Ben-Gurion University of the Negev

Data Structures

202.1.1031

Assignment No. 1

Responsible staff members:

Michal Shemesh(shemeshm@post.bgu.ac.il) Yotam Ashkenazi(yotamash@post.bgu.ac.il)

Authors:

John Doe (123456789) Jane Doe (987654321)

Publication date: March 23rd, 2023

Submission date: April 4th, 2023

אוניברסיטת בן-גוריון בנגב

Ben-Gurion University of the Negev

1 Complexity Hierarchy

For each of the functions f_i listed below find $g(n)$ s.t. $f_i(n) \in \theta(g(n))$ and place f_i in the table s.t.:

- Functions f_i and f_j will be located on the same line iff $f_i(n) = \theta(f_j(n))$
- For each function f_i in line k and f_j in line $k+1$, $f_i(n) = O(f_j(n))$

Prove your answer:

- Find the asymptotic order $g(n)$ and prove (using the definition) that $f_i(n) = \theta(g(n))$ for each i.
- For each two functions f_i, f_j placed in the same line prove that $f_i(n) = \theta(f_j(n))$.
- For each two functions f_i in line k and f_j in line $k+1$ prove that $f_i(n) = O(f_j(n))$.
  - $f_1(n) = 2023$
  - $f_2(n) = 2^{\sqrt{n}}$
  - $f_3(n) = 4^{(2^n)}$
  - $f_4(n) = 2^{64}$
  - $f_5(n) = n^n$
  - $f_6(n) = \log(2^n * n^2)$
  - $f_7(n) = \log(n^{\frac{1}{2}})$
  - $f_8(n) = \frac{2n}{7}$

f_i	Order - $\theta(g(n))$
f_1	$\theta(1)$
f_2	$\theta(2^{\sqrt{n}})$
f_3	$\theta(2^{(2^{n+1})})$
f_4	$\theta(1)$
f_5	$\theta(n^n)$
f_6	$\theta(n)$
f_7	$\theta(\log n)$
f_8	$\theta(n)$

Answer to Question 1:
First section:
in every question we must find C and n_0)

1: $f_1 = 2023 \in \theta(1)$
$c_1 = 2024$ for every $n > 0$ 2023 <= 2024 so $2023 = o(1)$
$c_2 = 1$ for every $n > 0$
$1 * 1 <= 2023$
therefor $2023 \in \Omega(1)$
so $2023 \in \theta(1)$

2:
$f_2 = \theta(2^{\sqrt{n}})$
there is no well known asymptotic behavior for this function so $f_2 \in \theta(f_2)$ (if we chose $n_0 = 1, c_1 = 1, c_2 = 1$)

3:
$f_3(n) = 4^{(2^n)} \in \theta(2^{(2^{n+1})}$
in this function also there is no other asymptotic behavior as this one consequently $4^{2n} = 2^{2n+1}$ so we can pick $c_1 = 0.5, n_0 = 1, c_2 = 2$
and the equation $c_1 * 2^{2n+1} <= 4^{2n} <= c2 * 2^{2n+1}$ foe every $n > n_0$

4: $f_4(n) = 2^{64} \in \theta(1)$
from the definition of $\theta(1)$ we can assume that in this case $\theta(1)$ is a rational number d

hence we can pick $c1 = \frac{1}{d}$
$n_0 = 1$
$c_2 = 2^{64} * \frac{1}{d}$
and the equation: $c_1 * d <= 2^{64} <= c_2 * d$
will be true for every $n > 0 (n_0 = 0)$

5:
$f_5(n) = n^n \in \theta n^n$
the function n^n is well defined in the asymptotic behavior
therefor if we chose $c_1 = 1, n_0 = 1, c_2 = 1$
the equation: $c1 * n^n <= n^n <= c2 * n^n$
for every $n > 0$
6:
$f_6(n) = \log(2^n * n^2) \in \theta(n)$
in this question we will use log identities (no need to write them down..)
$\log(2^n * n^2) = \log(2^n) + \log(n^2)$
$= n \log + 2\log(n) = n + 2log(n)$
therefor we must find c_1, c_2, d to solve the equation: $c_1 * (n) <= n + 2log(n) <= c_2 * (n)$
so we can pick $c_1 = 1$
the solve the right side of the equation we know for all $n > 1$ $\log(n) < n$
therefor we can pick $c_1 = 3$, $n_0 = 1$
and the equation will be true
therefor
$\log(2^n * n^2) \in \theta(n)$

7:
$f_7(n) = \log(n^{\frac{1}{2}}) \in \theta \log(n)$
also here we gonna use log identities
$\log(n^{\frac{1}{2}}) = \frac{1}{2} * \log(n)$
there for if we chose $c_1 = \frac{1}{2}, n_0 = 1, c_2 = 2$
the equation: $c_1 * log(n) \leq \log(n^{\frac{1}{2}}) \leq c_2 * log(n)$
for all $n \geq n_0$.

8: $f_8(n) = \frac{2n}{7} \in \theta(n)$
if we chose $c_1 = \frac{1}{10}, c_2 = 1, n_0 = 1$
the equation: $c_1 n \leq \frac{2n}{7} \leq c_2 * n$
is true (for all $n \geq n_0$)

second section:
1:
we must prove that $f(1) \in \theta f(4)$
$f(1) = 2023,$
$f(2) = (2^{64})$

if we chose $c_1 = \frac{1}{2^{64}}, n_0 = 1, c_2 = 1$
the equation:
$c_1 * 2^{64} \leq 2023 \leq c_2 * 2^{64}$
(for all $n_0 \geq n$)
there for : $f(1) \in \theta f(4)$

2:
$f(8) \in \theta(f(6))$
in other words: $\frac{2n}{7} \in \theta(\log(2^n * n^2))$
in the last section we proved that $\log(2^n * n^2)$
$= n + 2log(n)$

also as we know for all $n \geq 1$
$n \geq \log(n)$

therefor if we chose: $c_1 = 1$,
$n_0 = 1$,
$c_2 = 14$
the equation will be true :
$c_1 * \frac{2n}{7} \leq \log(2^n * n^2) \leq c_2 * \frac{2n}{7}$
(for all $n > n_0$)...

third section:

1: we will prove that $f(1) \in O(f(7))$ and result from transpose symmetry $\theta(1) \in O(\log(n))$
if we chose $c_1 = 2023$,
and $n_0 = 16$
the equation : $2023 \leq c_1 * \log(n^{\frac{1}{2}})$
is true for all $n \geq n_0$ (what we wanted to prove)

2:
$\log(n^{\frac{1}{2}}) \in O(\log 2^n * n^2)$
we have proved in the sections above that $O(log2^n * n^2) = n + 2\log(n)$
and also we have proved that for all $n > 0$, $n > \log(\frac{n}{2})$
therefor if we chose $c_1 = 1, n_0 = 0$
the equation: $\log(n^{\frac{1}{2}}) \leq 1 * (\log 2^n * n^2)$
will be true (for all $n \geq n_0$)

3:

$\frac{2n}{7} \in O(2^{(\sqrt{n})})$
if we chose:
$c_1 = 1, n_0 = 4$
the equation :
$\frac{2n}{7} \leq c1 * 2^{(\sqrt{n})}$

for all $(n > n_0)$

4:
$2^{(\sqrt{n})} \in O(n^n)$
if we chose $n_0 = 2, c_1 = 1$
the equation : $2^{(\sqrt{n})} \leq c_1 * n^n$
is true for all $(n > n_0)$

5:

$n^n \in O(4^{2n})$
we will prove in this clause that for all $n^n \leq (4^{2n})$
so we add log for the two sides and we get $\log(n^n) \leq \log 4^{(2^n)}$
$\log(n^n) = n\log(n)$(log identity)
$\leq 2^n \leq 2^n * \log(n) = log4^{(2^n)} \leq c1 * \log 4^{(2^n)}$

$c_1 = 1$,$n_0 = 1$
therefor:
the equation is true :
$n^n \in log4^{(2^n)}$

2 Properties of asymptotic bounds

Let $f(n)$ and $g(n)$ be asymptotically positive functions. Prove or disprove each of the following conjectures.

1. Transitivity: If $f(n) = \theta(g(n))$ and $g(n) = \theta(h(n))$ then $f(n) = \theta(h(n))$

2. Reflexivity: $f(n) = \theta(f(n))$

3. Transpose Symmetry: $f(n) = O(g(n)) \iff g(n) = \Omega(f(n))$

4. $f(n) = \theta(f(\frac{n}{2}))$

Answer to Question 2: (Write your answers here:)

1. we assume that $f(n) = \theta(g(n))$, $g(n) = \theta(h(n))$
   to prove that $f(n) = \theta(h(n))$
   we conclude from the hypothesis that exists c_1, c_2, c_3, c_4 s.t:
   $c_1 * g(n) <= f(n) <= c_2 * g(n)$, for every $n > n_0$
   $c_3 * h(n) <= g(n) <= c_4 * h(n)$, for every $n > n_1$
   $c_3 * h(n) <= g(n) \longrightarrow c_1 * c_3 * h(n) <= c_1 * g(n) <= f(n)$
   $h(n) = O(f(n))$ for every $n > n_2, n_2 = max(n_0, n_1)$
   $g(n) <= c_4 * h(n) \longrightarrow f(n) <= c_2 * g(n) <= c_2 * c_4 * h(n)$
   $h(n) = \Omega(f(n))$ for every $n > n_2, n_2 = max(n_0, n_1)$
   $c_1 * c_3 * h(n) <= f(n) <= c_2 * c_4 * h(n) \longrightarrow f(n) = \theta(h(n))$, for every $n > n_2, n_2 = max(n_0, n_1)$

2. to prove Reflexivity we should find c_1, c_2, n_0, s.t:
   $c_1 * f(n) <= f(n) <= c_2 * f(n)$
   because of $f(n) = f(n)$, for every $n > 0$, so we can choose $c_1 = \frac{1}{2}, c_2 = 2$
   we can see that: $\frac{1}{2} * f(n) <= f(n) <= 2 * f(n)$, for every $n > 0$
   so $f(n) = \theta(f(n))$

3. to prove Transpose Symmetry we should divide the proof for two sections:

   first section:
   we assume that $f(n) = O(g(n))$, we should prove that $g(n) = \Omega(f(n))$.
   $f(n) = O(g(n))$, so exists c_1, n_1 s.t $f(n) <= c_1 * g(n)$ (for every n $> n_1$)
   $\frac{1}{c_1} f(n) <= g(n)$, $c_2 = \frac{1}{c_1}$, $n_2 = n_1$
   therefor $g(n) = \Omega(f(n))$

   second section:
   we assume that $g(n) = \Omega(f(n))$, we should prove that $f(n) = O(g(n))$.
   $g(n) = \Omega(f(n))$, so exists c_1, n_1 s.t $g(n) >= c_1 * f(n)$ (for every n $> n_1$).
   $\frac{1}{c_1} * g(n) >= f(n)$, so exists $c_2 = \frac{1}{c_1}, n_1 = n_0$ s.t $c_2 * g(n) >= f(n)$
   therefor $f(n) = O(g(n))$

   we proved the two sections so we proved that $f(n) = O(g(n)) \iff g(n) = \Omega(f(n))$.

4. we will break through a counter example:
   we need to show that it does not exist c_1, c_2 s.t:
   $c_1 * f(\frac{n}{2}) <= f(n) <= c_2 * f(\frac{n}{2})$.

   $f(n) = n^n$, because of c_2 is a constant there is no c_2 maintain that $f(n) <= c_2 * f(\frac{n}{2})$.

3 Recurrence Examples

Find and prove an asymptotic tight bound Θ for $T(n)$ in each of the following recurrences. Assume that $T(c) = C$ (you can choose any 2 numbers for c and $C \in \mathbb{N}$).

1. $T(n) = T(\sqrt{n}) + 1$

2. $T(n) = 4T(\frac{n}{2}) + n^3 \log n$

3. $T(n) = T(\frac{2}{5}n) + 1$

4. $T(n) = 6T(\frac{n}{3}) + n$

Answer to Question 3:

1. Guess: $T(n) = \log(\log(n))$
   Base: for $n = 2$, $\log(\log(2)) = 0$
   Induction hypothesis: we assume that $T(m) = \log(\log(m))$, for all $m < n$.
   Induction step:
   $T(n) = T(\sqrt{n}) + 1 = \log(\log(\sqrt{n})) + 1 = \log(\log(\sqrt{n})) + \log(\log(4)) = \log(\log(\sqrt{n}) * \log(4))$
   $= \log(\log(\sqrt{n}) * 2) = \log(\log(\sqrt{n})^2) = \log(\log(n))$
   there for $T(n) = \Theta(\log(\log(n))$

2. we will use the Master method: (third case)
   $a = 4, b = 2, f(n) = n^3 * \log(n)$

   condition number 1: $(f(n) = O(n^{\log_b a+\epsilon}))$
   $\epsilon = \frac{1}{2}$ —–> $n^3 * \log(n) >= n^{2+\frac{1}{2}}$
   therefor $f(n) = \Omega(n^{2+\frac{1}{2}})$

   condition number 2: $a * f(\frac{n}{b}) <= c_1 * f(n)$
   $4 * (\frac{n^3}{8} * \log(\frac{n}{2})) <= c_1 * n^3 * \log(n)$, $c_1 = \frac{1}{2}$
   $\frac{n^3}{2} * (\log(n) - 1) <= \frac{n^3 * \log(n)}{2}$
   therefor $T(n) = \theta(n^3 * \log(n))$

3. we will solve this Withdrawal formula by using the master theorem:
   $a = 1, b = \frac{5}{2}$
   $f(n) = 1$
   therefore: $n^{log_{\frac{5}{2}} 1} = n^0 = 1$
   therefor $f(n) \in \theta(1)$
   (that means the second case)
   consequently the answer is : $T(n) \in \theta(\log n)$

4. we will solve this Withdrawal formula by using the master theorem:
   $a = 6$, $b = 3$, $f(n) = n$,

   $n^{\log_3 6}$
   $\log_3 6 = 1.63$ we chose $\epsilon = 0.1$
   that mean that we are in the first case..
   which means that:
   $T(n) = \theta(n^{\log_3 6})$

4 Time Complexity

Find the time complexity of the following algorithms in terms of Θ. You are expected to analyze the asymptotic tight bound of each line in the algorithms, as presented in class for the analysis of insertion sort.

Algorithm 1 Selection Sort (array[int])

1: **for** $i = 0;\ i < array.length - 1;\ i{+}{+}$ **do**
2: $\quad minInd \leftarrow MinIndex(array, i)$
3: $\quad Swap(array, i, minInd)$
4: **end for**

Algorithm 2 MinIndex (array,from)

1: $minIndex \leftarrow from$
2: **for** $i = from + 1;\ i < array.length;\ i{+}{+}$ **do**
3: $\quad$ **if** $arr[i] < arr[minIndex]$ **then**
4: $\quad\quad minIndex \leftarrow i$
5: $\quad$ **end if**
6: **end for**
7: **return** $minIndex$

Algorithm 3 Swap (array,i, j)

1: $tmp \leftarrow array[i]$
2: $array[i] \leftarrow array[j]$
3: $array[j] \leftarrow tmp$

Answer to Question 4: (Write your answer in the following tables)

Algorithm 1 line number	Times	Cost
1	n-1	$\theta(n)$
2	n	$\theta(n^2)$
3	n	$\theta(n)$
Total	—	$\theta(n^2)$

Algorithm 2 line number	Times	Cost
1	1	$\theta(1)$
2	n-1	$\theta(n)$
3	n-1	$\theta(n)$
4	n-1	$\theta(n)$
5	n-1	$\theta(0)$
6	1	$\theta(0)$
7	1	$\theta(1)$
Total	—	$\theta(n)$

Algorithm 3 line number	Times	Cost
1	1	$\theta(1)$
2	1	$\theta(1)$
3	1	$\theta(1)$
Total	—	$\theta(1)$

5 Algorithm Development

Describe in pseudo-code an efficient algorithm for solving the following problem, and analyze its running time and memory usage. The running time should be asymptotically low as possible.

Input: 2 unsorted arrays A and B of size $N \in \mathbb{N}$. The value of each element in the arrays is a natural number in the range $[0, 2N]$.
Output: True iff all the numbers in A are different from all the numbers in B (that is, iff no value exists in both arrays).

Examples:

Input:

A	5	2	14	2	2	7	1
B	6	9	9	10	13	0	8

Output: True

Input:

A	5	2	14	2	2	7	1
B	6	9	9	10	7	13	8

Output: False

Answer to Question 5: (Write your answer here)

Algorithm description, run-time and memory usage:

Algorithm description: let c to be a new array of size 2N

```
for (i=0 to n-1) do
  c[A[i]] = 1

for (i=0 to n-1)
  if (c[B[i]] =1 ) do
    return false

return true
```

run time : n
which is asymptotic is: $\theta(n)$

memory usage : 2N (the length of the array...)
which is asymptotic is : $\theta(n)$