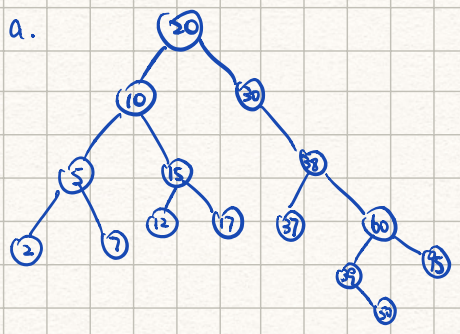
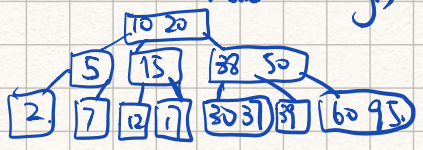
2014 Theory II

1. a.

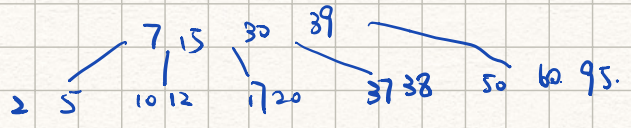


BST.

B-tree order 1. min 1 key
max 2 keys



B-tree order 2 min 2 keys
max 4 keys



b.

```
int getRequestedValue (int x) {
    int max = -INT_MAX; (any small enough value)
    int root = 0; (set root = 0)
    bool flag = FALSE; (not encountered x yet)
    inOrder(root, x, &max, &flag); (pass max and flag by reference)
    if (max == INT_MAX) return nil; else return max;
}

void inOrder (int root, int x, int *max, bool* flag) {
    if (root != NULL && *flag == FALSE) {
        inOrder (left[root], x, max, flag)
        if (value[root] == x) {
            *flag = TRUE (encountered x, no more predecessors)
            return;
        }
        if (value[root] < x && x - value[root] < x - *max) *max = value[root] (keep max updated)
        inOrder(right[root], x, max, flag);
    }
    return;
}
```

c. B-Tree of order n and height 3.

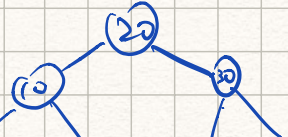
B-Tree only increase height when it is full upper 2 lvls

$$\min(n,3) = 1 + (n+1) + (n+1)^2 = n^2 + 2n + 1 + n + 1 + 1 = n^2 + 3n + 2.$$

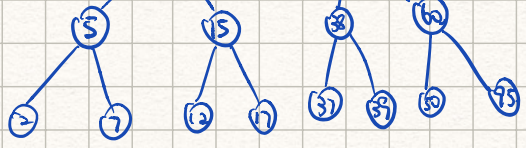
$$\max(n,3) = 1 + (2n+1) + (2n+1)^2 = 4n^2 + 4n + 1 + 2n + 1 - 1 = 4n^2 + 6n + 2.$$

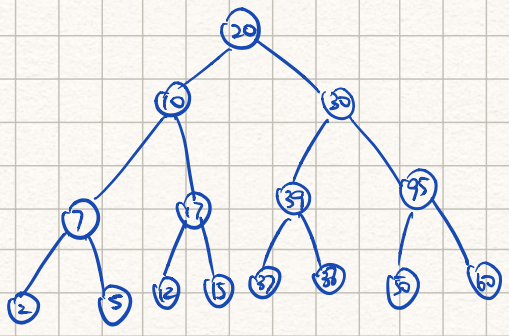
$$\text{max\_leaf} = (2n+1)^2 \times 2n.$$

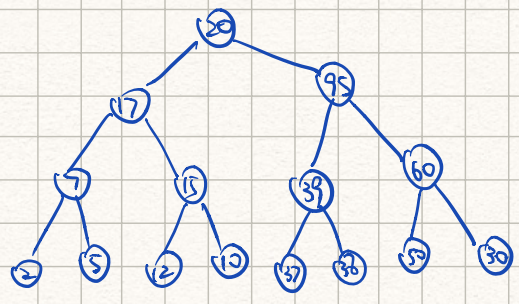
d.

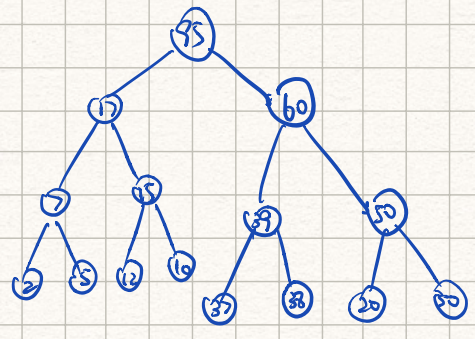


Assume building a max heap









2. a.

b. int TreeSum(l, r, x, k) {

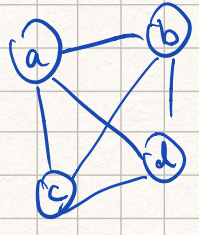
if (l(x)==nil && r(x)==nil) return 0;

else if (x==nil) return 0;

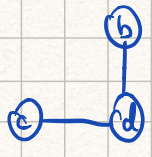
else return k(x)+TreeSum(l, r, l(x), k)+TreeSum(l, r, r(x), k);

}.

c. i. G.



G'



ii. $2^{|A|}$

iii. $2^{|N|}$

3. a. $h(st) = st[1]$ ← take the number.

0	1	2	3	4	5	6	7	8	9	10	11	12
	A1	B1	E2	G1	D5	F5	C7	H5				

B1 → B1

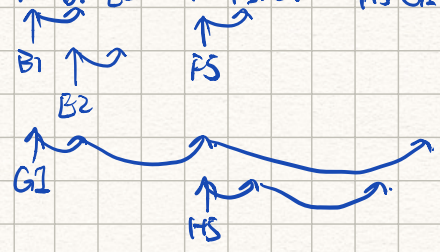
E2 → E2

D5 → F5

(clustering)

G1 HS

0	1	2	3	4	5	6	7	8	9	10	11	12
	A1	B1	E2		DS	PS	O1		HS	G1		

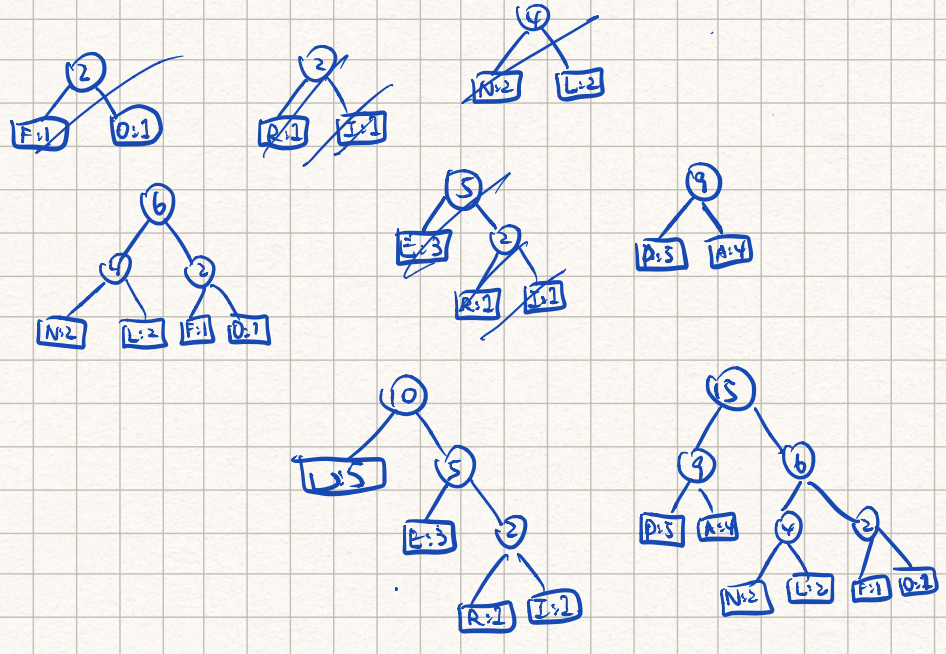


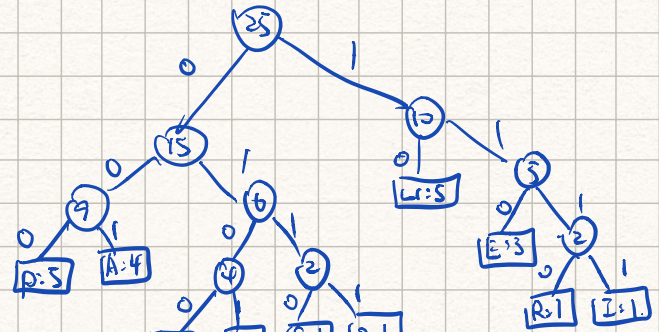
0	1	2	3	4	5	6	7	8	9	10	11	12
	A1	E2			DS		O1					
	B1				PS							
	G1				HS							

b.

A N P L E F O R I
A N P L E
A P E
A P
P

~~5~~ ~~A 4~~ ~~N 2~~ ~~P 5~~ ~~L 2~~ ~~E 3~~
F ~~1~~ ~~O 1~~ ~~R 1~~ ~~I 1~~





N:2 L:2 F:1 O:1

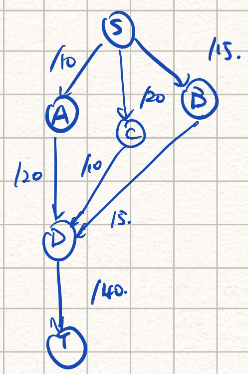
␣ 10 L: 0101 E: 110

A: 001 N: 0100 R: 1110

P: 000 F: 0110 I: 1111

O: 0111

c.



i. {D-T.}

{A→D, C→D, B→D}

{SA SB SC}

mix of the two

ii. for a super traversal if all forward traversal has a unfilled capacity of x

and all backward ... filled capacity of x.

x>0

iii. SA CD BD

10+10+5=25

minimum cut is how much can flow from

最大流最小割. S→T, minimal cut = maximum capacity

4. a. $f(n) \in O(g(n))$.

$\exists$ sufficiently large k and a const. c.

s.t $\forall n \geq k$ $f(n) \leq c \cdot g(n)$

b. let $f(n) \in O(\log_a n)$.

$f(n) \leq c \cdot \log_a n = c \cdot \frac{\log_b n}{\log_b a} = \frac{c}{\log_b a} \cdot \log_b n.$

$c' = c \times \frac{1}{\log_b n}$. $f(n) \leq c' \cdot \log_b n$.

$\therefore f(n) \in O(\log_b n)$ vice versa.

c. i. $a(n) \in O(n)$

$b(n) = n^2 + n + 4 \in O(n^2)$. $a(n)$ is better

ii. $4n+8 > n^2+n+4$

$0 > n^2-3n-4$

$0 > (n-4)(n+1)$ $\therefore 0 \le n < 4$ $b(n)$ is better

$n = 4$ $a(n) = b(n)$

$4 < n$ $a(n)$ is better

d. false

true.

false.

false

5. a. best case is determined by the problem itself

worst case --- algo itself.

b. decision tree?

c. basic operation / machine independent

d. (i) $\underbrace{1+3+5+\cdots+2n+1}_{n+1 \text{ terms.}}$ $\frac{(2n+2)(n+1)}{2} = (n+1)^2 = n^2+2n+1.$ $O(n^2)$

(ii) $n^2-2+n^2-4+\cdots+n^2-2n$

$= n^2 \times n - (2+4+\cdots+2n)$

$= n^2 \times n - 2(1+2+\cdots+n)$

$= n^3 - 2\frac{(1+n)\cdot n}{2} = n^3-n^2-n.$ $O(n^3)$.

(iii). $2\times1+1 + 2\times1+2+\cdots+2\times1+n.$

$+ 2\times2+\cdots \qquad +2\times2+n$

$+$

$+ 2\times n+1 + \cdots\cdots\cdots + 2\times n+n.$

$2\times\frac{(1+(n+1))(n+1)}{2} \times n + n+2n+\cdots+n^2.$

$= (n+2)(n+1)n + \frac{n(1+n)\cdot n}{2}$ $O(n^3)$

6. a) $a_1 = 4a_0 - 3.$

$a_2 = 4(4a_0-3)-3 = 4^2a_0 - 3\cdot4 - 3.$

$a_3 = 4^3a_0 - 3\cdot4^2 - 3\cdot4 - 3.$

$a_n = 4^n a_0 - 3\cdot(1+4+\cdots+4^{n-1})$

$= 2\cdot4^3 \approx O(4^n)$

recurrence

$f(n) - 4f(n-1) + 3f(n-2) = 0.$

$x^2 - 4x + 3 = 0$

$(x-3)(x-1)=0$

$f(n)=a\,3^n+b\,1^n$

$f(0)=a+b=2$. $\quad a=1 \quad b=1$.

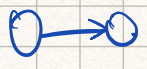
$f(1)=3a+b=4$.

$f(n)=3^n+1^n \qquad O(3^n)$

Theor II.

1. a. there is a path (regardless of direction) from one node to another

strongly connected for a directed graph, there is a route for every one node to another.



2. a directed graphe with no cycles in it.

a tree is a DAG. since no cycles for obvious reasons

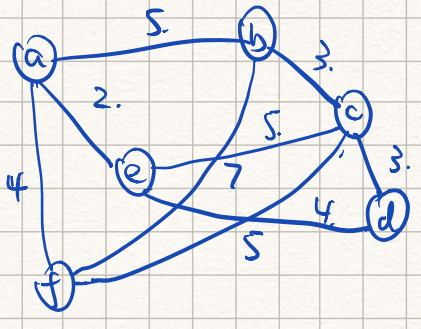
a DAG is not necessarily a tree

represent workflows / dependancies

DFS from any node. if every node is visited (not twice)

then it's ok.

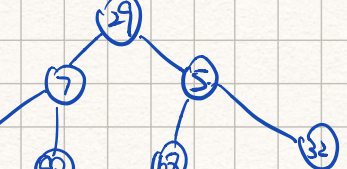
e.



3. a. DP used when have same subproblems that needs to be solved only once

88 subproblems $\longrightarrow$ solution of whole problem

b.



i) 29 7 78 12 47 98 89 41

5 68 48 32.

(ii) 29 7 5 78 98 68 32
12 47 89 41 48.

(iii) 29 7 78 12 47 — DFS ii)

(iii) 12 78 47 7898 41 29

48 68 5 32.

(iiii) 12 47 78 89 41 98 7

48 68 32 5 29