CS 460, HOMEWORK 1
(DUE: SEP 20, 11:59 PM)

INSTRUCTOR: HOA VU

- Each question is worth 20 points.
- Submission should be in PDF. There are some scanner apps that improves the readability if you handwrite the solutions.
- You can work in groups of 2. Make sure to list the names of people in your group. Each student must still submit a copy on Canvas.
- When you are asked to design an algorithm, do the following: a) describe the algorithm, b) prove/argue why it is correct, and c) provide the running time.
- [Erickson] denotes the book by Jeff Erickson (available for free online at `http://jeffe.cs.illinois.edu/teaching/algorithms/`).
- [DPV] denotes the book by Dasgupta, Papadimitriou, and Vazirani (the required textbook).

(1) **Question 1:**
   - Show that $1+1/2+1/3+\ldots+1/n = O(\ln n)$ using approximation by integrals. For a list of common integrals, see `https://en.wikipedia.org/wiki/Lists_of_integrals`
   - Problem 0.1 parts e, f, k, n, and o of [DPV]

(2) **Question 2:** Solve for big-O of the following recurrences. The answer should be as tight as possible.
   - $T(n) = T(2n/3) + O(1)$.
   - $T(n) = 7T(n/7) + O(n)$.
   - $T(n) = 7T(2n/3) + O(n^2)$.
   - $T(n) = T(n-2) + O(n)$.

(3) **Question 3:** Problem 37 on page 64 of [Erickson]. You can assume the tree is represented by a linked data structure. Each node has a pointer to its left child, right child, and its parent. For example, if v is a node, then $v.left$, $v.right$, $v.parent$ point to its left child, right child, and parent respectively. If $v.left = NULL$, that means v does not have a left child (and similarly for $v.right$ and $v.parent$).

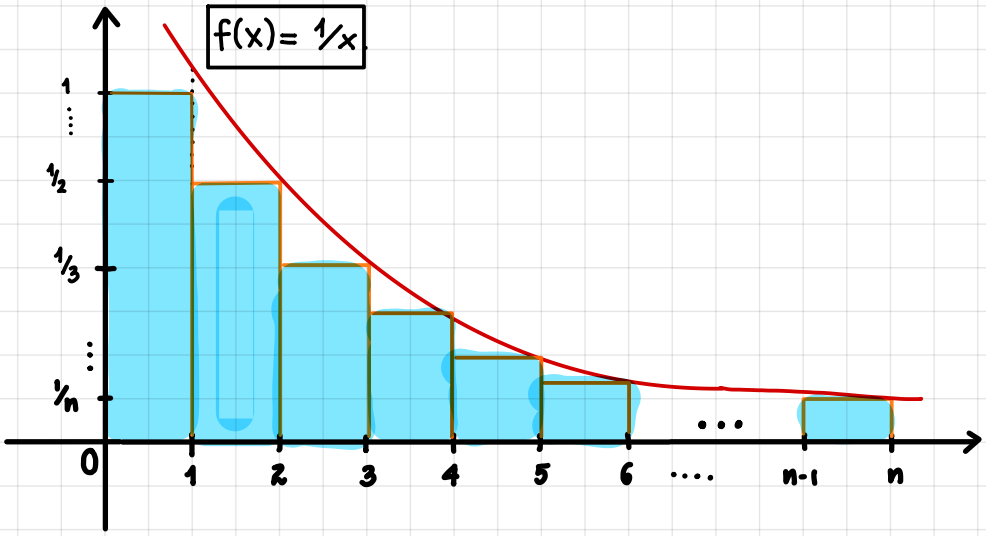
(4) **Question 4:** Problem 2.17 of [DPV].

(5) **Question 5:**
   - Use induction to prove that $\sum_{i=0}^{n} i2^i = 2 + (n-1)2^{n+1}$.
   - Use induction to prove that for any natural number n, $5^n - 1$ is divisible by 4.

(1) **Question 1:**

- Show that $1+1/2+1/3+\ldots+1/n = O(\ln n)$ using approximation by integrals. For a list of common integrals, see `https://en.wikipedia.org/wiki/Lists_of_integrals`.
- Problem 0.1 parts e, f, k, n, and o of [DPV]

(a) By using approximation by integrals, we have:



$$1+\frac{1}{2}+\frac{1}{3}+\ldots+\frac{1}{n} \le 1+\int_1^n \frac{1}{x}\,dx$$

$$1+\frac{1}{2}+\frac{1}{3}+\ldots+\frac{1}{n} \le 1+\ln x\Big|_1^n$$

$$1+\frac{1}{2}+\frac{1}{3}+\ldots+\frac{1}{n} \le 1+\ln n-\ln 1$$

$$1+\frac{1}{2}+\frac{1}{3}+\ldots+\frac{1}{n} \le 1+\ln n$$

$$\boxed{1+\frac{1}{2}+\frac{1}{3}+\ldots+\frac{1}{n} = O(\ln n)}$$

Hence, by using the approximation by integrals, $1+\frac{1}{2}+\frac{1}{3}+\ldots+\frac{1}{n} = O(\ln n)$

(b)

e) $f(n)=\log 2n$; $g(n)=\log 3n$

$f(n)=\log 2n=\log 2+\log n$

$g(n)=\log 3n=\log 3+\log n$

$$\lim_{n\to\infty}\frac{f(n)}{g(n)}=\lim_{n\to\infty}\frac{\log 2+\log n}{\log 3+\log n}=\lim_{n\to\infty}\frac{\frac{\log 2}{\log n}+1}{\frac{\log 3}{\log n}+1}$$

$$\lim_{n\to\infty}\frac{\log 2}{\log n}=\lim_{n\to\infty}\frac{\log 3}{\log n}=0$$

$$\lim_{n\to\infty}\frac{f(n)}{g(n)}=1$$

Since $0<1<\infty$, $f(n)=\Theta(g(n))$.□

f) $f(n)=10\log n$; $g(n)=\log(n^2)$

$$\lim_{n\to\infty}\frac{f(n)}{g(n)}=\lim_{n\to\infty}\frac{10\log n}{\log(n^2)}=\lim_{n\to\infty}\frac{10\log n}{2\log n}=\lim_{n\to\infty}5=5$$

Since $0<5<\infty$, $f(n) = \Theta(g(n))$. □

k) $f(n) = \sqrt{n}$; $g(n) = (\log n)^3$

Using the fact that a polynomial always grows faster than a poly-logarithmic, we have:

$f(n) = \sqrt{n} = n^{1/2}$

So $\sqrt{n} = n^{1/2} = \Omega\left((\log n)^3\right)$

Hence, $f(n) = \Omega(g(n))$. □

m) $f(n) = 2^n$; $g(n) = 2^{n+1}$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{2^n}{2^{n+1}} = \lim_{n\to\infty} \frac{2^n}{2\cdot 2^n} = \lim_{n\to\infty} \frac{1}{2} = 0.5$$

Since $0 < 0.5 < \infty$, $2^n = \Theta(2^{n+1})$

Hence, $f(n) = \Theta(g(n))$

o) $f(n) = n!$; $g(n) = 2^n$

We have $n! = 1\cdot 2\cdot 3\cdot \ldots \cdot n = 2\cdot 3\cdot \ldots \cdot n \geq \underbrace{2\cdot 2\cdot \ldots \cdot 2}_{n-1 \text{ times}}$

Then $n! \geq 2^{n-1}$ or $n! \geq \frac{1}{2}\cdot 2^n$ for all $n \geq 2$

Therefore, $f(n) = \Omega(g(n))$

(2) **Question 2:** Solve for big-O of the following recurrences. The answer should be as tight as possible.

- $T(n) = T(2n/3) + O(1)$.
- $T(n) = 7T(n/7) + O(n)$.
- $T(n) = 7T(2n/3) + O(n^2)$.
- $T(n) = T(n-2) + O(n)$.

• $T(n) = T(2n/3) + O(1)$

$a = 1$

$b = \frac{3}{2}$

$d = 0 = \log_{\frac{3}{2}} 1 \longrightarrow T(n) = O(n^d \log n)$

$T(n) = O(n^0 \log n)$

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

Hence, $\boxed{T(n) = T(2n/3) + O(1) = O(\log n)}$. □

• $T(n) = 7T(n/7) + O(n)$

$a = 7$

$b = 7$

$d = 1 = \log_7 7 \longrightarrow T(n) = O(n^d \log n)$

$T(n) = O(n^1 \log n)$

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

Hence, $\boxed{T(n) = 7T(n/7) + O(n) = O(n \log n)}$. □

• $T(n) = 7T(2n/3) + O(n^2)$

$a = 7$

$b = 3/2$

$d = 2 < \log_{\frac{3}{2}} 7 \simeq 4.799 \longrightarrow T(n) = O(n^{\log_b a}) = O(n^{\log_{3/2} 7})$

4.2 Master theorem

Theorem 1. *If the recurrence is in the following form*

$$T(n) = aT(n/b) + O(n^d).$$

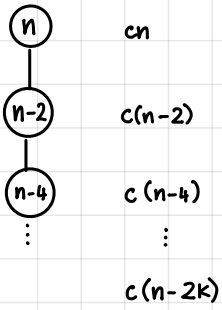
Then,

$$T(n) = \begin{cases} O(n^d) & \text{if } d > \log_b a \\ O(n^d \log n) & \text{if } d = \log_b a \\ O(n^{\log_b a}) & \text{if } d < \log_b a. \end{cases}$$

The proof is a formalization of the above examples. Read section 2.2 in the book for a proof.

Hence, $T(n) = 7T(2n/3) + O(n^2) = O(n^{\log_{3/2} 7})$. □

- $T(n) = T(n-2) + O(n)$.



$n - 2K = 0 \rightarrow K = \frac{n}{2}$ → There are at most $\frac{n}{2}$ levels.

Running Time $= cn + c(n-2) + c(n-4) + \dots + c(n-2k) + \dots$

$= \sum_{k=0}^{n/2} (n-2k) = c\sum_{k=0}^{n/2} n - 2c\sum_{k=0}^{n/2} k = c\cdot\frac{n^2}{2} - 2c\cdot\frac{\frac{n}{2}\left(\frac{n}{2}+1\right)}{2} = \frac{cn^2}{2} - c\left(\frac{n^2}{4} + \frac{n}{2}\right) = \frac{cn^2}{4} - \frac{cn}{2}$ (c: constant)

$= O(n^2)$

Hence, $T(n) = T(n-2) + O(n) = O(n^2)$. □

(3) **Question 3:** Problem 37 on page 64 of [Erickson]. You can assume the tree is represented by a linked data structure. Each node has a pointer to its left child, right child, and its parent. For example, if v is a node, then $v.left$, $v.right$, $v.parent$ point to its left child, right child, and parent respectively. If $v.left = NULL$, that means v does not have a left child (and similarly for $v.right$ and $v.parent$).

(4) **Question 4:** Problem 2.17 of [DPV]

Trees

37. For this problem, a *subtree* of a binary tree means any connected subgraph. A binary tree is *complete* if every internal node has two children, and every leaf has exactly the same depth. Describe and analyze a recursive algorithm to compute the *largest complete subtree* of a given binary tree. Your algorithm should return both the root and the depth of this subtree. See Figure 1.26 for an example.

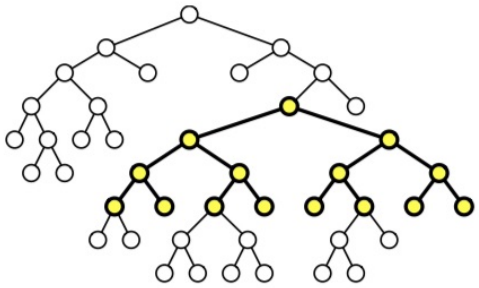
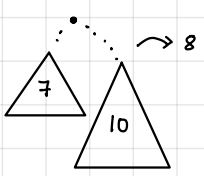


Figure 1.26. The largest complete subtree of this binary tree has depth 3.

Finding the largest complete subtree:

↳ MaxTree(v) = minimum{MaxTree(v.left); MaxTree(v.right)} + 1



Function: MaxTree(v)

```
if v.left == NULL or v.right == NULL
    depth[v] = 0
    return depth[v]
else
    MaxTree(v.left) = a
    MaxTree(v.right) = b
```

depth [v] = min {a,b} + 1

return depth [v]

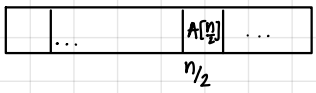
MaxTree (root)

Return the root and depth [v] which is largest.

* Running Time = O(n) since call MaxTree(v) once

(4) **Question 4:** Problem 2.17 of [DPV].

2.17. Given a sorted array of distinct integers $A[1, \ldots, n]$, you want to find out whether there is an index i for which $A[i] = i$. Give a divide-and-conquer algorithm that runs in time $O(\log n)$.



Compare $A[\frac{n}{2}]$ vs $\frac{n}{2}$ ⟶ 3 cases ⟶ if we have $\frac{n}{2} < A[\frac{n}{2}]$, there will be an index in $A[1 \ldots \frac{n}{2}-1]$

Algorithm searching for i = A[i]: search(i, K)

```
if i == K :
    return true
else :
    j = (i+K)/2
    if j < A[j]:
        search (i, j-1)
    else:
        search (j, K)
```

↳ Call search (1, n)

Running Time: $T(n) = T(\frac{n}{2}) + O(1)$

Apply Master theorem: $a=1, b=2, d=0$

$d = \log_b a = \log_2 1 = 0 \rightarrow T(n) = O(n^d \log n) = O(n^0 \log n) = O(\log n)$

(5) **Question 5:**

- Use induction to prove that $\sum_{i=0}^{n} i2^i = 2 + (n-1)2^{n+1}$.
- Use induction to prove that for any natural number n, $5^n - 1$ is divisible by 4.

(a) Prove $\sum_{i=0}^{n} i2^i = 2 + (n-1)2^{n+1}$: S(i)

- Base case: i=0

S(0): Left side = $0 \times 2^0 = 0$

Right side = $2 + (0-1)2^{0+1} = 2 - 2 = 0$

Then $0 \times 2^0 = 2 + (0-1)2^{0+1}$

- Induction hypothesis: Suppose S(k) is true or $\sum_{i=0}^{k} i2^i = 2 + (k-1)2^{k+1}$

↳ We want to show when $n = k+1$ is also true.

Adding $(k+1) \cdot 2^{k+1}$ to both sides of S(k)

Left side : $\sum_{i=0}^{k} i \cdot 2^i + (k+1) \cdot 2^{k+1} = \sum_{i=0}^{k+1} i2^i$

Righ side : $2 + (k-1)2^{k+1} + (k+1) \cdot 2^{k+1}$

$= 2 + (k - 1 + k + 1)2^{k+1}$

$= 2 + (2k)k^{k+1}$

$= 2 + k \cdot 2^{k+2}$

$= 2 + [(k+1) - 1]2^{(k+1)+1}$

Therefore, $\sum_{i=0}^{k+1} i2^i = 2 + [(k+1) - 1]2^{(k+1)+1}$. ⟶ S(k+1) is True.

Hence, $\sum_{i=0}^{n} i2^i = 2 + (n-1)2^{n+1}$.□

(b) Prove that for any natural number n, $5^n - 1$ is divisible by 4.

- Base case: n=1

$5^n - 1 = 5^1 - 1 = 5 - 1 = 4$ divisible by 4.

- Induction hypothesis: Assume $5^k - 1$ is divisible by 4
- When $n = k+1$, we have :

$5^{k+1} - 1 = 5^k \cdot 5 - 1 = 5^k(4+1) - 1 = 5^k \cdot 4 + (5^k - 1)$

Since $4 \mid (5^k - 1)$ and $4 \mid (4 \cdot 5^k)$

Then $4 \mid (5^{k+1} - 1)$

Hence, for any natural n, $5^n - 1$ is divisible by 4.□