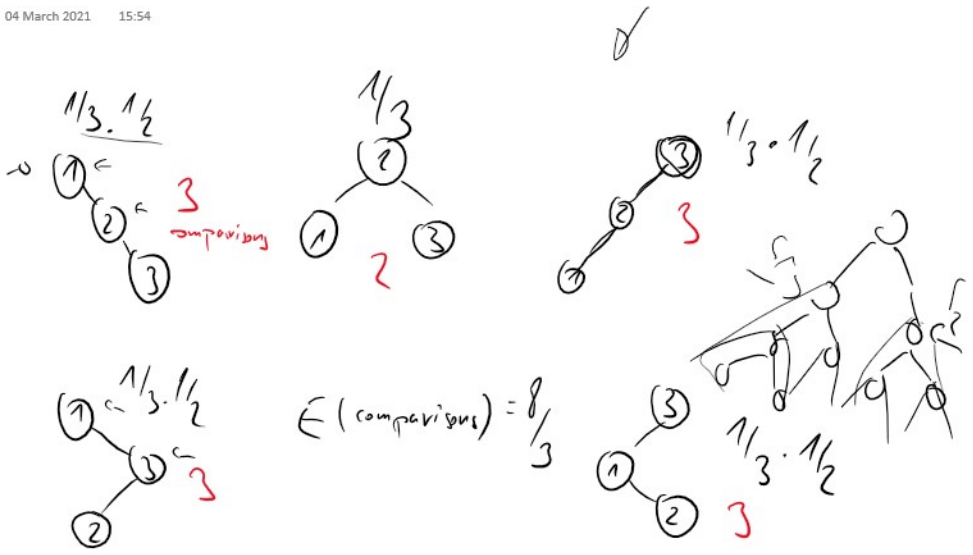
Tutorial II

04 March 2021 15:54



Expected number of steps is $O(n \cdot \log n)$ where n is the size of the set to be sorted.

$\forall i,j \in S$ define a random variable S_{ij}.

$S_{ij} = 1$ if i and j are compared in a given run.

(Last step (qsort S_1 | y | qsort S_2). $\forall i \in S_1 \; i < y$, $\forall i \in S_2 \; i > y$

$i < y < j \Rightarrow$ i and j do not get compared)

$S_{ij} = 0$ if i and j do not get compared

$$X = \sum_{i<j} S_{ij}$$ → the total number of comparisons in a run

$\forall i,j \quad Pr(S_{ij} = 1) \leq \frac{1}{j-i+1}$

$$E(X) = E\left(\sum_{i<j} S_{ij}\right) = \sum_{i<j} E(S_{ij}) = \sum_{i<j} Pr(S_{ij} = 1) \leq \sum_{i<j} \frac{1}{j-i+1}$$

$$E(S_{ij}) = 0 \cdot Pr(S_{ij} = 0) + 1 \cdot Pr(S_{ij} = 1) = Pr(S_{ij} = 1)$$

Step 1 What is the probability to compare i and j?

$Pr(i \text{ and } j \text{ getting compared in Step 1}) = \frac{2}{n} = \frac{?}{|S|}$

Step 2 $|S_1|, \not{S}, |S_2|$

Probability of i and j getting compared in step 2?

$Pr(i \text{ and } j \text{ get compared} \mid i < S < j) = 0$

$Pr(i \text{ and } j \text{ get compared} \mid S < i < j) = \frac{2}{|S_2|}$

$Pr(i \text{ and } j \text{ get compared} \mid i < j < S) = \frac{2}{|S_1|}$

Step k → let's bound the probability of comparing i, j conditioned on the event of both i, j being in the same subset $|S_k| \geq j - i + 1$

$i, i+1, \ldots, j$ — $j - i + 1$ elements

$Pr(i \text{ and } j \text{ get compared in round } k \mid \text{they are in set } S_k) = \frac{2}{|S_k|} < \frac{2}{j-i+1}$

$S_{ij} = 1$ if i and j get compared in some round

$Pr(S_{ij} = 1) =$

$= Pr(i \text{ and } j \text{ get compared in round } k \mid \text{they are in the same set}) \cdot Pr(i \text{ and } j \text{ are in the same set in } k\text{-th round})$

law of total probability

$+ Pr(i \text{ and } j \text{ get compared in round } k \mid i \text{ and } j \text{ are not in the same set}) \cdot Pr(i \text{ and } j \text{ are not in the same set}) = 0$

$= Pr(i \text{ and } j \text{ get compared in round } k \mid \text{they are in the same set}) \cdot Pr(i \text{ and } j \text{ are in the same set in } k\text{-th round})$

$< Pr(i \text{ and } j \text{ not compared in round } k \mid \text{they}$

$\leq Pr$ (i and j get compared in round k | they are in the same set)

$\leq \frac{2}{j-i+1}$

$$E(X) \leq \sum_{i<j} \frac{2}{j-i+1} = \ldots = n \cdot \underbrace{\sum_{i=1}^{n} \frac{1}{i}}_{\Theta(\log n)} = \Theta(n \log n)$$

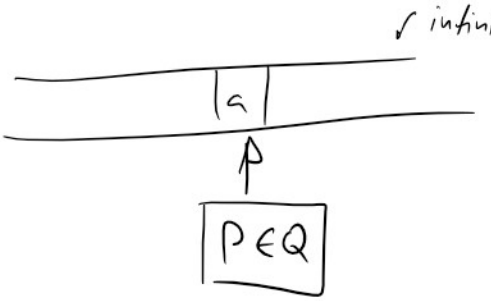
in slides

Classification of randomized algorithms

- Complexity classes
- Algorithm types

Complexity classes

DTM - deterministic TM



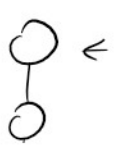
$\delta : (Q \times \Sigma) \to (Q \times \{\leftarrow, \rightarrow, \downarrow\})$

$Acc \subseteq Q$

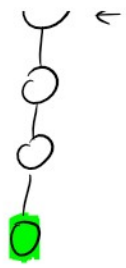
Decision problems

x - input Does x belong to a language L

$L \subseteq \Sigma^*$ (Σ^* a set of all strings over alphabet Σ)



$L \subseteq L$ ([illegible] strings over alphabet Σ)



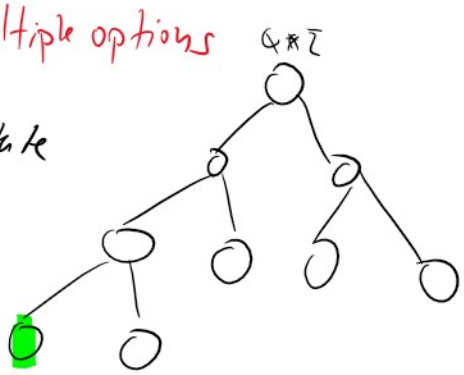
TM ends in accepting state whenever $x \in L$

TM doesn't end in accepting state $x \notin L$

NTM - non-deterministic $(Q \times \Sigma) \times (Q \times \{\leftarrow, \rightarrow, \phi\})$

multiple options

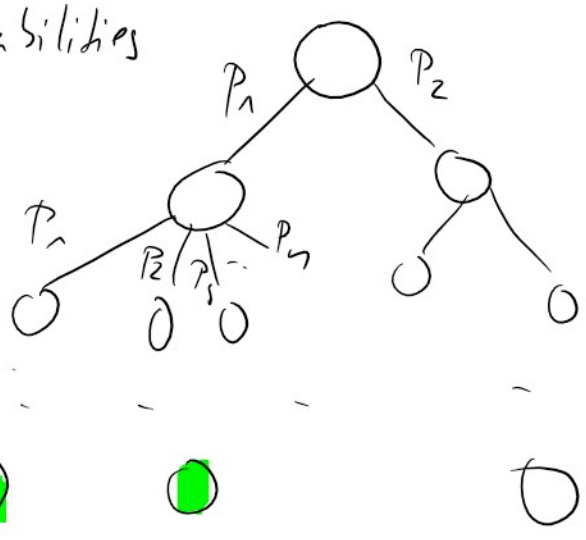
$x \in L \Rightarrow$ } accepting termination state



Probabilistic TM

→ multiple rules for same (state, input)

but these are assigned probabilities



Pr to accept input x is the total probability to end in an accepting state

↗ random polynomial

→ random polynomial

RP contains problems for which there exists a TM (algorithm)

s.t. $x \in L : Pr(TM(x) \text{ accepts}) \geq \frac{1}{2} \geq \varepsilon > 0$

$x \notin L : Pr(TM(x) \text{ accepts}) = 0$

Problems in RP have 1-sided MC-algorithms with NO-bias.

co-RP contains problems for which there exists a TM (algorithm)

s.t. $x \in L : Pr(TM(x) \text{ accepts}) = 1$

$x \notin L : Pr(TM(x) \text{ accepts}) \leq \frac{1}{2} \leq \varepsilon < 1$

Problems in co-RP have 1-sided MC-algorithm with YES-bias

Probability amplification (in RP)

We want to [illegible] a 1-MC with [illegible] polynomial

We want to show that a 1-MC with No-bias and probability of error δ can be used to construct 1-sided MC algorithm with $\varepsilon << \delta$ error

$x \notin L$ if you run the algorithm k-times you will get k answers 'No'

$x \in L$ — " — k-times one answer 'YES' is enough to convince you you have yes instance.

$$Pr\left[x \in L,\ A^k(x) = Y \in S\right] \geq 1 - \delta^k$$

i.e. error of k-fold run is $\varepsilon = \delta^k$

How many repetitions are needed to achieve target error ε.

Solve $\varepsilon \geq \delta^k$

"right" error $\leq$.

Solve $\varepsilon \geq \delta^k$

$$\log \varepsilon \geq \log \delta^k$$

$$\log \varepsilon \geq k \log \delta$$

$$\frac{\log(\varepsilon)}{\log(\delta)} \leq k$$

$$\frac{\log(\varepsilon)}{\log(\delta(n))} \leq k(n)$$

$\frac{1}{\log(\delta(n))}$ is a polynomial in n

$\log(\delta(n))$ is an inverse polynomial

bounded probabilistic polynomial

BPP $\rightarrow$

$x \in L : Pr(TM(x) \text{ accepts}) \geq 3/4$

$x \notin L : P(TM(x) \text{ accepts}) \leq 1/4$

$\Big\}$ arbitrary $\rightarrow \varepsilon > 1/2$

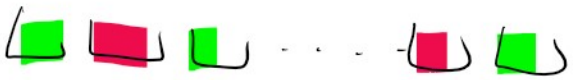
$\rightarrow a < 1/2$

$x \notin L : P(TM(x) \text{ accepts}) < 1/4$) $\rightarrow a < 1/2$

2-sided MC algorithm

Again polynomially many repetitions are enough to get arbitrarily small error → Majority voting

k-repetitions

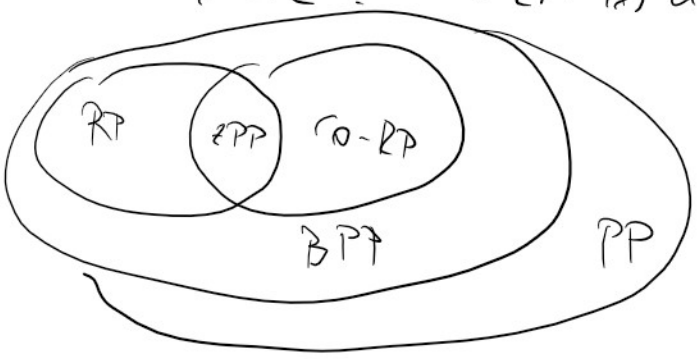


→ Answer = majority of answers

We need Chernoff bounds for an easy proof ⇒ NEXT tutorial

PP - $x \in L : Pr[TM_{(x)} \text{accepts}] > 1/2$ ($1/2 + 1/2^n$)

$x \notin L : Pr[TM_{(x)} \text{ accepts}] \leq 1/2$



$ZPP = co\text{-}RP \cap RP$ → associated to Las Vegas algorithms

(A_y)

0.) Problems in ZPP have both 1-MC with NO-Bias, 1-MC with YES-Bias

⇕

1.) They have LV algorithm of type 1: (A_N)

Always gives a correct answer w.p. 1 and has expected polynomial (in input size) running time

⇕

2.) They have LV algorithm of type 2:

Always polynomial running time and it gives answer "I don't know" w.p. $\leq 1/2$

1.) => 2.) Run type 1 algorithm and if the number of steps exceeds $2E(n)$ then terminate and output "I don't know"

2.) => 1.) Probability amplification

Run LV_2 -> if it says "I don't know"

run it again (with different random choices)

0.) => 1.) Run $A_y(x)$ and if it says 'YES' it is correct answer otherwise run $A_N(x)$ if it says 'NO' it is a correct answer otherwise

2.) => 0.) 1.) Run $LV_2(x)$ if correct answer is found give it as an output

2.) In case of "I don't know"

$A_y(x)$ -> answer NO

$A_N(x)$ -> answer YES