CS435 HW2 - Anarch Abraham

1) $T(n) = \begin{cases} aT(n/b) + cn^\beta, & n>1 \\ d & n=1 \end{cases}$; $h = \log_b a$; $T(n) = \begin{cases} An^h + Bn^\beta = \Theta(n^\beta), & h<\beta \\ An^h + Bn^\beta = \Theta(n^h), & h>\beta \\ An^h + Bn^\beta \log n = \Theta(n^h \log n), & h=\beta \end{cases}$

a) $T(n) = 2T(n/2) + 1$

$= 2T(n/2) + 1 \cdot n^0 \Rightarrow a=2, b=2, \beta=0, c=1 \Rightarrow h = \log_b a = \log_2 2 = 1$ → $h>\beta$

$T(n) = An^1 + Bn^0 = An + B = \Theta(n^h)$ ←

$= \Theta(n)$

b) $T(n) = 2T(n/2) + n \Rightarrow a=2, b=2, \beta=1, c=1 \Rightarrow h = \log_b a = \log_2 2 = 1$ → $h=\beta$

$T(n) = An^1 + Bn^1 \log n = \Theta(n^h \log n)$ $= \Theta(n \log n)$

c) $T(n) = 2T(n/2) + n^2 \Rightarrow a=2, b=2, \beta=2, c=1 \Rightarrow h = \log_2 2 = 1$ → $h<\beta$

$T(n) = An^1 + Bn^2 = \Theta(n^\beta) = T(n)$ $= \Theta(n^2)$

d) $T(n) = T(n/2) + 1 \Rightarrow a=1, b=2, \beta=0, c=1 \Rightarrow h = \log_2 1 = 0$ → $h=\beta$

$T(n) = An^0 + Bn^0 \log n = A + B\log n$ $= \Theta(\log n)$

2) $T(1)=0$, $T(n) = 4T(n/2) + n$, $n \geq 1$

a) $T(n) = 4T(n/2) + n = 4(4T(n/4) + n/2) + n = 4^2 T(n/2) + 2^1 n + 2^0 n$

$= 4^3 T(n/2^3) + 2^2 n + 2^1 n + 2^0 n$

$= 4^k T(n/2^k) + 2^{k-1} n + 2^{k-2} n + \ldots + 2^3 n + \ldots + 2^1 n + 2^0 n \Rightarrow$ have $2^k = n$, & take log

$k = \log n \Rightarrow 4^k = 2^{2k} = 2^2 \log n$

↳ $T(n) = O(n^2)$

b) $a=4, b=2, f(n)=n \Rightarrow n^{\log_b a} = n^{\log_2 4} = n^2 \Rightarrow$ $T(n) = \Theta(n^2)$

3) U.B. $f(n) \leq n\log n$ (logs in base 2)

$f(n) = \begin{cases} 2f(n/2)+n-1, & n>1 \\ 0, & n=1 \end{cases} \Rightarrow f(n) \leq n\log n \ \forall n \in \mathbb{N}$

Ind: Base case $\Rightarrow n=1 \rightarrow f(n) = 0 \leq 1 \cdot \log(1) = 0$ ✓

Suppose $f(k) \leq k\log k \ \forall \ k<n \rightarrow$ know, $f(n) = f(n/2)+n\cdot 1$

by I.H. get $f(n/2) \leq n/2 \log(n/2)$ so $f(n) \leq n/2 \log(n/2) + n - 1$

$\leq n/2 [\log(n) - \log 2] + n - 1$

$= n/2 \log(n) - n/2 + n - 1 = n/2 \log(n) + n/2 - 1$

$\leq n/2 \log(n) + \log(n) = n\log n \Rightarrow$ then, $\boxed{f(n) \leq n\log n \ \forall n \geq 1}$

4) Prove sol $f(n) = \lfloor \log_2 n \rfloor + 1$ by ind. $\Rightarrow f(n) = \begin{cases} 1, & n=1 \\ f(\lfloor n/2 \rfloor)+1, & n \geq 2 \end{cases}$

Prove $n=1 \Rightarrow f(1) = 1 \rightarrow \log_2 1 + 1 = 0+1 = 1$ ✓

↳ $n=k \Rightarrow f(k) = \lfloor \log_2 k \rfloor + 1$ is true, then prove $n = k+1 \Rightarrow f(k+1) =$

$= f(\lfloor (k+1)/2 \rfloor) + 1$

$= \lfloor \log \lfloor (k+1)/2 \rfloor \rfloor + 1 + 1$

$= \lfloor \log \lfloor (k+1) \rfloor - \log 2 \rfloor + 1 + 1 = \lfloor \log \lfloor k+1 \rfloor \rfloor - 1 + 1 + 1$

$= \lfloor \log \lfloor k+1 \rfloor \rfloor + 1 = \lfloor \log(k+1) \rfloor + 1$ where $k+1 = n$ ✓

7)

5) $f(n) = \begin{cases} f(\lceil n/2 \rceil) + f(\lfloor n/2 \rfloor) + n - 1, & n \geq 2 \\ 0, & n = 1 \end{cases}$

Prove $f(n) \leq n\lceil \log n \rceil \Rightarrow$ prove $n=1 \rightarrow f(1)=0 \Rightarrow 1\lceil \log 1 \rceil = 0$ ✓

$\Rightarrow$ prove $n=2 \rightarrow f(2) = f(\lceil 2/2 \rceil) + f(\lfloor 2/2 \rfloor) + 2 - 1 = f(1) + f(1) + 1 = 2f(1) + 1 = 2 \cdot 0 + 1 = 1$

$\hookrightarrow n\lceil \log n \rceil = 2\lceil \log 2 \rceil = 2\lceil 1 \rceil = 2 > 1 \Rightarrow 2\lceil \log 2 \rceil > f(2)$ ✓ for $n=2$

Now let $f(n) \leq n\lceil \log n \rceil$ be true for $n = k > 2$ for $k \in \mathbb{N}$.

Then $f(k) \leq k\lceil \log k \rceil$ holds $\rightarrow$ So prove it's true for $n = k+1$

$\Rightarrow f(k+1) = f(\lceil k/2 \rceil) + f(\lfloor k/2 \rfloor) + k$ ~~$+1$~~ $= f(\lceil k/2 \rceil) + f(\lfloor k/2 \rfloor) + k$

$\Rightarrow f(\lceil k/2 \rceil) \leq \lceil k/2 \rceil \lceil \log \lceil k/2 \rceil \rceil \Rightarrow f(\lfloor k/2 \rfloor) \leq \lfloor k/2 \rfloor \lceil \log \lfloor k/2 \rfloor \rceil$

then $f(k+1) = f(\lceil k/2 \rceil) + f(\lfloor k/2 \rfloor) + k$

$\leq \lceil k/2 \rceil \lceil \log \lceil k/2 \rceil \rceil + \lfloor k/2 \rfloor \lceil \log \lfloor k/2 \rfloor \rceil + k$ $(\lceil \log \lfloor n/2 \rfloor \rceil \leq \lceil \log \lceil n/2 \rceil \rceil)$ ✓

for any integer $n \geq 2$, $\lceil \log \lceil n/2 \rceil \rceil = \lceil \log (n/2) \rceil = \lceil \log n \rceil - 1$

Therefore $f(k+1) \leq \lceil k/2 \rceil \lceil \log \lceil k/2 \rceil \rceil + \lfloor k/2 \rfloor \lceil \log \lfloor k/2 \rfloor \rceil + k$

$\leq \lceil k/2 \rceil \lceil \log \lceil k/2 \rceil \rceil + \lfloor k/2 \rfloor \lceil \log \lceil k/2 \rceil \rceil + k$

(known, $\lceil k/2 \rceil + \lfloor k/2 \rfloor = k$)

Therefore $f(k+1) \leq k\lceil \log \lceil k/2 \rceil \rceil + k$

$= k(\lceil \log k \rceil - 1) + k$

$= k\lceil \log k \rceil - k + k$

$= k\lceil \log k \rceil$

$\leq (k+1)\lceil \log (k+1) \rceil$ ($k\lceil \log k \rceil$ is Inc. function)

Therefore $f(n) \leq n\lceil \log n \rceil$ true for all n

6) b)
```
void PUSHDOWN(dtype A[], int r, int n){
  while (true){
    int left = 2*r;
    int right = 2*r+1;
    int min;
    if (left <= n && A[left] < A[r])
      min = left;
    else
      min = r;
    if(right <= n && A[right] < A[min])
      min = right;
    if(min == r)
      break;
    dtype temp = A[r];
    A[r] = A[min];
    A[min] = temp;
    r = min;
  }
}
```

a)
```
void PUSHDOWN (dtype A[], int r, int n){
  int left = 2*r;
  int right = 2*r+1;
  int min;
  if (left <= n && A[left] < A[r])
    min = left;
  else
    min = r;
  if (right <= n && A[right] < A[min])
    min = right;
  if (min != r){
    dtype temp = A[r];
    A[r] = A[min];
    A[min] = temp;
    PUSHDOWN(A, min, n);
  }
}
```

$f(n) = \begin{cases} 2f(n/2) + \log n, & n > 1 \\ 0, & n = 1 \end{cases}$

7) a) Best case: $n = 2$

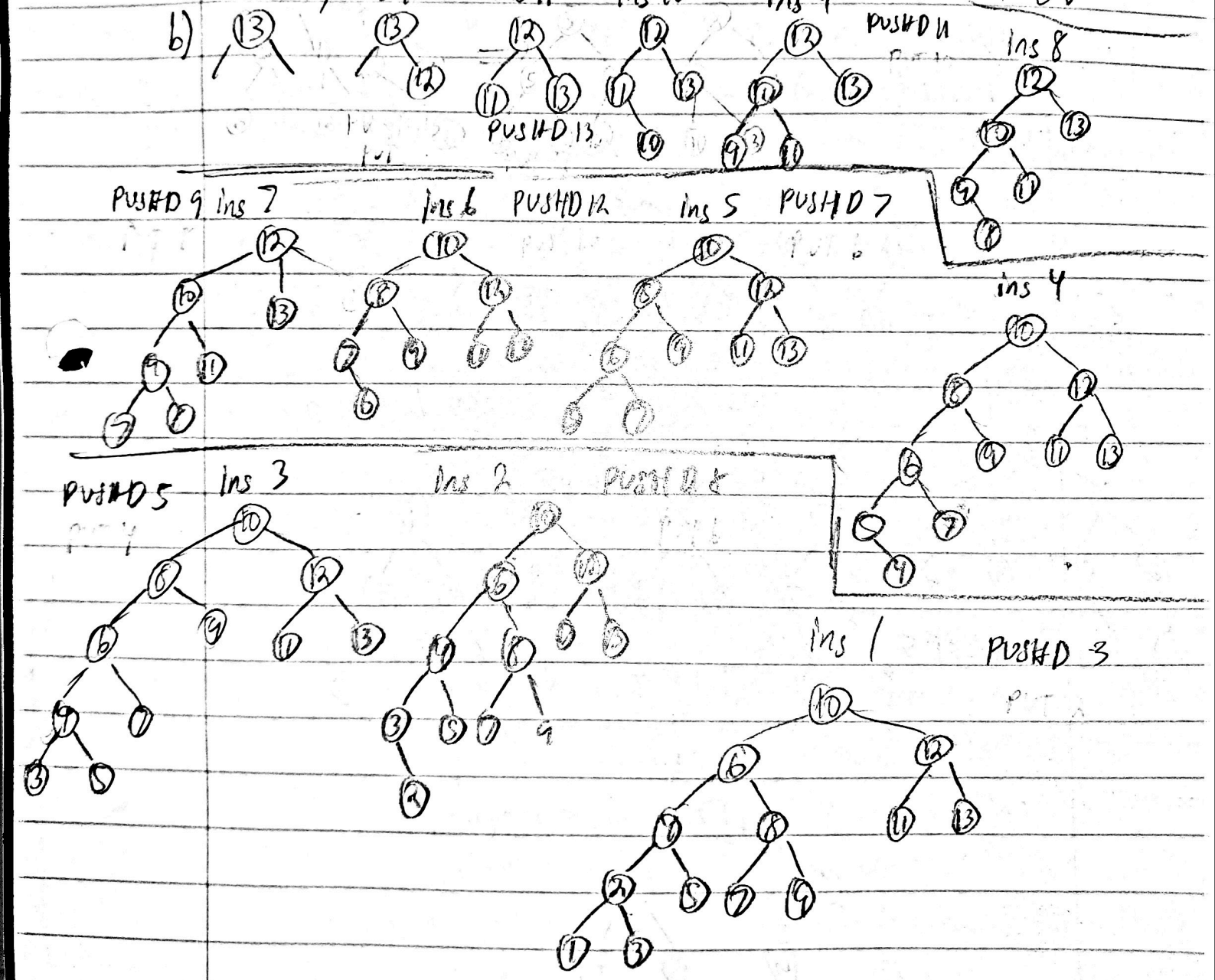
$f(2) = 2f(2/2) + \log 2 = 2f(1) + 1 = 2 \cdot 0 + 1 = 1$ ✓

IH: $f(n) = An + B\log n + C$ true for all $n = k > 2$

IS: $n = k+1 \Rightarrow f(k+1) = 2f((k+1)/2) + \log k$

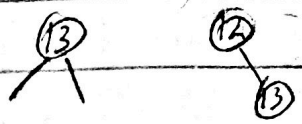
$= Ak/2 + B\log k/2 + C = At + Bt + C \Rightarrow \therefore, f(n) = Ak + B\log k + C$ is true ✓

b)

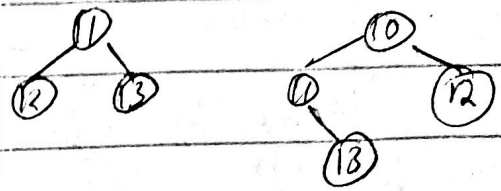


c) BUILD-HEAP(dtype A[], int r, int n){
int s;
if(2r<n && (A[2r] > A[r])) s=2r;
else s=r;
if(2r+1<s && A[2r]>A[s]) s=2r;
if(s!=r) swap(A[r], A[s])
BUILDHEAP(A,s,n)
PUSHDOWN(A,r,n)

d) ins 13 ins 12 PUSHD 13

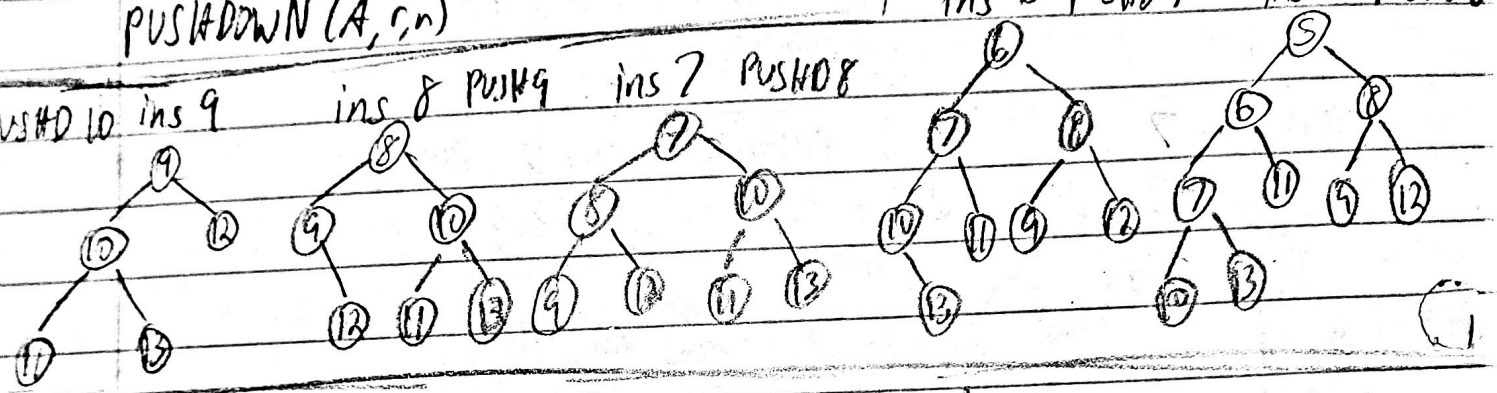


ins 11 PUSHD 12 ins 10 PUSH 11

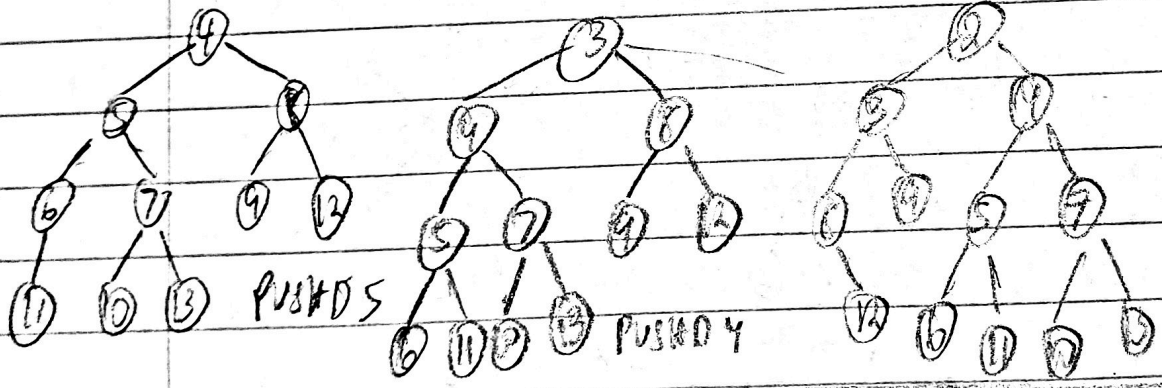


ins 6 PUSHD 7 ins 5 PUSHD 6

PUSHD 10 ins 9 ins 8 PUSH 9 ins 7 PUSHD 8



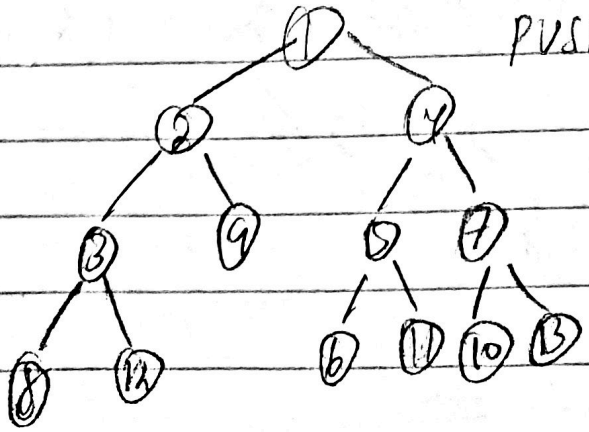
ins 4 ins 3 ins 2 PUSHD 3



PUSHD 5 PUSHD 4

ins 1

PUSHDOWN 2



8) a) left child of node i is 2i and right child is 2i+1 w/ parent being $\lfloor i/2 \rfloor$

⇒ total # nodes in full bin tree w/ i internal nodes is $n = 2i+1$

Base case: A tree w/ 0 internal nodes has 1 node together, A tree w/ 1 internal node has 3 nodes together. Thus base case proved for $i \geq 0 \leq i \leq 1$,

I.S. Assume any full bin tree w/ i internal nodes has $2i+1$ total nodes.

Prove any full binary tree w/ i+1 internal nodes has $2(i+1)+1$ leaves.

Let T be full bin tree w/ i+1 internal. T has at least 1 internal node, so must have at least 2 leaves. Choose any leaf v at max depth. so v's sibling also leaf at max depth. Now, remove v & it's sibling and let T' be resulting tree. v's parent/sibling is an internal node in T but a leaf in T'. where T' is a full bin tree w/ i internal nodes. This tree must contain $n = 2i+1$ total nodes through I.H. Adding leaf v and it's respective siblings to the parent gives i+1 internal nodes and $(2i+1)+2 = 2i+3$ total nodes. that equals $2(i+1)+1$.

b) total # nodes, N(h) as function of h ⇒ $N(h) = 1+2+4+\ldots+2^h = 2^{h+1}-1$

where height h in full bin tree as function of n is ⇒ $h(n) = \log(n+1) - 1 = \log\frac{n+1}{2}$

c) let N(h) be function defined above where h(n) have h as function of n.

$h(n) = 2^h = \left(\frac{n+1}{2}\right)$

d) INSERT (data type A[], int n, int height) ⇒ time complexity is O(log n) where worst case is new element goes to root (Θ(height)). # comparisons/swaps is log n, where n is # of elements after insert.

```
n = n+1;
A[n] = key; p = n;
while (p > 1 && A[p] < A[parent(p)]) {
    swap(A[p], A[parent(p)]);
    p = parent(p);
}
```

9) e) root node of heap w/ n nodes has smallest elements, thus making it easy to get. However, the resulting heap needs to be into a heap w/ one less element. Then remove last node/copy key to empty root node temporarily. Time complexity for DELETEMIN is O(log n). Worst case # comparisons is 2 comparisons per binary level. ⇒ 2⌊log n⌋, where n is # elements after delete.

9) a) leaf nodes in terms of n ⇒ $\frac{n+1}{2}$

Base case: Tree w/ 1 node has single leaf node & full binary tree w/ 3 nodes w/ 2 leaf nodes. (base case is n=1) ✓

I.S. Assume full tree w/ x nodes such that it has 2 subtrees w/ one left, right, w/ x & y nodes where n = x+y+1. The leaf node for tree will be a leaf node from Left or right tree. From Ind. hypothesis ⇒

⇒ (x+1)/2 + (y+1)/2 = (x+y+2)/2 = (n+1)/2

b)
```
LBRANCH(root) {
  if(root == null) return 0;
  LDepth = LBRANCH(root.left())
  RDepth = LBRANCH(root.right())
  if(LDepth > RDepth) return LDepth+1;
  else return RDepth+1;
}
```

c) Time complexity for 1 branch is O(n) b/c of tree traversals time 2 for left/right w/ traversal being O(n). This traversal is O(n) b/c the max # of total edges in binary tree w/ n nod is n-1. Thus overall complexity in time to traverse tree is O(n+n-1) ≈ O(n)