$, where $f(n)$ is the maximum number of tape cells that M scans for any input of length n.

Definition. Let N be a **N**TM which halts on every input. The space complexity of N is $f : \mathbb{N} \to \mathbb{N}$, where $f(n)$ is the maximum number of tape cells that N scans **on any computation branch** for any input of length n.

Definition. Let $f : \mathbb{N} \to \mathbb{R}^+$. SPACE$(f(n))$ is the class of all languages decided by an $O(f(n))$-space TM.

Space complexity for TMs

Definition. Let M be a TM which halts on every input. The **space complexity** of M is $f : \mathbb{N} \to \mathbb{N}$, where $f(n)$ is the maximum number of tape cells that M scans for any input of length n.

Definition. Let N be a **N**TM which halts on every input. The space complexity of N is $f : \mathbb{N} \to \mathbb{N}$, where $f(n)$ is the maximum number of tape cells that N scans **on any computation branch** for any input of length n.

Definition. Let $f : \mathbb{N} \to \mathbb{R}^+$. SPACE($f(n)$) is the class of all languages decided by an $O(f(n))$-space TM.

Definition. Let $f : \mathbb{N} \to \mathbb{R}^+$. **N**SPACE($f(n)$) is the class of all languages decided by an $O(f(n))$-space **N**TM.

Space complexity classes

Definition. $\text{PSPACE} := \bigcup_k \text{SPACE}(n^k)$ $\qquad$ $\text{NPSPACE} := \bigcup_k \text{NSPACE}(n^k)$

Space complexity classes

Definition. $\mathrm{PSPACE} := \bigcup_k \mathrm{SPACE}(n^k)$ $\qquad$ $\mathrm{NPSPACE} := \bigcup_k \mathrm{NSPACE}(n^k)$

Example. $SAT \in \mathrm{SPACE}(n)$.

Space complexity classes

Definition. PSPACE $:= \bigcup_k \text{SPACE}(n^k)$ NPSPACE $:= \bigcup_k \text{NSPACE}(n^k)$

Example. $SAT \in \text{SPACE}(n)$.

Theorem. For any $f : \mathbb{N} \to \mathbb{N}$ with $f(n) \geq n$:

$$\text{SPACE}(f(n)) \subseteq \text{TIME}(2^{O(f(n))})$$

Space complexity classes

Definition. $\mathrm{PSPACE} := \bigcup_k \mathrm{SPACE}(n^k)$ $\qquad$ $\mathrm{NPSPACE} := \bigcup_k \mathrm{NSPACE}(n^k)$

Example. $SAT \in \mathrm{SPACE}(n)$.

Theorem. For any $f : \mathbb{N} \to \mathbb{N}$ with $f(n) \geq n$:

$$\mathrm{SPACE}(f(n)) \subseteq \mathrm{TIME}(2^{O(f(n))})$$

$\big($Recall: $\mathrm{NTIME}(f(n)) \subseteq \mathrm{TIME}(2^{O(f(n))})\big)$

Space complexity classes

Definition. $\mathrm{PSPACE} := \bigcup_k \mathrm{SPACE}(n^k)$ $\qquad$ $\mathrm{NPSPACE} := \bigcup_k \mathrm{NSPACE}(n^k)$

Example. $SAT \in \mathrm{SPACE}(n)$.

Theorem. For any $f : \mathbb{N} \to \mathbb{N}$ with $f(n) \geq n$:

$$\mathrm{SPACE}(f(n)) \subseteq \mathrm{TIME}(2^{O(f(n))})$$

(Recall: $\mathrm{NTIME}(f(n)) \subseteq \mathrm{TIME}(2^{O(f(n))})$)

Theorem. *(Savitch)* For any $f : \mathbb{N} \to \mathbb{N}$ with $f(n) \geq n$ we have

$$\mathrm{NSPACE}(f(n)) \subseteq \mathrm{SPACE}(f^2(n))$$

Space complexity classes

Definition. PSPACE $:= \bigcup_k \mathrm{SPACE}(n^k)$ NPSPACE $:= \bigcup_k \mathrm{NSPACE}(n^k)$

Example. $SAT \in \mathrm{SPACE}(n)$.

Theorem. For any $f : \mathbb{N} \to \mathbb{N}$ with $f(n) \geq n$:

$$\mathrm{SPACE}(f(n)) \subseteq \mathrm{TIME}(2^{O(f(n))})$$

(Recall: $\mathrm{NTIME}(f(n)) \subseteq \mathrm{TIME}(2^{O(f(n))})$)

Theorem. *(Savitch)* For any $f : \mathbb{N} \to \mathbb{N}$ with $f(n) \geq n$ we have

$$\mathrm{NSPACE}(f(n)) \subseteq \mathrm{SPACE}(f^2(n))$$

$$\boxed{\mathrm{P} \subseteq \mathrm{NP} \subseteq \mathrm{PSPACE} = \mathrm{NPSPACE} \subseteq \mathrm{EXP}}$$

PSPACE-completeness

Definition. A language B is **PSPACE-complete** if

- $B \in$ PSPACE, and
- every A in PSPACE is polynomial-time reducible to B

PSPACE-completeness

Definition. A language B is **PSPACE-complete** if

- $B \in$ PSPACE, and
- every A in PSPACE is polynomial-time reducible to B

Quantified formulas

- Quantifiers:
  - $\forall$: for all
  - $\exists$: there exists
- Let $\phi(x_1, \ldots, x_n)$ be a Boolean formula.
- A **totally quantified** Boolean formula has a quantifier for every variable at the beginning.

PSPACE-completeness

Definition. A language B is **PSPACE-complete** if

- $B \in$ PSPACE, and
- every A in PSPACE is polynomial-time reducible to B

Quantified formulas

- Quantifiers:
  - $\forall$: for all
  - $\exists$: there exists
- Let $\phi(x_1, \ldots, x_n)$ be a Boolean formula.
- A **totally quantified** Boolean formula has a quantifier for every variable at the beginning.

$$\mathit{TQBF} := \{\langle\psi\rangle \mid \psi \text{ is a true totally quantified Boolean formula}\}$$

PSPACE-completeness

Definition. A language B is **PSPACE-complete** if

- $B \in$ PSPACE, and
- every A in PSPACE is polynomial-time reducible to B

Quantified formulas

- Quantifiers:
  - $\forall$: for all
  - $\exists$: there exists
- Let $\phi(x_1, \ldots, x_n)$ be a Boolean formula.
- A **totally quantified** Boolean formula has a quantifier for every variable at the beginning.

$$TQBF := \{\langle\psi\rangle \mid \psi \text{ is a true totally quantified Boolean formula}\}$$

Theorem. $TQBF$ is PSPACE-complete.

Outline

Space Complexity

Logarithmic Space

Separations

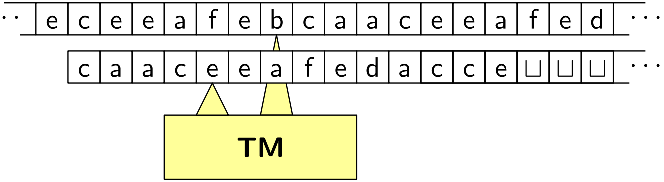
Randomised Computation

Sublinear Space

Sublinear space? Space complexity $f(n) < n =$ input size

Sublinear Space

Sublinear space? Space complexity $f(n) < n =$ input size

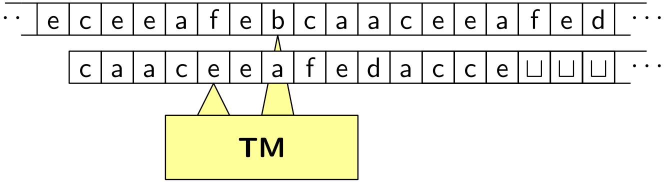


Space-bounded TM Two-tape TM

- Input tape is *read only*
- Work tape

Sublinear Space

Sublinear space? Space complexity $f(n) < n =$ input size



Space-bounded TM Two-tape TM

- Input tape is *read only*
- Work tape

The **space complexity** is defined by the number of cells scanned on the *work tape only*

Logarithmic space

Definition. $\mathbf{L} = \mathrm{SPACE}(\log n)$
$\mathbf{NL} = \mathrm{NSPACE}(\log n)$

Logarithmic space

Definition. $\mathbf{L} = \mathrm{SPACE}(\log n)$
$\mathbf{NL} = \mathrm{NSPACE}(\log n)$

Example. $\mathit{PATH} \in \mathrm{NL}$

Logarithmic space

Definition. $\mathbf{L} = \text{SPACE}(\log n)$
$\mathbf{NL} = \text{NSPACE}(\log n)$

Example. $PATH \in \text{NL}$

More results.

- NL-completeness: defined via log-space reductions

Logarithmic space

Definition. $\mathbf{L} = \text{SPACE}(\log n)$
$\mathbf{NL} = \text{NSPACE}(\log n)$

Example. $PATH \in \text{NL}$

More results.

- NL-completeness: defined via log-space reductions
- *PATH* is NL-complete

Logarithmic space

Definition. $\mathbf{L} = \text{SPACE}(\log n)$
$\mathbf{NL} = \text{NSPACE}(\log n)$

Example. $PATH \in \text{NL}$

More results.

- NL-completeness: defined via log-space reductions
- *PATH* is NL-complete
- NL = co-NL

Logarithmic space

Definition. $\mathbf{L} = \text{SPACE}(\log n)$
$\mathbf{NL} = \text{NSPACE}(\log n)$

Example. $PATH \in \text{NL}$

More results.

- NL-completeness: defined via log-space reductions
- *PATH* is NL-complete
- NL = co-NL

$$\boxed{\text{L} \subseteq \text{NL} = \text{co-NL} \subseteq \text{P} \subseteq \text{NP} \subseteq \text{PSPACE} = \text{NPSPACE} \subseteq \text{EXP}}$$

Outline

Space Complexity

Logarithmic Space

Separations

Randomised Computation

Space hierarchy

- Can we compute *more* when given more time/space?
- Could it be that all encountered classes are the *same*?

Space hierarchy

- Can we compute *more* when given more time/space?
- Could it be that all encountered classes are the *same*?

Theorem. *(Space hierarchy)* For any space-constructible $f : \mathbb{N} \to \mathbb{N}$, there exists a language A that is decidable in $O(f(n))$ space but not in $o(f(n))$ space.

Space hierarchy

- Can we compute *more* when given more time/space?
- Could it be that all encountered classes are the *same*?

Theorem. *(Space hierarchy)* For any space-constructible $f : \mathbb{N} \to \mathbb{N}$, there exists a language A that is decidable in $O(f(n))$ space but not in $o(f(n))$ space.

- $\mathrm{SPACE}(n^{\varepsilon_1}) \subsetneq \mathrm{SPACE}(n^{\varepsilon_2})$, for $0 \leq \varepsilon_1 < \varepsilon_2$

Space hierarchy

- Can we compute *more* when given more time/space?
- Could it be that all encountered classes are the *same*?

Theorem. *(Space hierarchy)* For any space-constructible $f : \mathbb{N} \to \mathbb{N}$, there exists a language A that is decidable in $O(f(n))$ space but not in $o(f(n))$ space.

- $\mathrm{SPACE}(n^{\varepsilon_1}) \subsetneq \mathrm{SPACE}(n^{\varepsilon_2})$, for $0 \leq \varepsilon_1 < \varepsilon_2$
- $\mathrm{NL} \subsetneq \mathrm{PSPACE}$

Space hierarchy

- Can we compute *more* when given more time/space?
- Could it be that all encountered classes are the *same*?

Theorem. *(Space hierarchy)* For any space-constructible $f : \mathbb{N} \to \mathbb{N}$, there exists a language A that is decidable in $O(f(n))$ space but not in $o(f(n))$ space.

- $\text{SPACE}(n^{\varepsilon_1}) \subsetneq \text{SPACE}(n^{\varepsilon_2})$, for $0 \leq \varepsilon_1 < \varepsilon_2$
- $\text{NL} \subsetneq \text{PSPACE}$
- $\text{PSPACE} \subsetneq \text{EXPSPACE} := \bigcup_k \text{SPACE}(2^{n^k})$

Time hierarchy

Theorem. *(Time hierarchy)* For any time-constructible $t : \mathbb{N} \to \mathbb{N}$, there exists a language A that is decidable in $O(t(n))$ time but not in $o\left(\frac{t(n)}{\log t(n)}\right)$ time.

Time hierarchy

Theorem. *(Time hierarchy)* For any time-constructible $t : \mathbb{N} \to \mathbb{N}$, there exists a language A that is decidable in $O(t(n))$ time but not in $o\left(\frac{t(n)}{\log t(n)}\right)$ time.

- $\mathrm{TIME}(n^{\varepsilon_1}) \subsetneq \mathrm{TIME}(n^{\varepsilon_2})$, for $1 \leq \varepsilon_1 < \varepsilon_2$

Time hierarchy

Theorem. *(Time hierarchy)* For any time-constructible $t : \mathbb{N} \to \mathbb{N}$, there exists a language A that is decidable in $O(t(n))$ time but not in $o\left(\frac{t(n)}{\log t(n)}\right)$ time.

- $\mathrm{TIME}(n^{\varepsilon_1}) \subsetneq \mathrm{TIME}(n^{\varepsilon_2})$, for $1 \leq \varepsilon_1 < \varepsilon_2$
- $\mathrm{P} \subsetneq \mathrm{EXP}$

Time hierarchy

Theorem. *(Time hierarchy)* For any time-constructible $t : \mathbb{N} \to \mathbb{N}$, there exists a language A that is decidable in $O(t(n))$ time but not in $o\left(\frac{t(n)}{\log t(n)}\right)$ time.

- $\mathrm{TIME}(n^{\varepsilon_1}) \subsetneq \mathrm{TIME}(n^{\varepsilon_2})$, for $1 \leq \varepsilon_1 < \varepsilon_2$
- $\mathrm{P} \subsetneq \mathrm{EXP}$

$$\mathrm{L} \subseteq \mathrm{NL} \subseteq \mathrm{P} \subseteq \mathrm{NP} \subseteq \mathrm{PSPACE} \subseteq \mathrm{EXP} \subseteq \mathrm{EXPSPACE}$$

Time hierarchy

Theorem. *(Time hierarchy)* For any time-constructible $t : \mathbb{N} \to \mathbb{N}$, there exists a language A that is decidable in $O(t(n))$ time but not in $o\left(\frac{t(n)}{\log t(n)}\right)$ time.

- $\mathrm{TIME}(n^{\varepsilon_1}) \subsetneq \mathrm{TIME}(n^{\varepsilon_2})$, for $1 \leq \varepsilon_1 < \varepsilon_2$
- $\mathrm{P} \subsetneq \mathrm{EXP}$

$$\boxed{\mathrm{L} \subseteq \mathrm{NL} \subseteq \overbrace{\mathrm{P} \subseteq \mathrm{NP} \subseteq \mathrm{PSPACE} \subseteq \mathrm{EXP}}^{\neq} \subseteq \mathrm{EXPSPACE}}$$

Time hierarchy

Theorem. *(Time hierarchy)* For any time-constructible $t : \mathbb{N} \to \mathbb{N}$, there exists a language A that is decidable in $O(t(n))$ time but not in $o\left(\frac{t(n)}{\log t(n)}\right)$ time.

- $\mathrm{TIME}(n^{\varepsilon_1}) \subsetneq \mathrm{TIME}(n^{\varepsilon_2})$, for $1 \leq \varepsilon_1 < \varepsilon_2$
- $\mathrm{P} \subsetneq \mathrm{EXP}$

$$\boxed{\mathrm{L} \subseteq \mathrm{NL} \subseteq \mathrm{P} \subseteq \mathrm{NP} \subseteq \mathrm{PSPACE} \subseteq \mathrm{EXP} \subseteq \mathrm{EXPSPACE}}$$

$\neq$

$\neq$ $\neq$

Outline

Space Complexity

Logarithmic Space

Separations

Randomised Computation

Randomised Computation

- Allow a TM to make **random** choices of the next step

Randomised Computation

- Allow a TM to make **random** choices of the next step

Example. *(Polynomial identities)*

- Given: $Q(x_1, \ldots, x_n)$, a polynomial in n variables.
- Decide: Is Q identically zero?

Randomised Computation

- Allow a TM to make **random** choices of the next step

Example. *(Polynomial identities)*

- Given: $Q(x_1, \ldots, x_n)$, a polynomial in n variables.
- Decide: Is Q identically zero?

Fact. Let $Q(x_1, \ldots, x_n)$ have degree $\leq d$ in every variable and Q not identically zero.
Then for any set S of values, with $|S| \geq 2nd$, the number of tuples $(a_1, \ldots, a_n) \in S^n$ s.t. $Q(a_1, \ldots, a_n) = 0$, is at most $\frac{1}{2}|S|^n$.

Checking $Q(x_1, \ldots, x_n)$ is identically zero:

$R =$ "On input Q

1. Choose S with $|S| > 2nd$.
2. Choose $(a_1, \ldots, a_n)$ at random from $S \times \ldots \times S$
3. If $Q(a_1, \ldots, a_n) \neq 0$ output *reject*

   Otherwise output *accept*."

Checking $Q(x_1, \ldots, x_n)$ is identically zero:

$R =$ "On input Q

1. Choose S with $|S| > 2nd$.
2. Choose $(a_1, \ldots, a_n)$ at random from $S \times \ldots \times S$
3. If $Q(a_1, \ldots, a_n) \neq 0$ output *reject*

   Otherwise output *accept*."

If Q **is zero** then R outputs *accept* with probability 1

If Q **is not zero** then R outputs *accept* with probability $\leq 1/2$

Checking $Q(x_1, \ldots, x_n)$ is identically zero:

$R =$ "On input Q

1. Choose S with $|S| > 2nd$.
2. Choose $(a_1, \ldots, a_n)$ at random from $S \times \ldots \times S$
3. If $Q(a_1, \ldots, a_n) \neq 0$ output *reject*

   Otherwise output *accept*."

If Q **is zero** then R outputs *accept* with probability 1

If Q **is not zero** then R outputs *accept* with probability $\leq 1/2$

Amplification. Repeat Steps 2 and 3 k times

Checking $Q(x_1, \ldots, x_n)$ is identically zero:

$R =$ "On input Q

1. Choose S with $|S| > 2nd$.
2. Choose $(a_1, \ldots, a_n)$ at random from $S \times \ldots \times S$
3. If $Q(a_1, \ldots, a_n) \neq 0$ output *reject*
   Otherwise output *accept*."

If Q **is zero** then R outputs *accept* with probability 1

If Q **is not zero** then R outputs *accept* with probability $\leq 1/2$

Amplification. Repeat Steps 2 and 3 k times

...

If Q **is not zero** then R outputs *accept* with probability $\leq 1/(2^k)$

Probabilistic TMs

Definition. A **probabilistic Turing machine** is a NTM in which each non-deterministic step ("coin flip") has *two* legal moves.

Probabilistic TMs

Definition. A **probabilistic Turing machine** is a NTM in which each non-deterministic step ("coin flip") has *two* legal moves.

- For every computation branch b, let k be the number of coin flips on b. Then
$$\Pr[b] := 1/2^k$$

Probabilistic TMs

Definition. A **probabilistic Turing machine** is a NTM in which each non-deterministic step ("coin flip") has *two* legal moves.

- For every computation branch b, let k be the number of coin flips on b. Then

$$\Pr[b] := 1/2^k$$

- Accepting probability:

$$\Pr[M \text{ accepts } w] := \sum_{b \text{ is an accepting branch}} \Pr[b]$$

Probabilistic TMs

Definition. A **probabilistic Turing machine** is a NTM in which each non-deterministic step ("coin flip") has *two* legal moves.

- For every computation branch b, let k be the number of coin flips on b. Then
$$\Pr[b] := 1/2^k$$
- Accepting probability:
$$\Pr[M \text{ accepts } w] := \sum_{b \text{ is an accepting branch}} \Pr[b]$$
- Rejecting probability: $1 - \Pr[M \text{ accepts } w]$

Probabilistic TMs

Definition. A **probabilistic Turing machine** is a NTM in which each non-deterministic step ("coin flip") has *two* legal moves.

- For every computation branch b, let k be the number of coin flips on b. Then

$$\Pr[b] := 1/2^k$$

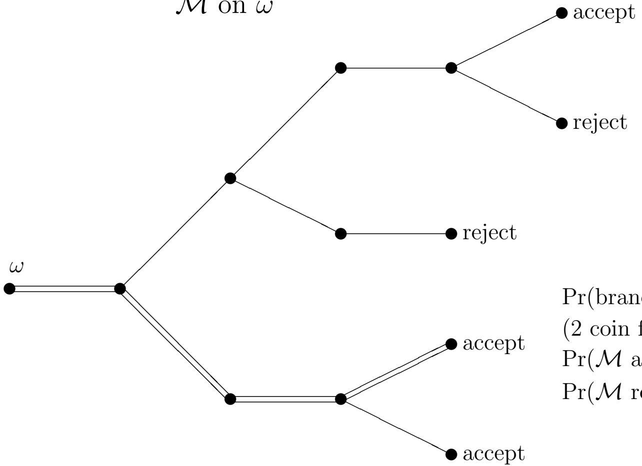
- Accepting probability:

$$\Pr[M \text{ accepts } w] := \sum_{b \text{ is an accepting branch}} \Pr[b]$$

- Rejecting probability: $1 - \Pr[M \text{ accepts } w]$

PTMs are real devices, NTMs are not

$\mathcal{M}$ on ω



$\Pr(\text{branch } =\!=\!=) = \frac{1}{4}$
(2 coin flips on this branch)
$\Pr(\mathcal{M} \text{ acc. } \omega) = \frac{1}{8} + \frac{1}{4} + \frac{1}{4} = \frac{5}{8}$
$\Pr(\mathcal{M} \text{ rej. } \omega) = \frac{1}{8} + \frac{1}{4} = \frac{3}{8}$

The class BPP

Definition. A TM M recognises L with **error probability** ε if

- $w \in L \;\Rightarrow\;$ Pr[M accepts w] $\geq 1 - \varepsilon$; and
- $w \notin L \;\Rightarrow\;$ Pr[M rejects w] $\geq 1 - \varepsilon$

The class BPP

Definition. A TM M recognises L with **error probability** ε if

- $w \in L \;\Rightarrow\; \Pr[M \text{ accepts } w] \geq 1 - \varepsilon$; and
- $w \notin L \;\Rightarrow\; \Pr[M \text{ rejects } w] \geq 1 - \varepsilon$

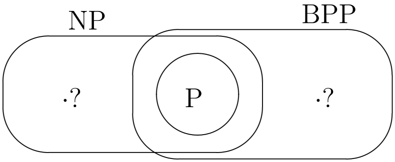
Definition. **BPP** (bounded-error probabilistic polynomial time) is the class of languages that are recognised by a polynomial-time PTM with error probability 1/3.

The class BPP

Definition. A TM M recognises L with **error probability** ε if

- $w \in L \;\Rightarrow\; \Pr[M \text{ accepts } w] \geq 1 - \varepsilon$; and
- $w \notin L \;\Rightarrow\; \Pr[M \text{ rejects } w] \geq 1 - \varepsilon$

Definition. **BPP** (bounded-error probabilistic polynomial time) is the class of languages that are recognised by a polynomial-time PTM with error probability 1/3.

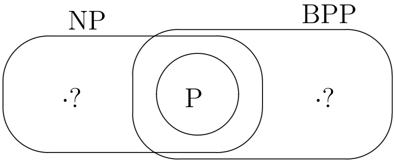


The class BPP

Definition. A TM M recognises L with **error probability** ε if

- $w \in L \;\Rightarrow\; \Pr[M \text{ accepts } w] \geq 1 - \varepsilon$; and
- $w \notin L \;\Rightarrow\; \Pr[M \text{ rejects } w] \geq 1 - \varepsilon$

Definition. **BPP** (bounded-error probabilistic polynomial time) is the class of languages that are recognised by a polynomial-time PTM with error probability 1/3.



Lemma. *(Amplification)* Let $0 < \varepsilon_1 \leq \varepsilon_2 < 1/2$.
If L can be recognised by a poly-time PTM with error probability ε_2 then it can be recognised by a poly-time PTM with error prob. ε_1.

The class RP

Definition. **RP** (randomised polynomial time) is the class of languages for which there is a poly-time PTM with

- $w \in L \;\Rightarrow\;$ Pr[M accepts w] $\geq 1/2$; and
- $w \notin L \;\Rightarrow\;$ Pr[M rejects w] $= 1$

The class RP

Definition. **RP** (randomised polynomial time) is the class of languages for which there is a poly-time PTM with

- $w \in L \;\Rightarrow\; \Pr[M \text{ accepts } w] \geq 1/2$; and
- $w \notin L \;\Rightarrow\; \Pr[M \text{ rejects } w] = 1$

If M accepts, we know $w \in L$ *("one-sided error")*

The class RP

Definition. **RP** (randomised polynomial time) is the class of languages for which there is a poly-time PTM with

- $w \in L \;\Rightarrow\; \Pr[M \text{ accepts } w] \geq 1/2$; and
- $w \notin L \;\Rightarrow\; \Pr[M \text{ rejects } w] = 1$

If M accepts, we know $w \in L$ *("one-sided error")*

Example. The Fermat primality test is given p s.t.

- p is prime $\;\Rightarrow\; \Pr[M \text{ accepts } p] = 1$
- p is composite $\;\Rightarrow\; \Pr[M \text{ accepts } p] \leq 1/2^k$

The class RP

Definition. **RP** (randomised polynomial time) is the class of languages for which there is a poly-time PTM with

- $w \in L \;\Rightarrow\; \Pr[M \text{ accepts } w] \geq 1/2$; and
- $w \notin L \;\Rightarrow\; \Pr[M \text{ rejects } w] = 1$

If M accepts, we know $w \in L$ *("one-sided error")*

Example. The Fermat primality test is given p s.t.

- p is prime $\Rightarrow\; \Pr[M \text{ accepts } p] = 1$
- p is composite $\Rightarrow\; \Pr[M \text{ accepts } p] \leq 1/2^k$

Thus $\mathit{COMPOSITES} \in \mathrm{RP}$

Fin