Connie Ma
56047129 // p6z8
CPSC 221 202
16.3.17

pg 1

THEORY ASSIGNMENT #2

PART 1 – Complexity

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

Proof by contradiction: i. Suppose it is the case that x^5 is $O(x^2)$.

$\Rightarrow \exists\, c, n_0$ s.t. $\cancel{x}\, x^5 \le cx^2 \;\forall\, x > n_0$ (definition of O)

ii. But $x^5 \le cx^2 \Rightarrow x^3 \le c$, and we can choose some x (e.g. $x=c$) s.t. $x^3 \nleq c \Rightarrow$ CONTRADICTS the original assumption.

iii. $\therefore x^5$ is not $O(x^2)$. ∎

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

A. Proof by induction: i. Let $T(n) = 1^3+2^3+3^3+\ldots+n^3$, we can prove $T(n) = \left(\frac{n(n+1)}{2}\right)^2$

ii. Base case $T(1)$: $1^3 = \left(\frac{1(1+1)}{2}\right)^2$

$1^3 = \left(\frac{2}{2}\right)^2 \Rightarrow 1=1$ ✓

iii. Induction hypothesis: $T(k) = \left(\frac{k(k+1)}{2}\right)^2$

(where ☆ $T(k) = 1^3+2^3+3^3+\ldots+k^3$ for $k \in \mathbb{Z}$)

iv. We have $T(k+1) = \underbrace{1^3+2^3+3^3+\ldots+k^3}+(k+1)^3$

LHS

by ☆, we can substitute this w/ $T(k)$

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

~ALGEBRA~

$= \left(\frac{(k+1)((k+1)+1)}{2}\right)^2$ = RHS

$\therefore$ By math induction we have shown that $T(n) = \left(\frac{n(n+1)}{2}\right)^2$

~ALGEBRA~

$\left(\frac{k(k+1)}{2}\right)^2 + \underbrace{(k+1)^3}_{\text{factor}}$

$= \frac{k^2(k+1)^2}{4} + \underbrace{(k+1)^2(k+1)}_{\times 4}$

$= \frac{4(k+1)(k+1)^2 + k^2(k+1)^2}{4}$

$= \frac{(k+1)^2(k^2+4k+4)}{4}$ factor out $(k+1)^2$

$= \frac{(k+1)^2(k+2)^2}{4} = \left(\frac{(k+1)((k+1)+1)}{2}\right)^2$

factor

B. Big Θ proof: Given $T(n) = 1^3+2^3+3^3+\ldots+n^3 = \left(\frac{n(n+1)}{2}\right)^2$

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

Big-O: $\rightarrow \frac{1}{4}n^4 + \frac{1}{2}n^3 + n^2 \le \frac{3}{4}n^4 + n^2 \le \frac{7}{4}n^4 \quad \therefore T(n) \in O(n^4)$ w/ $c = 7/4$, $n_0 = 1$

Big Ω: $\rightarrow \frac{1}{4}n^4 + \frac{1}{2}n^3 + n^2 \ge \frac{1}{4}n^4 \quad \therefore T(n) \in \Omega(n^4)$ w/ $c = 1/4$, $n_0 = 1$

$c, n_0 \in \mathbb{R}^+$

$\therefore T(n) \in \Theta(n^4)$ $c = 1/4$, $n_0 = 1$ ∎

Connie Ma
56047129 // p6z8
CPSC 221 202
16.3.17

THEORY ASSIGNMENT #2

pg 2

PART 2 — Recurrences

3 Consider the following recurrence relation: $E_1 = 0$, $E_k = E_{k-1} + k + 1 \quad \forall k > 1, k \in \mathbb{Z}$

a) Use substitution to find explicit formula for the sequence.

$E_1 = 0$

$E_2 = 0 + k + 1 = k+1$

$E_3 = (k+1) + k + 1 = 2k+2$

$E_4 = (2k+2) + k + 1 = 3k+3$

$\boxed{E_n = (n-1)(k+1)} \quad \forall n > 0, n \in \mathbb{Z}$ (all ~~the~~ integers)

b) Use induction to verify correctness of the formula.

Proof by induction → i. Let $E(n) = E(n-1) + k + 1$ derived from given recurrence relation

we wish to prove $E(n) = (n-1)(k+1)$ ← formula from a)

ii. Base case: $E(1) = ((1)-1)(k+1) = (0)(k+1) = 0 = E_1$ ✓

iii. Induction hypothesis:

we have → ☆ $E(i) = E(i-1) + k + 1 = (i-1)(k+1) \quad \forall i \in \mathbb{Z}, i > 0$

then → $E(i+1) = E((i+1)-1) + k + 1$ plug in $i+1$

LHS $= \underbrace{E(i)}_{\text{substitute ☆}} + k + 1 = (i-1)(k+1) + k + 1$

$= ik + i - \cancel{k} - \cancel{1} + \cancel{k} + \cancel{1}$ expand expression

$= i(k+1) = ((i+1)-1)(k+1) = \text{RHS}$ $\quad \forall n \in \mathbb{Z}, n > 0$

iv. by math induction, we have shown that $E(n) = E(n-1) + k + 1 = (n-1)(k+1)$ ▨

PART 3 — Quicksort

4 Using Bentley's Quicksort alg on

5	3	2	8	1	0	6	7	4

a) Find first pivot = median$(x[lo], x[hi], x[\frac{lo+hi}{2}])$ = median$(5, 4, 1)$ = 4 ← swap w/ lo
($x[0]$ $x[8]$ $x[4]$)

so we have

4	3	2	8	1	0	6	7	5

lo = 0, hi = 8, p = element x[0] = 4

~~array~~ after $\boxed{\text{qsort}(x_0[], 0, 8)}$ ⇒

0	3	2	1	4	8	6	7	5

$x_1[]$ p $x_2[]$

b) Find pivots for $x_1[]$, $x_2[]$

$p_1 = \text{med}(0, 1, 2) = 1$; $p_2 = \text{med}(8, 5, 7) = 7$

so we have

1	3	2	0	4	~~8~~ 7	6	8	5

p_1 $x_1[]$ p p_2 $x_2[]$

i. array after $\boxed{\text{qsort}(x_1[], 0, 3)}$ ⇒

0	1	2	3	4	7	6	8	5

→ x[] is still passed every time

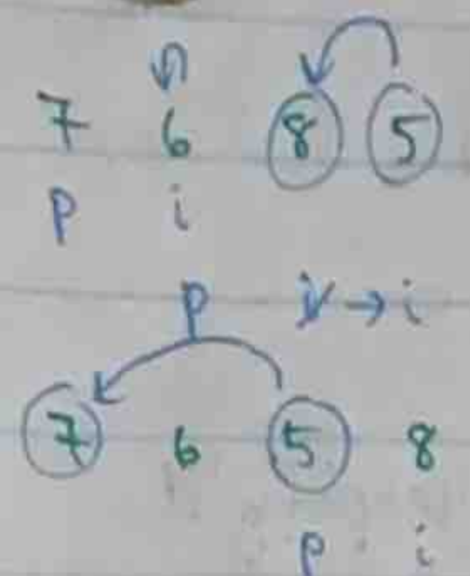
~~★NOTE~~: $X_0[]$, $X_1[]$ etc are just used to denoted which partition of the same array X[] we are

4 3 2 8 1 0 6 7 5
p i
p i
p i
4 3 2 1 8 0 6 7 5
p i
4 3 2 1 0 8 6 7 5
p i → i → i → i
1 3 2 0
p i → i → i
1 0 2 3
p i → i

Connie Ma
56047129 // p6z8
CPSC 221 202
16.3.17

THEORY ASSIGNMENT #2

pg 3



PART 3 — Quicksort [4 cont'd]

4 b) ii. array after $\boxed{qsort(x_2[], 5, 8)}$ ⇒

≡ SORTED ≡

0	~~1~~	2	3	4	5	6	7	8
$x_3[]$	P_1	$x_4[]$		P	$x_5[]$		P_2	$x_6[]$

c) i. array does not change after $\boxed{qsort(x_3[], 0, 0)}$ b/c 0=0

ii. array does not change after $\boxed{qsort(x_4[], 2, 3)}$ b/c ~~2~~ 2 < 3 (p < i)

iii. same holds for $\boxed{qsort(x_5[], 5, 6)}$ & $\boxed{qsort(x_6[], 8, 8)}$

☆ NOTE again that $x[] = x_0[] = x_1[] = \ldots = x_n[]$; I am just using subscripts to keep track of sorting

⇒ Actual order of execution of qsorts:

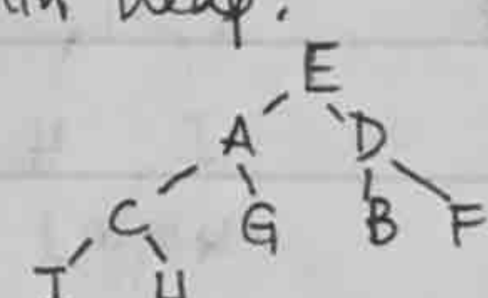
1. qsort(x[], 0, 8)
2. qsort(x[], 0, 3) ret
3. qsort (x[], 0, 0) ret
4. qsort (x[], 2, 3) ret
5. qsort (x[], 2, 2) ret
6. qsort (x[], 4, 3) ret
7. qsort(x[], 5, 8) ret
8. qsort(x[], 5, 6)
9. qsort (x[], 5, 5) ret
10. qsort (x[], 7, 6) ret
11. qsort (x[], 8, 8) ret

PART 4 — Heaps

5 a) Converted the following unordered array to a min heap.

0	1	2	3	4	5	6	7	8
E	A	D	C	G	B	F	I	H

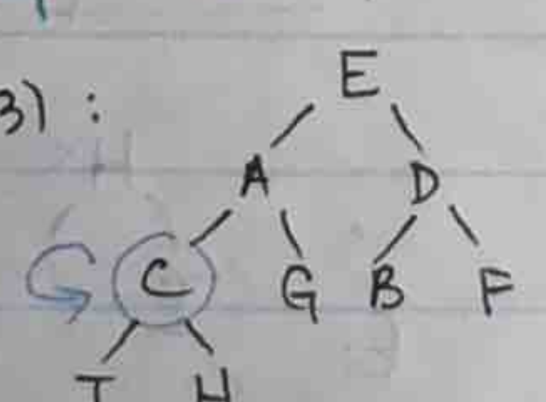
=



☆ using heapify alg provided in lab4, where for loop has ⇒ for (int i = (size-2)/2 ; i >= 0; i--) (size/2 - 1)

after swapDown (3):

C < I, H
⇒ no change

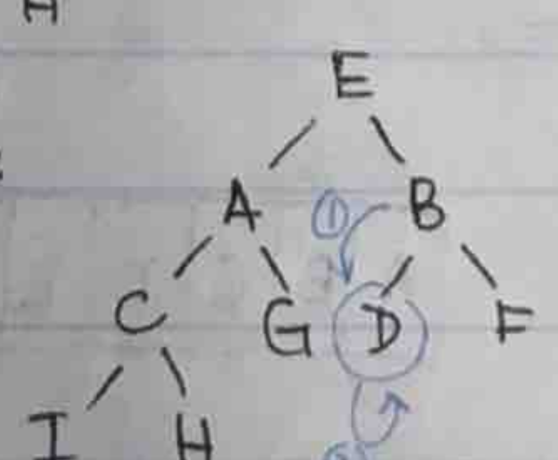


⇒

E	A	D	C	G	B	F	I	H

after swapDown (2):

① D > B, B < F
⇒ swap D & B
② B < D, F
⇒ no change



⇒

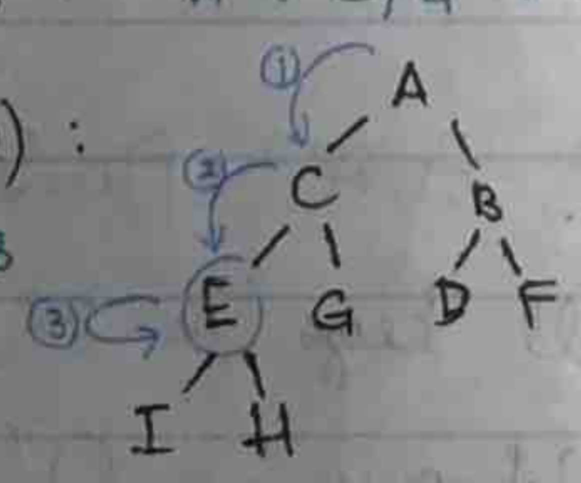
E	A	B	C	G	D	F	I	H

after swapDown (1): A < C, G ⇒ no change

same as this

after swapDown (0):

① E > A, B ; A < B
⇒ swap E & A
② E > C , C < G



③ E < I, H ⇒ no change

⇒

A	C	B	E	G	D	F	I	H

☆ result after heapify completed

Connie Ma
50047129 // p6z8
CPSC 221 202
16.3.17

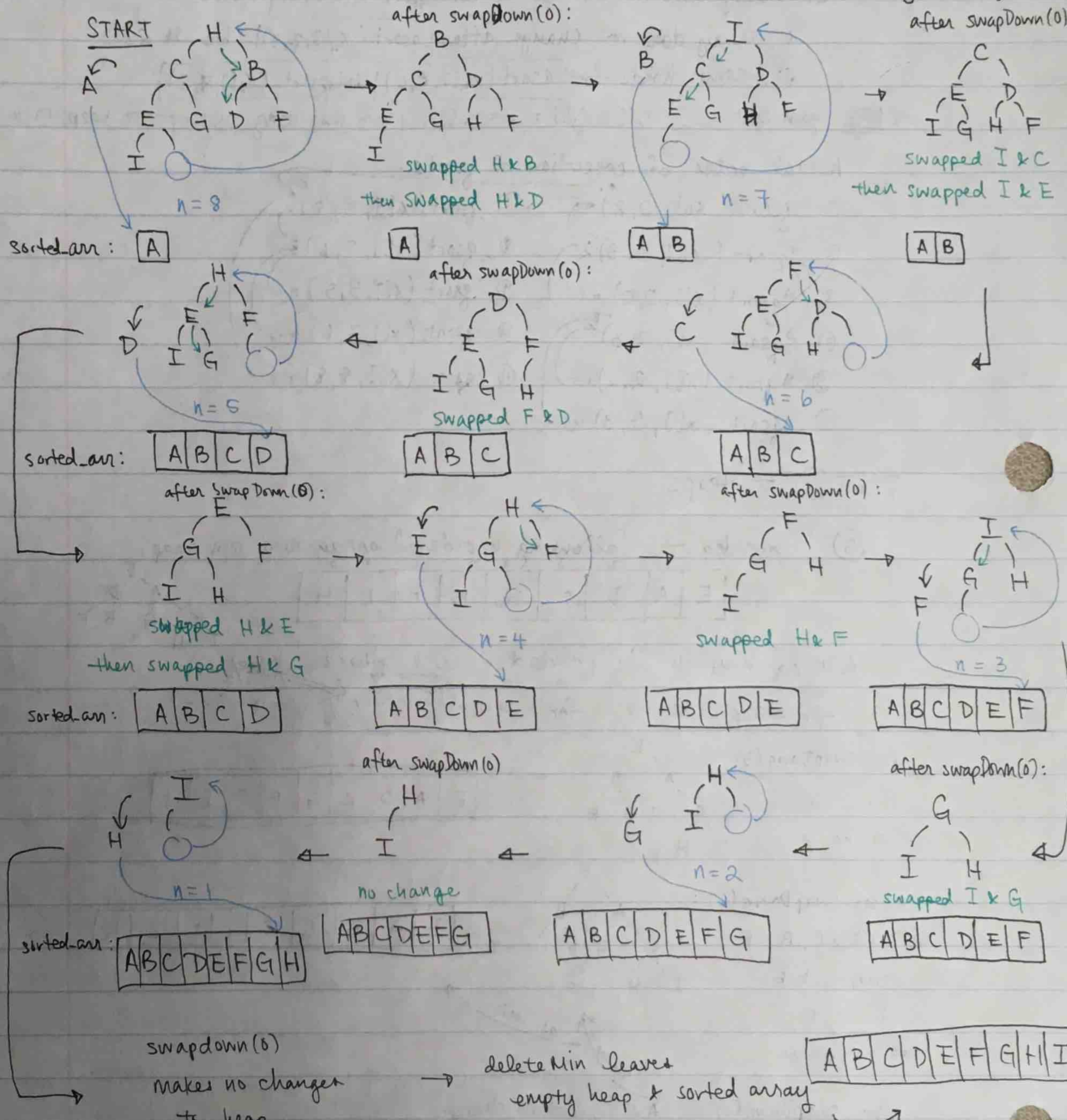
THEORY ASSIGNMENT #2

PART 4 – Heaps [5 cont/d]

5 b) Perform Heapsort on the heap in (a). ⇒ | A | C | B | E | G | D | F | I | H | $n=9$

→ $n = 9$, so we perform deleteMin 9 times.

→ after each deleteMin, we need only swapDown(0) since we take minHeap[n-1] and put it in the top, & rest of heap is already Heapify'd



6 Consider (min heap) H w/ n keys. Give efficient algorithm for reporting all keys in H that are <= to given query key q. Should run in O(k) time, where k is num keys reported

Connie Ma
56047129 // p6z8
CPSC 221 #202
16.3.17

THEORY ASSIGNMENT #2

Pg 5

PART 4 – Heaps [6 cont/d]

6 pseudocode heap H, size n, query key q

```
void printSmallerKeys ( int k, int q) {
  //base cases: make sure k is an index in range of H
              and that it is not > q, otherwise
              we know that its children will all be > q
  if ( k >= n || H[k] > q) return;
  // print/report the key value if it is <= q
  if ( H[k] <= q)   cout << " H[k] << " ";
  printSmallerKeys ((k*2)+1 , q);  //recurse on left child
  printSmallerKeys ((k*2)+2, q);   // recurse on right child
}
```

minHeap invariant

PART 5 — Program Correctness & Loop Invariants

7 Precondition: A & B are +ve integers, x=A, y=B, product=0

```
while (y!=0)
  r = y%2;
  if(r=0)   x *= 2 ; y /= 2 ;
  if (r=1)  product += x ; y-- ;
```

Post condition: product = A * B

→ Prove correctness of loop wrt PreC & PostC using loop invariant "x*y + product = A*B"

Approach: This algorithm computes how many times x must add x to itself, which is given by y. W/ that in mind, looking at r = y%2 we see that in each iteration of the loop y must be even or odd. ⇒ do a proof by cases to cover all iterations ⇒ proves loop invariant

Direct proof → i. Case: y even (r=y%2=0) ⇒ we have x = x*2 and y = y/2

x=A, y=B, product=0 plug-in precondition to loop invariant x*y + product → $\underbrace{(A*2)}_{x} * \underbrace{(B/2)}_{y} + \underbrace{0}_{product} = A*B\left(\frac{2}{2}\right) = A*B$ ✓

ii. Case: y odd (r=y%2=1) ⇒ we have product = product + x and y = y−1

plug-in precond again → $\underbrace{A}_{x} * \underbrace{(B-1)}_{y} + \underbrace{(0+A)}_{product} = A*B - A + A = A*B$ ✓

iii. We have shown for all cases that the loop invariant holds. ∎

Connie Ma
56047129 // p6z8
CPSC 221 202
16.3.17

THEORY ASSIGNMENT #2

PART 5 cont'd

8) Use a loop invariant to prove that when the following algorithm terminates, pow is equal to a^n:

```
i=1, pow=1;
while (i ≤ n)
  { pow = pow * a; i++; }
```

a)

★ Termination: The loop always terminates because i increases by 1 every iteration and will eventually $= n$.

b)

★ Loop invariant: $pow = a^i$ that always holds @ end of each iteration

Proof by induction →

i. Base case: $i=1$, $pow=1$

→ we have $pow = pow * a$

→ $pow = (1) * a = a = a^1 = a^{i=1}$ ✓

ii. Inductive step: (induction hypothesis = loop invariant)

We have from the base case & loop invariant that

$i=1 \Rightarrow pow = a$

$\Rightarrow$ pow is always divisible by a

$\Rightarrow$ pow will always $= a^{\text{some power}} = a^i$

For each loop we increment i at the same time we multiply pow by a $\Rightarrow$ $pow = a \cdot a^i$ each time

★ LHS $a \cdot a^i \overset{\text{add exponents}}{=} a^{i+1} = RHS$

loop invariant $(i) \Rightarrow$ loop invariant $(i+1)$

∴ by induction we have shown that the loop invariant holds ✓.

★ c) We have proven the loop invariant holds and that the loop terminates when $i=n$, therefore when the loop terminates

$\Rightarrow pow = a^{i=n} = a^n$

$\Rightarrow pow = a^n$ @ loop termination.

∎

Connie Ma
56047129 // p6z8
CPSC 221 202
16.3.17

THEORY ASSIGNMENT #2

pg 7

PART 7 — Trees & Induction

(11) A ternary tree is either empty or consists of root & left, middle, right subtrees. Prove that ternary tree of height h has at most $(3^{h+1}-1)/2$ nodes, by using induction on the height of the tree. Note that empty tree $h=-1$.

→ Let maxNodes (h) be function that returns the maximum nodes of a ternary tree of height h.

→ By counting nodes, we wish to prove that: $\sum_{i=0}^{h} 3^i$

$$\text{maxNodes}(h) = 1 + 3^1 + 3^2 + \ldots + 3^h = \frac{3^{h+1}-1}{2}$$

Proof by induction:

i. Base case, $h=0$ or $h=1$

$$\text{maxNodes}(0) = \frac{3^{(-1)+1}-1}{2} = \frac{3^0-1}{2} = \frac{1-1}{2} = 0 \checkmark$$ no nodes

$$\text{maxNodes}(1) = \frac{3^{1+1}-1}{2} = \frac{3-1}{2} = \frac{2}{2} = 1 \checkmark$$ root node

ii. induction hypothesis → ☆ Let $\text{maxNodes}(k) = 1 + 3^1 + 3^2 + \ldots + 3^k = \frac{3^{k+1}-1}{2}$ be true $\forall k \in \mathbb{Z}$, $k > 1$

LHS

$$\text{Then maxNodes}(k+1) = \underbrace{1 + 3^1 + 3^2 + \ldots + 3^k}_{} + 3^{k+1}$$

substitute in $\text{maxNodes}(k) = \frac{3^{k+1}-1}{2}$ here ☆

$$\rightarrow \frac{3^{k+1}-1}{2} + 3^{k+1}$$

$$= \frac{3^{k+1}-1+2(3^{k+1})}{2}$$ combined the expression ~~added to same~~

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

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

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

iii. We have shown by induction on the height of the ternary tree
∴ that the maximum number of nodes for a ternary tree of height $h = \frac{3^{h+1}-1}{2}$.