Introduction to Algorithms Analysis

Data Structures and Algorithms (094224)

Tutorial 1

Winter 2022/23

Staff

צוות הקורס

שם	תפקיד	דוא"ל	שעת קבלה	משרד
פרופ'ח יובל עמק	מרצה	yemek@technion.ac.il	יום ד' 08:30-09:30 (בתיאום מראש)	בלומפילד 309
מר יובל גיל	מתרגל אחראי	yuval.gil@campus.technion.ac.il	יום ה' 10:30-11:30	יעודכן בהמשך
גברת נגה הרלב	מתרגלת	snogazur@campus.technion.ac.il	יום ב' 10:30-11:30	קופר 424
מר אורי פרקש	מתרגל	orifa@campus.technion.ac.il	יום ג' 12:30-13:30	ייקבע בזמן אמת
מר אדם גולדבראיך	אחראי תרגילים עיוניים	sgoadam@campus.technion.ac.il	בתיאום מראש	בתיאום מראש
מר ויסאם היגא	אחראי תרגיל תכנותי	hija.wesam@campus.technion.ac.il	בתיאום מראש	בתיאום מראש

Miscellaneous

- Tutorials do not replace lectures. It is highly recommended to attend both
- Tutorials are based on lectures. It is highly recommended to go over last lecture before tutorial
- All tutorial slides are mandatory material, even if not learned in class
- Homework submissions must be uploaded to Moodle. Submission will remain open for an extra 48 hours after official deadline. Any submission that will not be uploaded to Moodle will be graded 0 (exceptions are only according to Technion regulations)
- Email guidelines (email subject must contain the keyword DS_Algs)

1 Asymptotic notation

2 Running time function

3 Insertion sort

Asymptotic Notation

The functions we consider:

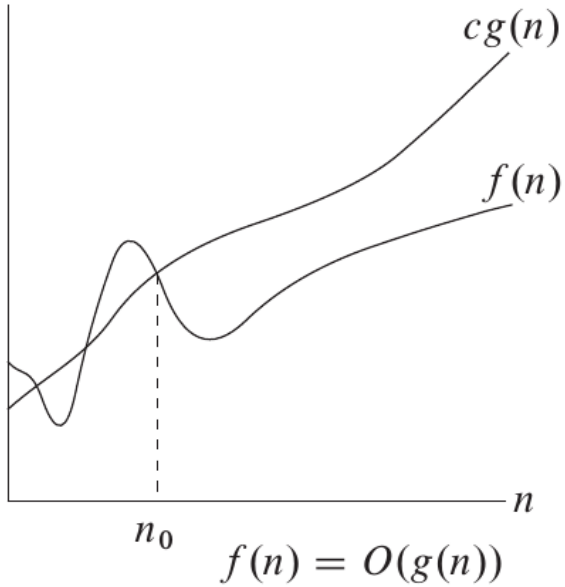
- Domain $= \mathbb{Z}_{\geq 0}$
  - Sometimes: $\mathbb{Z}_{>0}$, $\mathbb{R}_{\geq 0}$, $\mathbb{R}_{>0}$
- Range $= \mathbb{R}$
- Asymptotically positive:
  there exists $n_0 > 0$ such that $f(n) > 0$ for all $n \geq n_0$

Big O notation

Definition (Big O)

Function $f(n)$ belongs to the class $O(g(n))$ if there exist positive constants c and n_0 such that $0 \leq f(n) \leq cg(n)$ for all $n \geq n_0$.

- Interpreted as $f \leq g$
- Often write $f(n) = O(g(n))$ rather than $f(n) \in O(g(n))$

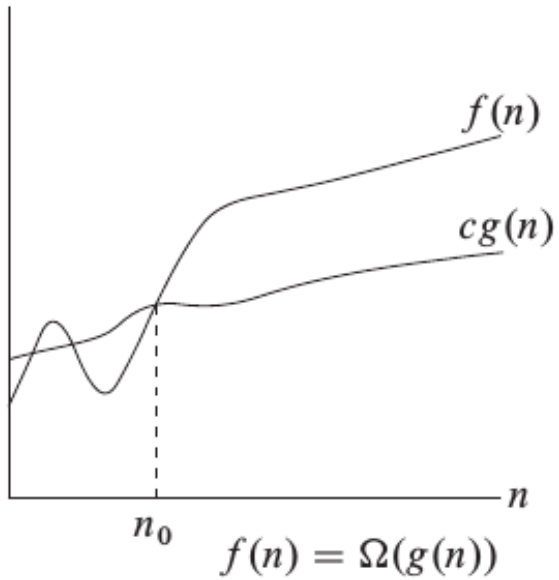


Big Ω notation

Definition (Big Ω)

Function $f(n)$ belongs to the class $\Omega(g(n))$ if there exist positive constants c and n_0 such that $0 \leq cg(n) \leq f(n)$ for all $n \geq n_0$.

- Interpreted as $f \geq g$
- Often write $f(n) = \Omega(g(n))$ rather than $f(n) \in \Omega(g(n))$

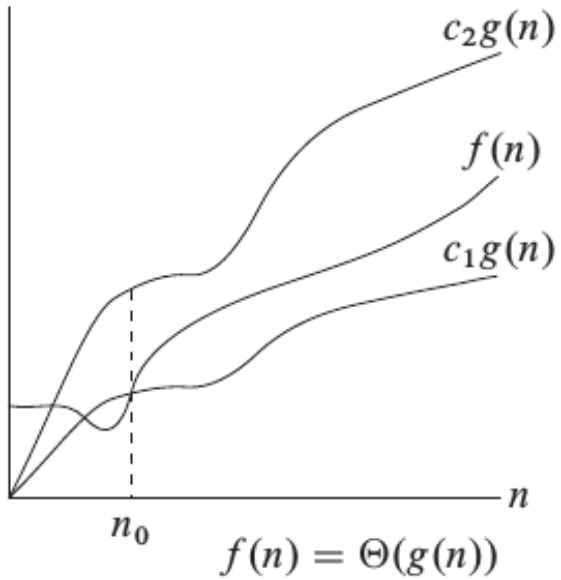


Θ notation

Definition (Θ)

Function $f(n)$ belongs to the class $\Theta(g(n))$ if there exist positive constants c_1, c_2, and n_0 such that $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$ for all $n \geq n_0$.

- Interpreted as $f = g$
- Often write $f(n) = \Theta(g(n))$ rather than $f(n) \in \Theta(g(n))$



Little o notation

Definition (Little o)

Function $f(n)$ belongs to the class $o(g(n))$ if for any constant $c > 0$, there exists a constant $n_0 > 0$ such that $0 \leq f(n) < cg(n)$ for all $n \geq n_0$.

- Interpreted as $f \lesssim g$ (strict inequality)
- Often write $f(n) = o(g(n))$ rather than $f(n) \in o(g(n))$

Little ω notation

Definition (Little ω)

Function $f(n)$ belongs to the class $\omega(g(n))$ if for any constant $c > 0$, there exists a constant $n_0 > 0$ such that $0 \leq cg(n) < f(n)$ for all $n \geq n_0$.

- Interpreted as $f \gneq g$ (strict inequality)
- Often write $f(n) = \omega(g(n))$ rather than $f(n) \in \omega(g(n))$

Asymptotic notation definition using limits

- $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0 \iff f(n) \in o(g(n))$
- $\lim_{n\to\infty} \frac{g(n)}{f(n)} = 0 \iff f(n) \in \omega(g(n))$
- $\lim_{n\to\infty} \frac{g(n)}{f(n)} = c \implies f(n) \in \Theta(g(n))$

Question 1

Prove:
$f(n) = \Theta(g(n)) \iff f(n) = O(g(n)) \wedge f(n) = \Omega(g(n))$

Solution:

- Direction $\Longrightarrow$
- There exist constants $c_1, c_2, n_0 > 0$ such that
  $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$ for all $n \geq n_0$
- We will prove there exist constants $c_3, c_4, n_3, n_4 > 0$ such that
  $0 \leq c_3 g(n) \leq f(n)$ for all $n \geq n_3 \Longrightarrow f(n) = \Omega(g(n))$ and
  $0 \leq f(n) \leq c_4 g(n)$ for all $n \geq n_4 \Longrightarrow f(n) = O(g(n))$
- Set $c_3 = c_1$, $c_4 = c_2$ and $n_3 = n_4 = n_0$ to complete the proof

Solution — cont.

- Direction $\Longleftarrow$
- There exist constants $c_1, c_2, n_1, n_2 > 0$ such that
  $0 \leq c_1 g(n) \leq f(n)$ for all $n \geq n_1$ and
  $0 \leq f(n) \leq c_2 g(n)$ for all $n \geq n_2$
- We will prove there exist constants $c_3, c_4, n_3 > 0$ such that
  $0 \leq c_3 g(n) \leq f(n) \leq c_4 g(n)$ for all $n \geq n_3$
- Set $c_3 = c_1$, $c_4 = c_2$ and $n_3 = \max\{n_1, n_2\}$ to complete the proof

Question 2

Prove/Disprove:

Given that for every $n \geq 2^{100}$ it holds $0 \leq f(n) \leq g(n) \leq h(n)$ and $f(n) = \Omega(h(n))$ then $f(n) = \Omega(g(n))$

Solution (the claim is true):

- $f(n) = \Omega(h(n)) \Longrightarrow$ there exist constants $c, n_0 > 0$ such that $0 \leq ch(n) \leq f(n)$ for every $n \geq n_0$
- For every $n \geq 2^{100}$ it holds $0 \leq g(n) \leq h(n)$ thus $0 \leq cg(n) \leq ch(n)$ for every $n \geq 2^{100}$
- $0 \leq cg(n) \leq ch(n) \leq f(n)$ for every $n \geq \max\{2^{100}, n_0\}$ $\Longrightarrow f(n) = \Omega(g(n))$

Question 3

Prove/Disprove:

Given that $f(n) = \log_a(n)$ for some parameter $a > 1$ and $g(n) = \lg(n)$ then $f(n) = \Theta(g(n))$

Solution (the claim is true):

- $f(n) = \log_a(n) = \dfrac{\lg(n)}{\lg(a)}$
- The parameter a is a constant
- For $c_1 = c_2 = \dfrac{1}{\lg(a)}$ and $n_0 = 1$ it holds that

$$0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \text{ for all } n \geq n_0 \Longrightarrow f(n) = \Theta(g(n))$$

Question 4

Let $p(n)$ be a degree d asymptotically positive polynomial i.e. $p(n) = \sum_{i=0}^{d} a_i n^i$, $a_d > 0$, $d \geq 0$. Prove $p(n) = \Theta(n^d)$.

Solution:

- According to question 1 it suffices to prove
  - $p(n) = O(n^d)$ and,
  - $p(n) = \Omega(n^d)$
- $p(n) = O(n^d)$:
- Denote by n_p the constant for which $p(n) > 0$ for all $n \geq n_p$
- Let $a_{max} = \max\{a_0, \ldots, a_d\}$. $a_d > 0$ thus $a_{max} > 0$
- $p(n) = \sum_{i=0}^{d} a_i n^i \leq \sum_{i=0}^{d} a_{max} n^i \leq a_{max} \sum_{i=0}^{d} n^i \leq a_{max}(d+1)n^d$
- For $c = a_{max}(d+1)$ and for all $n \geq 1$, $p(n) \leq cn^d$
- For all $n \geq n_0 = \max\{1, n_p\}$ it holds that $0 \leq p(n) \leq cn^d$ $\implies p(n) = O(n^d)$

Solution — cont.

- $p(n) = \Omega(n^d)$:
- If for all $0 \leq i \leq d$, $a_i \geq 0$ then, choose some c such that $0 < c \leq a_d$. For every $n \geq 1$ it holds $0 \leq cn^d \leq p(n) \implies p(n) = \Omega(n^d)$
- Otherwise, let $a_{min} = \min\{a_0, \ldots, a_d\}$, $a_{min} < 0$
- $p(n) = \sum_{i=0}^{d} a_i n^i \geq a_d n^d - \sum_{i=0}^{d-1} |a_{min}| n^i \geq a_d n^d - |a_{min}| d n^{d-1}$
- It is sufficient to find constants $c, n_0 > 0$ such that $0 \leq cn^d \leq a_d n^d - |a_{min}| d n^{d-1}$ for every $n \geq n_0$
- Clearly, $c < a_d$ is a necessary condition
- Set $c = \frac{a_d}{2}$ and we are left to set n_0
- After setting c we get: $0 \leq \frac{a_d}{2} n^d \leq a_d n^d - |a_{min}| d n^{d-1}$.

  For $n \geq 1$, $\frac{a_d n^d}{2} > 0$ thus we can divide by $\frac{a_d n^d}{2}$
- $0 \leq 1 \leq 2 - \frac{2|a_{min}|d}{a_d n}$. Set $n_0 = \left\lceil \frac{2|a_{min}|d}{a_d} \right\rceil$ to complete the proof

Solution — cont.

- Proof using limit
- $\lim_{n\to\infty} \frac{p(n)}{n^d} = \lim_{n\to\infty} \frac{\sum_{i=0}^{d} a_i n^i}{n^d} = \lim_{n\to\infty} \left(a_d + \sum_{i=0}^{d-1} \frac{a_i}{n^{d-i}} \right) = a_d$
- Since $a_d > 0$ is a constant $\Longrightarrow p(n) = \Theta(n^d)$

1. Asymptotic notation

2. Running time function

3. Insertion sort

Running time function

- Let I be an instance of size n ($|I| = n$), e.g. for sorting problem an instance is an array of size n with input in each cell
- Let $T_{\mathcal{A}}(I)$ be the running time of algorithm $\mathcal{A}$ on input instance I
- The running time function of algorithm $\mathcal{A}$ is

$$T_{\mathcal{A}}(n) = \max_{I:|I|=n} T_{\mathcal{A}}(I)$$

- Notice, the definition of $T_{\mathcal{A}}(n)$ corresponds to worst case analysis

Question 5

Prove/Disprove:

We are given an algorithm $\mathcal{A}$. If there exists an infinite series of instances $\{I_n\}$, $n \in \mathbb{Z}_{>0}$, $|I_n| = n$ such that the run time of algorithm $\mathcal{A}$ on every instance I of size n in $\{I_n\}$ is $T_{\mathcal{A}}(I) = 20n^2 - 5n + 3$, then $T_{\mathcal{A}}(n) = \Omega(n^2)$.

Solution (the claim is true):

- Recall $T_{\mathcal{A}}(n) = \max_{I:|I|=n} T_{\mathcal{A}}(I)$
- For every instance I, $|I| = n$ in $\{I_n\}$ it holds $20n^2 - 5n + 3 = T_{\mathcal{A}}(I) \leq T_{\mathcal{A}}(n)$
- Thus, for every $n \in \mathbb{Z}_{>0}$, $20n^2 - 5n + 3 \leq T_{\mathcal{A}}(n)$
- There exist constants $c, n_0 > 0$ such that $0 \leq cn^2 \leq 20n^2 - 5n + 3$ for every $n \geq n_0$ (proved in question 4)
- Thus $0 \leq cn^2 \leq T_{\mathcal{A}}(n)$ for every $n \geq n_0 \implies T_{\mathcal{A}}(n) = \Omega(n^2)$

Question 6

Prove/Disprove:

We are given an algorithm $\mathcal{A}$. If there exists a constant $n_0 > 0$ and an infinite series of instances $\{I_n\}_{n \geq n_0}$, $|I_n| = n$, such that the run time of algorithm $\mathcal{A}$ on every instance I of size n in $\{I_n\}$ satisfy $T_{\mathcal{A}}(I) \geq 20n^2 - 5n + 3$, then $T_{\mathcal{A}}(n) = \Omega(n^2)$.

Solution (the claim is true):

- For every instance I, $|I| = n \geq n_0$ in $\{I_n\}$ it holds $20n^2 - 5n + 3 \leq T_{\mathcal{A}}(I) \leq T_{\mathcal{A}}(n)$
- Thus, for every $n \geq n_0$, $20n^2 - 5n + 3 \leq T_{\mathcal{A}}(n)$
- There exists constants $c, n_1 > 0$ such that $0 \leq cn^2 \leq 20n^2 - 5n + 3$ for every $n \geq n_1$ (proved in question 4)
- Thus $0 \leq cn^2 \leq T_{\mathcal{A}}(n)$ for every $n \geq \max\{n_0, n_1\} \Longrightarrow T_{\mathcal{A}}(n) = \Omega(n^2)$

Question 7

Prove/Disprove:

We are given an algorithm $\mathcal{A}$. If there exist constants $c, n_0 > 0$ such that for every $n \geq n_0$ there exists an instance I, $|I| = n$, such that $T_{\mathcal{A}}(I) \leq cn^2$, then $T_{\mathcal{A}}(n) = O(n^2)$.

Solution (the claim is false)

- The run time $T_{\mathcal{A}}(n)$ of algorithm $\mathcal{A}$ is determined by the worst case instance of size n.
- The fact that on some instances the algorithm runs fast, does not rule out the existence of instances with greater run time

Question 8

Prove/Disprove:

We are given an algorithm $\mathcal{A}$. If $T_{\mathcal{A}}(n) = O(\lg(n))$, then there exist constants $c, n_0 > 0$ such that for every instance I, $|I| = n \geq n_0$ it holds that $T_{\mathcal{A}}(I) \leq c \lg(n)$.

Solution (the claim is true)

- $T_{\mathcal{A}}(n) = O(\lg(n)) \Longrightarrow$ there exist constants $c, n_0 > 0$ such that $0 \leq T_{\mathcal{A}}(n) \leq c \lg(n)$ for every $n \geq n_0$
- For every I, $|I| = n$ it holds $T_{\mathcal{A}}(I) \leq T_{\mathcal{A}}(n)$
- Thus $T_{\mathcal{A}}(I) \leq c \lg(n)$ for every instance I such that $|I| = n \geq n_0$

Question 9

Analyze the run time of the following algorithm:

```
Q9(n)
1: for i = 0 to n do
2:     print i^2
```

Solution:

- Each line by itself takes $\Theta(1)$ time
- Each iteration of the for loop requires $\Theta(1)$ time
- $\implies T_{Q9}(n) = (n+1)\Theta(1) = \Theta(n+1)\Theta(1) = \Theta(n)$

Question 10

Analyze the run time of the following algorithm:

```
Q10(n)
1: count = 0
2: for i = 0 to n do
3:     for j = 0 to n do
4:         print i + j
5:         count+ = 1
```

Solution:

- Line 1 requires $\Theta(1)$ time
- Each iteration of the for loop in lines 2-5 takes $\Theta(n)$ time
- $\implies T_{Q10}(n) = \Theta(1) + (n+1)\Theta(n) = \Theta(n^2)$

Question 11

Analyze the run time of the following algorithm:

```
Q11(n)
1: for i = 0 to n do
2:     for j = 0 to i do
3:         print i + j
```

Solution:

- Each iteration of the for loop in lines 1-3 takes at most $O(n)$ time
- $\implies T_{Q11}(n) \leq (n+1)O(n) = O(n^2)$
- For $\frac{n}{2} \leq i \leq n$ each iteration of the for loop in lines 1-3 takes at least $\Omega(\frac{n}{2}) = \Omega(n)$ time
- $\implies T_{Q11}(n) \geq \frac{n}{2}\Omega(n) = \Omega(n^2)$
- $\implies T_{Q11}(n) = \Theta(n^2)$

1 Asymptotic notation

2 Running time function

3 Insertion sort

Sorting Problem

The *sorting* ⟨מיון⟩ problem:

- Input: an array A of n numbers
- Output: reordering A so that $A[1] \leq A[2] \leq \cdots \leq A[n]$

`Insertion_Sort`(A)

1: **for** $j = 2$ to $A.length$ **do**
2: $\quad key = A[j]$
3: $\quad i = j - 1$ $\quad \triangleright$ insert key into the sorted sequence $A[1 \ldots j-1]$
4: $\quad$ **while** $i > 0$ and $A[i] > key$ **do**
5: $\quad\quad A[i+1] = A[i]$
6: $\quad\quad i = i - 1$
7: $\quad A[i+1] = key$

Question 12

Show $T_{\texttt{Insertion\_Sort}}(n) = \Omega(n^2)$.

Solution:

- Consider an array A of size n sorted in decreasing order
- In iteration j of the for loop the while loop makes at least $(j-1)$ iterations
- Iteration j of the for loop takes $\geq \Omega(1) + (j-1)\Omega(1)$ time
- $T_{\texttt{Inseration\_Sort}}(n) \geq T_{\texttt{Inseration\_Sort}}(A) \geq \sum_{j=2}^{n} \Omega(1) + (j-1)\Omega(1) \geq \Omega(1) \sum_{j=2}^{n} 1 + (j-1) = \Omega(1) \sum_{j=2}^{n} j = \Omega(1) \frac{(n-1)(2+n)}{2} = \Omega(1) \cdot \Omega(n^2) = \Omega(n^2)$
- From lecture: $T_{\texttt{Inseration\_Sort}}(n) = O(n^2)$ thus $T_{\texttt{Inseration\_Sort}}(n) = \Theta(n^2)$
- What is the running time of a sorted array A of size n?
  - $T_{\texttt{Inseration\_Sort}}(A) = \Theta(n)$ since each iteration j of the for loop takes only $\Theta(1)$ time