CPSC 221: Theory Assignment #2

Eduardo
Goiza
r1v0b

(1) Use contradiction to prove x^5 is not $O(x^2)$

Given $T(x) = x^5$, suppose $T(x) \in O(x^2)$.

Then there must exist a function $g(x)$ such that

$$T(x) \leq c \cdot g(x), \quad \forall x \in \mathbb{R}, \ x \geq x_0$$

this means that

$$x^5 \leq c \cdot x^2,$$

Right away we see a contradiction as

$x < x^2 < \ldots x^5$ and

x^2 is not greater than x^5.

Therefore arriving at a contradiction, and proving that $T(x) = x^5 \notin O(x^2)$. ■

(2) Prove $1^3 + 2^3 + \ldots n^3$ is $\Theta(n^4)$

Let $T(n) = 1^3 + 2^3 + \ldots n^3$, a geometric series can be simplified (approximated) to the following function:

$$T(n) = \frac{1}{4} n^2 (n+1)^2 = \frac{1}{4} n^4 + \frac{1}{2} n^3 + \frac{1}{4} n^2.$$

Given the Theorem of Polynomial Orders (from textbook EPP), it is stated that given

$f(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots a_1 x + a_0$, then

$f(x) \in \Theta(x^n)$ when no bounds are specified.

Therefore the highest degree of $T(n) = \frac{1}{4} n^4 + \frac{1}{2} n^3 + \frac{1}{4} n^2$ is degree 4. and it states that

$$T(n) \in \Theta(n^4),$$

proving our initial equation and finishing our proof. ■

(3) Consider:

$E_1 = 0$, $E_K = E_{K-1} + (K+1)$, $\forall K \in \mathbb{Z}, K > 1$

a) Use substitution at each iteration to find an explicit formula

$E_K = E_{K-1} + (K+1)$

$\downarrow$

$= (E_{K-2} + K) + (K+1)$

$\downarrow$

$= [(E_{K-3} + (K-1)) + (K)] + (K+1)$

$= E_{K-3} + (K-1) + K + (K+1) \sim E_{K-3} + 3 \cdot K + c$

$= E_{K-n} + n \cdot K + c$ and when $K - n = 1$

$(K-1) = n$

$\downarrow$

$E_K = E_1 + (K-1) \cdot K + \cancel{c}$ constant is neglible

and $E_1 = 0$

$$\boxed{E_K = K^2 - K}$$

b) Use induction to verify correctness of formula

Given $E_K = K^2 - K$.

(1) Let our base case be $K = 1$

$E(1) = 1^2 - 1 = 0$ ✓ true because we know $E_1 = 0$.

(2) Suppose $E_K = K^2 - K$ for all $K = n$ such that $E_n = n^2 - n = E_{n-1} + (n+1)$ then we must prove the E_{n+1} case

Let $K = n+1$: then for

$E_{n+1} = E_n + (n)$

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

$E_{n+1} = n^2$ and $E_n = E_{n-1} + (n+1)$

therefore $E_{n+1} = E_n + (n+1)$

$n^2 - (n+1) = E_n$ we arrive at our conclusion where ...

... our proof is valid for $(K+1)$ items.

Quicksort

5	3	2	8	1	0	6	7	4

where pivot median of {first element, middle, last element}
and
middle is $k = \lceil (i+j)/2 \rceil$ index $i = 1\text{st}$ $j = \text{last}$

Given: 5 3 2 8 1 0 6 7 4

1) choose pivot {5, 1, 4} ⇒ 4, swap to place in first element.
2) reorder so all els. < and > are to LEFT and right.

→ 4 ‖ 3 2 8 1 0 6 7 5

piv. ↑ wall ↑ i →

when x[i] < x[wall] : swap.

⇓

4 2, 3 ‖ 8, 1, 0, 6, 7, 5

↑ wall ↑ i (continue ... until end).

4, 2, 3, 1, ‖ 8, 0, 6, 7, 5
↑ i

4, 2, 3, 1, 0, 8, 7, 5 (done partition) : now swap 'wall' with pivot
↑ wall.

0, 2, 3, 1, 4, 8, 7, 5

0, 2, 3, 1
└ pivot = 2 call w/ Qs(A, 0, piv − 1)

8, 7, 5 call w/ Qs(A, piv + 1, end)
└ pivot = 7

⇓

5, 7, 8 (after partition)

0, 1
└ middle: 1
pivot = 1

3
low = high
return ✓

5
low = high
return ✓

8
low = high
return ✓

0
return

therefore 0 1 2 3 4 5 6 7 8

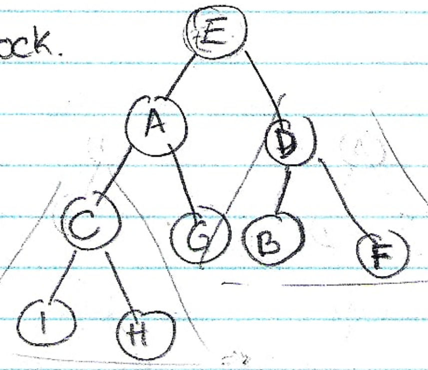
(5)

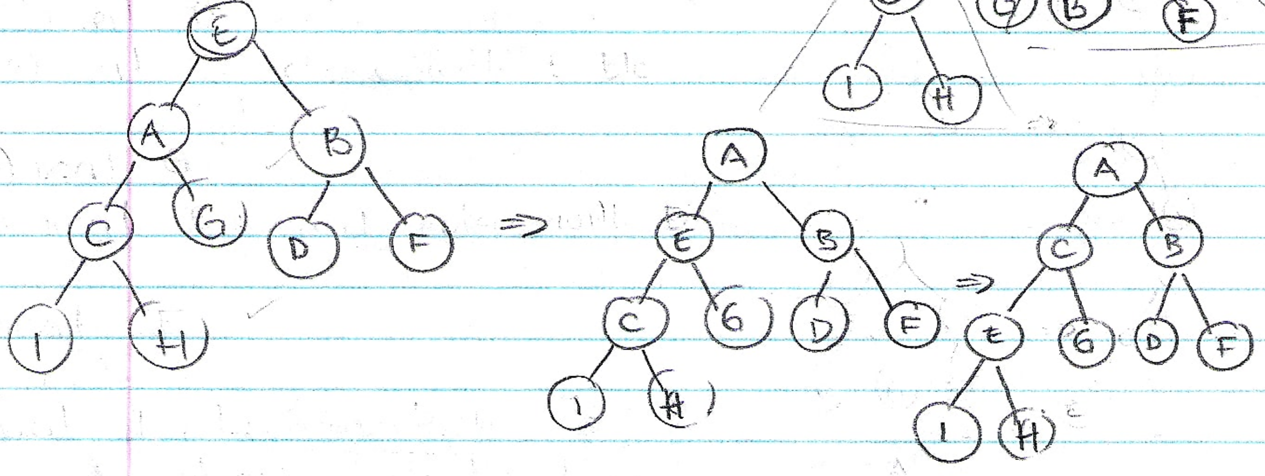
E	A	D	C	G	B	F	I	H

a) Convert to min heap using Heapify ⟶ O(n)

1) Insert linear-tree to 1st free block.

2) call swap down last parent.
moving back towards root

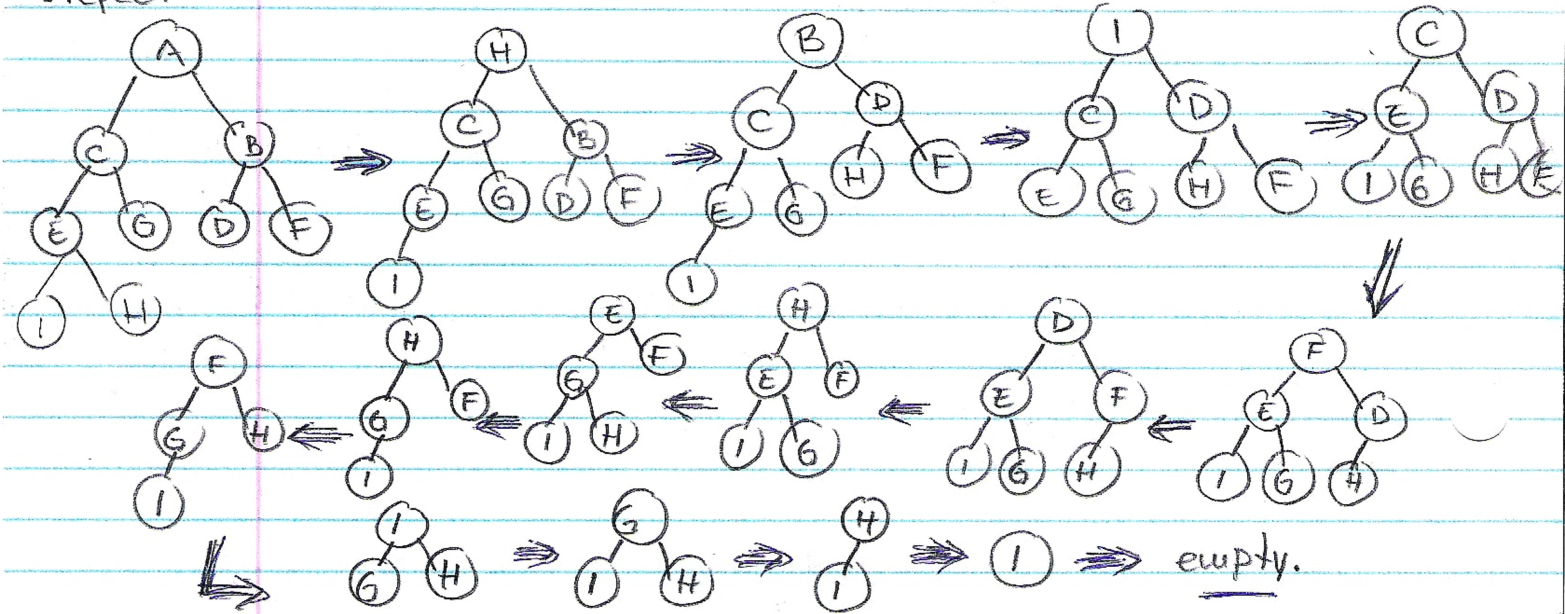




b) Perform Heapsort

1) remove top item and replace w/ last element. store in array
2) reestablish heap
3) repeat.

A	B	C	D	E	F	G	H	I



(6) Number of Keys Algorithm in O(K) time

```
printKeys(int i, key q, int size)
    if (i < size and H[i] <= q) then
        cout << H[i] << endl;
        printKeys(2*i + 1, q, size)
        printKeys(2*i + 2, q, size)
    endif
```

Prints keys that match requirements with recursive calls and of worst O(K) time.
If H[i] <= q is not met, a branch dies off and we do not explore any values in subheap because we know they will all be greater than its parent.

(7) Prove correctness w/ respect to pre-/post-conditions by using the loop invariant: $x * y + \text{product} = A*B$

Given pre-condition where $x = A$, $y = B$, product = 0.
We must prove our invariant that:
— $A \times B = x * y + \text{product}$ —

@ beginning (pre-loop): $A \times B = x \times y + \emptyset$ ✓

Because (A and B) $\in \mathbb{Z}$, $(A, B) > 0$. then our loop will run at least once.

in loop: $r = y \% 2 \longrightarrow r = (B \% 2)$: r is either EVEN or ODD

r (EVEN): $x = 2 * x \longrightarrow x = 2 \cdot A$
$y = y/2 \longrightarrow y = (B/2)$
where $A \times B = (2A) \cdot (B/2) + \emptyset$ ✓ preserved in loop

r (ODD): $\text{product} = \text{product} + x \longrightarrow \text{product} = 0 + A$
$y = y - 1 \longrightarrow y = B - 1$
$A \times B = (A) \cdot (B-1) + (0 + A) \Rightarrow A \times B = A \times B$

therefore, because our loop invariant is preserved at all locations, it proves the correctness of our loops pre-/post conditions ☒

(8) Use an invariant to prove that pow is equal to a^n

```
i=1
pow=1
while (i<=n) do:
    pow *= a
    i++;
endloop
```

[loop invariant
pow <= a^n]

@ beginning: pow=1 and a, n are parameters passed to our function.
We verify that pow <= a^n because $(pow)_0 = i = 1$ and $i <= n$, according to the validity of our loop.

@ during loop: pow is initially '1' and changes as follows:

pow=a	i=1
pow=a*a	i=2
⋮	⋮
pow=a^n	i=n

Therefore pow <= a^n at all points in our loop,
and, as we proved all scenarios, our loop invariant validates the ~~st~~ post-condition: pow = a^n ∎

~~(10) Compare & choose best hash function for 11-digit account number~~

(11) Prove a ternary tree height h has at most $(3^{h+1}-1)/2$ nodes.

(1) Given empty tree: h=-1. so $1-1/2=0$.
and h=1 is $3^2-1/2=4$ nodes (root & children)

(2) suppose a ternary tree height h has $(3^{h+1}-1)/2$ nodes,
then lets prove (h+1) has $(3^{h+2}-1)/2$ nodes.

Let height be h, and n be number of nodes.

$n = \frac{(3^{h+1})}{2}$ then (h+1) has n+x nodes where x is the extra nodes added.

$(n+x) = \frac{3^{h+2}}{2} = \frac{(3^{h+1})\cdot 3}{2}$ and because $\frac{3^{h+1}}{2} = n$

$(n+x) = n \cdot 3$

So given $n=0$, $x=0$: when no nodes
OR when $n \neq 0$,
then we add 3 more nodes. ∎