Solutions for Exercise Set #2

CMPUT 204, Fall 2019
Department of Computing Science
University of Alberta

Problem 1. Exercise 2.3-6 from the book (page 39).

Problem 2. True or False? Justify your answer briefly. Some of the questions are alerady answered in Exercises Set 1 or the course notes, but here a justification is required.

True **a.** $n \log^5 n \in O(n^{\frac{5}{4}})$

False **b.** $2^{n \log n} \in \Theta(6^n)$

True **c.** For constant $a > 1$: $a^n \in o(n!)$.

True **d.** $(\sqrt{2})^{\log n} \in o(2^{\log n})$ 1.414

True **e.** $2^{\sqrt{\log n}} \in o((\sqrt{2})^{\log n})$ $2^{(\log n)^n}$

true **f.** $n! \in O(2^{n \log n})$ $\leq O(n^n) \leq$

true **g.** $\log(n!) \in \Theta(n \log n)$

g. $\log(n!) = \sum_{i=1}^{n} \log(i) \leq \sum_{i=1}^{n} \log(n) = n \log n$

$n \log n = O(\log(n!))$

$\log(n!) = \log n + \log(n-1) + \cdots + \log(1) \geq \log(n) + \log(n-1) + \cdots + \log(\frac{n}{2})$

$\geq \frac{n}{2} \log(\frac{n}{2}) = \frac{n}{2}(\log n - \log 2) = \frac{n}{2}(\log n - 1) \geq \frac{n}{4} \log n$

Problem 3. The sequence of Fibonacci numbers: $F(0) = F(1) = 1$ and $F(i) = F(i-1) + F(i-2)$ for $i \geq 2$. Prove that $F_n \in \Omega((3/2)^n)$.

$O(n^n) =$

Problem 4. Prove or disprove each of the following statements:

a. For any functions $f, g : \mathbb{N} \longrightarrow R^+$, either $f \in O(g)$ or $g \in O(f)$.

b. For any functions $f, g : \mathbb{N} \longrightarrow R^+$, if $f(n) > g(n)$ for all $n > 0$ then $f(n) + g(n) \in \Theta(f(n))$.

c. For any functions $f, g : \mathbb{N} \longrightarrow R^+$, if $f(n) \in \Theta(n)$ and $g(n) \in \Theta(n)$ then $2^{f(n)} \in \Theta(2^{g(n)})$.

d. For any functions $f, f', g, g' : \mathbb{N} \longrightarrow R^+$, if $f \in O(f')$ and $g \in O(g')$ then $(f + g) \in O(\max(f', g'))$, where $(f + g)(n)$ is defined to be $f(n) + g(n)$ for any value of $n \in \mathbb{N}$.

e. For any functions $f, g, h : \mathbb{N} \longrightarrow R^+$, if $(f \cdot g) \in O(h)$ (where $(f \cdot g)(n) = f(n) \cdot g(n)$) then $f \in O(h)$ and $g \in O(h)$.

f. There is a fixed constant $\sigma > 1$ such that $n^\sigma \in \Theta(n \log n)$.

Problem 5. Exercise 4.3-1, 4.3-6, 4.4-2, 4.4-4, 4.5-1, 4.5-4 from the book.

Problem 6. Find asymptotic upper/lower bounds for $T(n)$. Assume that $T(n)$ is constant for small n.

- $T(n) = 3T(n/2) + \sqrt{n}$
- $T(n) = 5T(n/5) + n \log n$
- $T(n) = \sqrt{n}T(\sqrt{n}) + n$.
- $T(n) = T(n-2) + 2 \log n$.

Problem 7. Suppose that we are given an array A (not necessarily sorted) with n integers as well as another integer S. Give an algorithm with running time $O(n \log n)$ which determines if there are two elements of A whose sum is exactly S and if so, finds them.

Problem 8. An array A[1...n] is unimodal if it consists of an increasing sequence followed by a decreasing sequence, or more precisely, if there is an index $m \in 1, 2, ..., n$ such that

- $A[i] < A[i + 1]$ for all $1 \leq i < m$, and
- $A[i] > A[i + 1]$ for all $m \leq i < n$.

Give an algorithm to compute the maximum element of a unimodal input array A[1...n] in $O(\log n)$ time. Argue for the correctness of your algorithm by presenting the loop invariant(s) that your algorithm maintains and show why they lead to the correctness of your algorithm. Finally, prove the bound on its running time.

1. while $i > 0$ and $A[i] >$ key do

$A[i+1] = A[i]$

$i = i - 1$

Although we can reduce the number of comparisons by binary search, we still need to shift all elements greater than key towards the end to insert key in the proper position. and this shifting takes $\Theta(n)$. so the overall-worst case is still $\Theta(n^2)$

2 e. $2^{\sqrt{\log n}} \in O(\sqrt{2}^{\log n})$

$= \log(2^{\sqrt{\log n}})$ $\quad \log(\sqrt{2}^{\log n})$

$= \sqrt{\log n} \cdot \log 2 < \log n \log \sqrt{2}$

f. $n! \in O(2^{n \log n})$

$n! < n^n = (2^{\log n})^n < 2^{n \log n}$

Problem 3. The sequence of Fibonacci numbers: $F(0) = F(1) = 1$ and $F(i) = F(i-1) + F(i-2)$ for $i \geq 2$. Prove that $F_n \in \Omega((3/2)^n)$.

base case: when $n=5$, $F(n) = 8$, $(\frac{3}{2})^5 = 7.59$

$F(5) \in \Omega(\frac{3}{2})^5$

inductive step: assume $F(k) > c(\frac{3}{2})^n$ for $k \geq n_0$

show $F(k+1) > c(\frac{3}{2})^{k+1}$

$F(k+1) = F(k) + F(k-1)$

$= c(\frac{3}{2})^k + c(\frac{3}{2})^{k-1}$

$= c(\frac{3}{2}+1)(\frac{3}{2})^{k-1}$

$> c(\frac{3}{2})^2(\frac{3}{2})^{k-1}$

$= c(\frac{3}{2})^{k+1}$

Problem 4. Prove or disprove each of the following statements:

a. For any functions $f, g : \mathbb{N} \longrightarrow R^+$, either $f \in O(g)$ or $g \in O(f)$.

b. For any functions $f, g : \mathbb{N} \longrightarrow R^+$, if $f(n) > g(n)$ for all $n > 0$ then $f(n) + g(n) \in \Theta(f(n))$.

c. For any functions $f, g : \mathbb{N} \longrightarrow R^+$, if $f(n) \in \Theta(n)$ and $g(n) \in \Theta(n)$ then $2^{f(n)} \in \Theta(2^{g(n)})$.

d. For any functions $f, f', g, g' : \mathbb{N} \longrightarrow R^+$, if $f \in O(f')$ and $g \in O(g')$ then $(f + g) \in O(\max(f', g'))$, where $(f + g)(n)$ is defined to be $f(n) + g(n)$ for any value of $n \in \mathbb{N}$.

e. For any functions $f, g, h : \mathbb{N} \longrightarrow R^+$, if $(f \cdot g) \in O(h)$ (where $(f \cdot g)(n) = f(n) \cdot g(n)$) then $f \in O(h)$ and $g \in O(h)$.

f. There is a fixed constant $\sigma > 1$ such that $n^\sigma \in \Theta(n \log n)$.

a: suppose $f(n) = \begin{cases} 0 & \text{if } n \text{ is odd} \\ n & \text{if } n \text{ is even} \end{cases}$ $g(n) = \begin{cases} n & \text{if } n \text{ is odd} \\ 0 & \text{if } n \text{ is even} \end{cases}$

for any constant c, d such that $c, d > 0$, even if $n > d$, always

$f(n) > c(g(n))$ if n is even and $g(n) > c \cdot f(n)$ in n is odd

b. since $f(n) > (gn)$, $g(n) \in O(fn)$ and $fn \in \Omega(g(n))$

then $fn + gn = \Omega(gn) + O(fn) = O(fn)$

$= \Omega(fn)$

c. false. if $fn = (2n)$ and $g(n) = n$

$2^{fn} = 2^{2n} = 4^n$ $2^{g(n)} = 2^n$

$4^n \notin \Theta(2^n)$

d. there are constant $c_1, c_2 \in R^+$ and $n_1, n_2 \in N$ such that:

$f(n) \leq c_1 \cdot f'(n)$ $g(n) \leq c_2 \cdot g'(n)$

$(f+g)(n) = f(n) + g(n) \le c_1 \cdot f'(n) + c_2 \cdot g'(n)$

$\le \max(c_1, c_2) \cdot \max(f'(n), g'(n)) + \max(c_1, c_2) \cdot \max(f'(n), g'(n))$

$= 2\max(c_1, c_2) \max(f'(n), g'(n)) \in O(\max(f'(n), g(n)))$

e. False. if $f(n) = n^2$, $g(n) = \frac{1}{n}$, $h(n) = n$

but $n^2 \notin O(n)$

f. false. for any $\sigma > 1$, n^{σ} is a polynomial and this is always asymptotically greater than $\log n$.

5. assume $T(n) \le cn^2$, where $c \in \mathbb{R}^+$, then $T(n) \le c(n-1)^2 + n$

$= c(n^2 - 2n + 1) + n$

$= cn^2 - 2nc + c + n$

$= cn^2 - n(2c - 1) + c$

$\le cn^2 - 2c + c \le cn^2$

since $n \ge 1$

4-3-6: assume $T(n) \le cn\log n$

$T(n) = 2T(\lfloor \frac{n}{2} \rfloor + 17) + n$

$= 2c(\frac{n}{2} + 17)\log(\frac{n}{2} + 17) + n$

since $2(\frac{n}{2}) < n$

$T(n) \le 2c(\frac{n}{2} + 17)\log(\frac{n}{2} + \frac{n}{4}) + n \qquad \forall \frac{n}{4} > 17$

$= (cn + 34)\log(\frac{3n}{4}) + n$

$= cn\log(n) - cn\log(\frac{4}{3}) + 34\log n - 34\log(\frac{4}{3}) + n$

$\leq cn\log(n) - cn\log\frac{4}{3} + 34\lg n + n$ since $\log n \leq O(n)$ $34\log n = kn$

$T(n) \leq cn\log n + n(1 - c\log\frac{4}{3}) + kn$

$= cn\log n + (k+1 - c\log\frac{4}{3})n$

show $k+1 - c\log\frac{4}{3} < 0$

$k+1 < c\log\frac{4}{3}$

$c \geq \frac{k+1}{\log\frac{4}{3}}$

$\therefore T(n) \leq cn\log n$

4.4-2

n^2 $(\frac{n}{1})^2$

$\frac{n^2}{4}$ $(\frac{n}{2})^2$

$\frac{n^2}{16}$ $(\frac{n}{4})^2$

$\frac{n^2}{256}$ $(\frac{n}{16})^2$

$\vdots$

$\frac{n^2}{2^i}$

$T(n) = \sum_{i=0}^{\lg n} c\left(\frac{n}{2^i}\right)^2$

$= cn^2 \sum_{i=0}^{\lg n} \left(\frac{1}{4}\right)^i$

$= cn^2 \sum_{i=0}^{\infty} \left(\frac{1}{4}\right)^i$

$= O(n^2)$

assume $T(n) = O(n^2)$ holds,

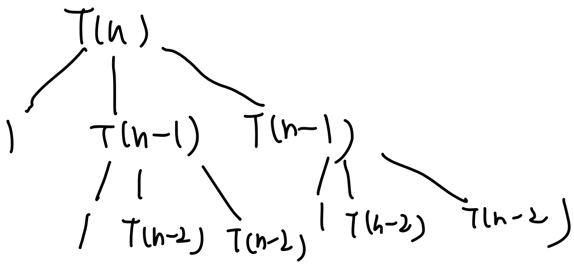
$T(n) = T(\frac{n}{2}) + n^2$

$\leq c(\frac{n}{2})^2 + n^2$

$= c(\frac{n^2}{4}) + n^2$

$= (\frac{c}{4} + 1)n^2 \leq cn^2$

4.4-4



1	1
2	2^1
4	2^2
8	2^3
$\vdots$	

$$T(n) = 1+2^1+2^2+2^3+\cdots 2^n = \sum_{i=0}^{n-1} 2^i$$
$$= \frac{2^{n-1}-1}{2-1}$$
$$= 2^{n-1}-1 \leq 2^n-1$$

$$T(n) = 2T(n-1)+1$$
$$= 2(2^{n-1}-1)+1$$
$$= 2^n-2+1$$
$$= 2^n-1$$

4.5-1: 1. $a=2$ $b=4$ $d=0$

since $b^d = 4^0 = 1 < a$

$T(n) = \Theta(n^{\log_b a}) = \Theta(n^{\log_4 2}) = \Theta(n^{\frac{1}{2}})$

2. $a=2$ $b=4$ $d=\frac{1}{2}$

$b^d = 4^{\frac{1}{2}} = 2 = a$

$T(n) = \Theta(n^d \log n) = \Theta(n^{\frac{1}{2}} \log n)$

3. $a=2$ $b=4$ $d=1$

$b^d = 4^1 = 4 > a$

$T(n) = \Theta(n^d) = \Theta(n)$

4.5-4: $a=4$ $b=2$ $f(n)=n^2\log n$

$f(n) = n^2 \log n \neq O(n^{2-\epsilon}) \neq \Omega(n^{2-\epsilon})$

4. $a=2$ $b=4$ $d=2$

$b^d = 4^2 = 16 > a$

$T(n) \in \Theta(n^d) = \Theta(n^2)$

Problem 6. Find asymptotic upper/lower bounds for $T(n)$. Assume that $T(n)$ is constant for small n.

- $T(n) = 3T(n/2) + \sqrt{n}$
- $T(n) = 5T(n/5) + n \log n$
- $T(n) = \sqrt{n}T(\sqrt{n}) + n$.
- $T(n) = T(n-2) + 2\log n$.

1. $a=3$ $b=2$ $d=\frac{1}{2}$

$b^d = 2^{\frac{1}{2}} = \sqrt{2} < a$

$T(n) = \Theta(n^{\log_b a})$

$= \Theta(n^{\log_2 3})$

2. $a=5$ $b=5$ $d=\Theta(n\log n)$

$\Theta = (n \log^2 n)$

3.

4. $T(n-2) + 2\log n$ assume $T(n) \leq cn\log n$ for $c \geq 1$

$= c(n-2)\log(n-2) + 2\log n$

$\leq (cn - 2c)\log(n) + 2\log n$

$\leq cn\log n - 2c\log n + 2\log n$

$= cn\log n - 2\log n(c-1)$ since $c \geq 1$

$\leq cn\log n$

$\therefore T(n) = O(n\log n)$

Problem 7. Suppose that we are given an array A (not necessarily sorted) with n integers as well as another integer S. Give an algorithm with running time $O(n\log n)$ which determines if there are two elements of A whose sum is exactly S and if so, finds them.

```
mergeSort(S, 1, n)
for i=1 to n
    if BinarySearch(S, x - S[i]) != NIL
        return true
return false
```

Problem 8. An array A[1...n] is unimodal if it consists of an increasing sequence followed by a decreasing sequence, or more precisely, if there is an index $m \in 1, 2, ..., n$ such that

- $A[i] < A[i + 1]$ for all $1 \leq i < m$, and
- $A[i] > A[i + 1]$ for all $m \leq i < n$.

Give an algorithm to compute the maximum element of a unimodal input array A[1...n] in $O(\log n)$ time. Argue for the correctness of your algorithm by presenting the loop invariant(s) that your algorithm maintains and show why they lead to the correctness of your algorithm. Finally, prove the bound on its running time.

a, b ← 1, n
while a < b
    mid = (a+b)/2
    if A[mid] < A[mid+1]   then a ← mid+1
    if A[mid] > A[mid+1]   then b ← mid
end while;
return A[a].