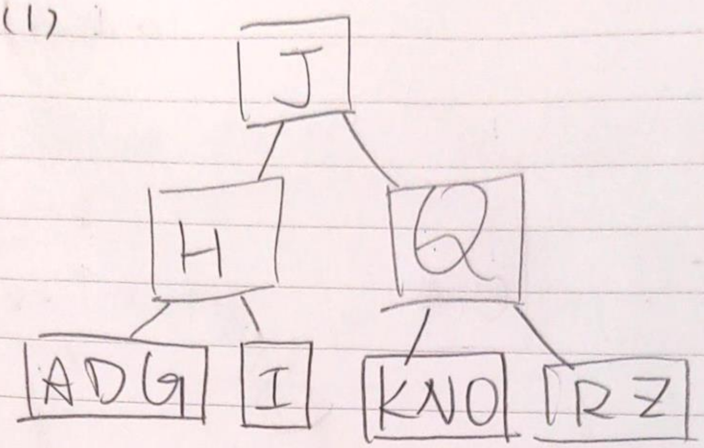
TITLE/NO.

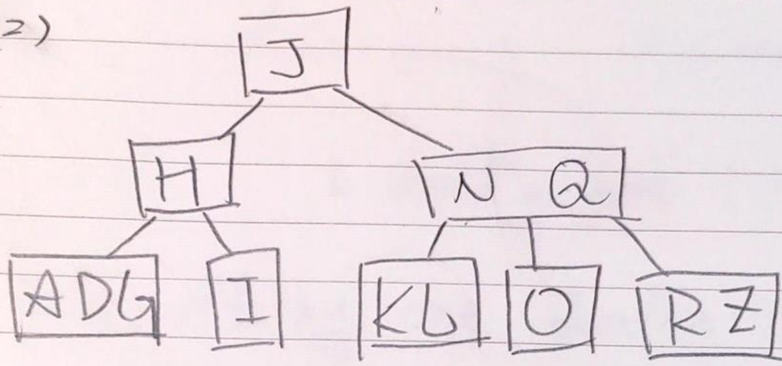
M T W T F S S

Refs: W16 B-Tree Video Part 10

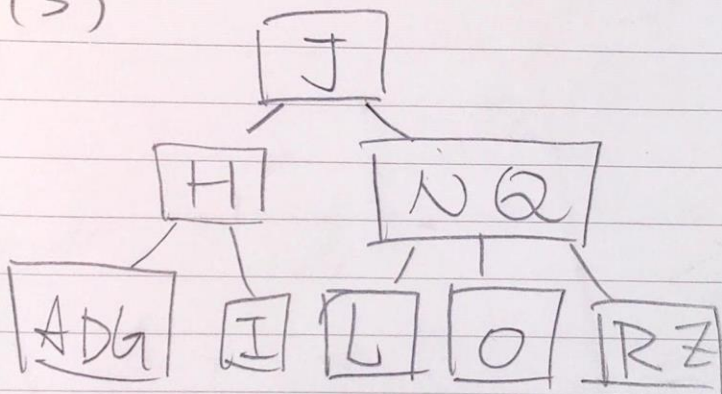
1. (1)



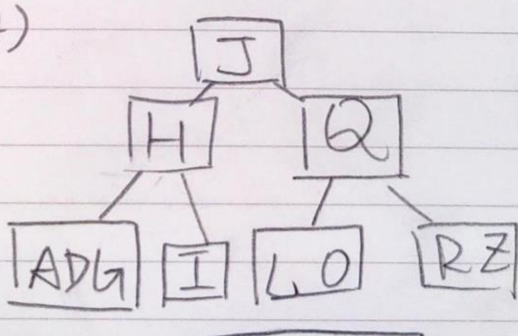
(2)



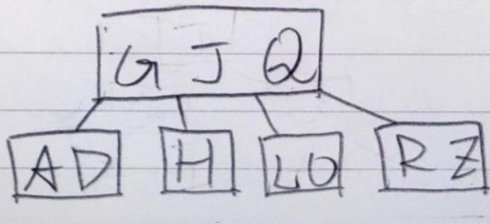
(3)



(4)



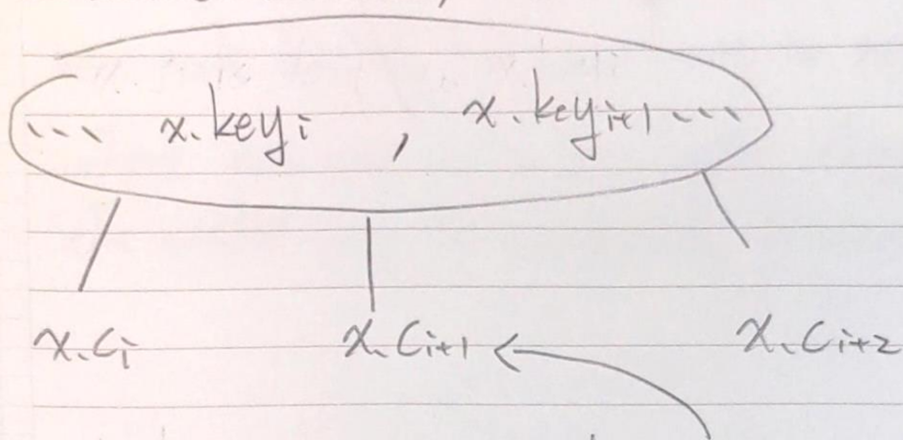
(5)



Refs: None

2. No.

Before $i = i+1$, it looks like



k should go here $\because x.key_i \le k < x.key_{i+1}$

It's off by one, therefore $i = i+1$ is needed

TITLE/NO.

M T W T F S S

Refs: None

3、No.

If the root needs to be splited, allocating a new node to be the new root is needed. But when non-root nodes are split, they just throw the middle key to the parent. Operations are different.

4.

Refs: None

```
function BFS_MOD(G)
    for each vertex u in G.V
        u.color = WHITE
        u.d = 1
        u.pi = NIL
    cc_cnt = 0
    for s in G.V
        if s.color = WHITE
            cc_cnt += 1
            s.color = GRAY
            s.d = 0
            s.pi = NIL
            Q = empty_queue()
            Enqueue(Q s)
            while Q ≠ empty_queue()
                u = Dequeue(Q)
                for each v in G.Adj[u]
                    if v:color == WHITE
                        v.color = GRAY
                        v.d = u.d + 1
                        v.pi = u
                        Enqueue(Q, v)
                u:color = BLACK
```

The first for loop runs $|V|$ times, so it's $O(|V|)$

The inner most for loop runs $|E|$ times in total, because it tries to go to every edge, so it's $O(|E|)$.

The second for loop ignoring the original BFS part are just $O(1)$ operations run for $|V|$ times.

Therefore total time complexity of this BFS mod is $O(|V| + |E|)$

TITLE/NO.

M T W T F S S

Refs: Disjoint set slide P.18

5. First for loop runs $|V|$ times, so it's in total

$O(|V|) + |V|$ make-set() time

Second for loop runs $|E|$ times. In worst case, union() runs $|V|-1$ times and find-set() always run $2 \cdot |E|$ times.

So total time complexity is $O(1) + O(|V|) + O(|E|)$ $+ |V|$ make-set() $+ (|V|-1)$ union() $+ 2 \cdot |E|$ find-set().

Using union-by-rank + path compression makes those disjoint set operations take $O((|V|+|E|)\,\alpha(|V|+|E|))$

Therefore total time complexity $= O(|V|+|E|\,\alpha(|V|+|E|))$

$\alpha(x)$ is inverse function of ackermann's function.

6

Refs: None

TITLE/NO.

M T W T F S S

Refs: Graph Slide P.40

7. If (u,v) is a cross edge

$\Rightarrow$ u, v are in different tree

$\Rightarrow$ When we are on u, looking at (u,v), v is either gray of black

v is black $\Rightarrow$ (u,v) is undirected edge so v should have visited u with it

$\Rightarrow$ conflict

v is gray $\Rightarrow$ u, v is in the same tree

$\Rightarrow$ conflict

$\Rightarrow$ (u,v) must not be a cross edge

8.

Refs: Graph Slide P.40

```
DFS(G)
    for each vertex u in G.V
        u.color=WHITE
        u.pi=NIL
    time=0
    for each vertex u in G.V
        if u.color==WHITE
            cycle = DFS-VISIT(G,u)
            if cycle
                return TRUE
    return FALSE
```

```
DFS-VISIT(G,u)
    time=time+1
    u.d=time
    u.color=GRAY
    for each v in G.Adj[u]
        if v.color==WHITE
            v.pi=u
            DFS-VISIT(G,v)
        if v.volor==GRAY
            return TRUE
    u.color=BLACK
    time=time+1
    u.f=time
    return FALSE
```

TITLE/NO.

M T W T F S S

Refs = None

9. (a) T = 314159265358979**3**

P = 41592

41592 % 17 = 10

65358 % 17 = 10

35897 % 17 = 10 → 2 spurious hits

(b)

P	A	B	A	B	A	A	B	C	A	B	A	B
π	0	0	1	2	3	1	2	0	1	2	3	4

M T W T F S S

Refs: W10 KMP Part 2 Comments

10.

$T = ABABAB$

$P = ABAB$

$\pi = 0\ 0\ 1\ 2$

It's obvious that valid shifts are 0, 2

@ $i=4$, $q = 4 = m$,

if at line 12 we set $q=0$, then @ $i=6$

$q = 2 \neq m$

$\Rightarrow$ we missed a valid shift

Refs: None

11.

(a)

If $h(x)$ does a uniform hashing, then each possible value of $h(x)$ has equal chance being chosen regardless of x.

(b)

$h_1(x) = x \bmod 10$

linear probing: $L(x,i) = h_1(x) + i \bmod 10$

quadratic probing: $Q(x,i) = h_1(x) + i + 3i^2 \bmod 10$

keys: 13, 12, 2, 3, 4

using $L(x,i)$ =>

0	1	2	3	4	5	6	7	8	9
		12	13	2	3	4			

using $Q(x,i)$ =>

		12	13	4		2	3		

TITLE/NO.

M T W T F S S

Refs: Hashing Slide P.16, 19

12 (a)

When load factor $\frac{n}{m}$ is close to 1, using chaining is better.

Time of using chaining = $O(1+\frac{n}{m})$

" open addressing = close to $O(n)$

(b)

Dynamic hashing uses less space when only a few (key, value) is inserted.

Standard hashing uses a fixed amount of space, can't expand when needed, and wastes space when load factor is low.

TITLE/NO.

M T W T F S S

Refs: youtube/otK2NuoMODk

13、

(1) $r=2$ $g=3$

000	0xA0	
001		
010	0xBA	
011		
100	0xCC	
101		
110	0x1E	0x2E

(2) $r=2$ $g=3$

000	0xA0	
001		
010	0xBA	0xA2
011		
100	0xCC	
101		
110	0x1E	0x2E

(3) $r=2$ $g=3$

000	0xA0	0x08
001		
010	0xBA	0xA2
011		
100	0xCC	
101		
110	0x1E	0x2E

(4) $r=2$ $g=5$

0000	0xA0	0xB0
001		
010	0xBA	0xA2
011		
100	0xCC	
101		
110		
111	0x1E	0x2E
1000	0x08	

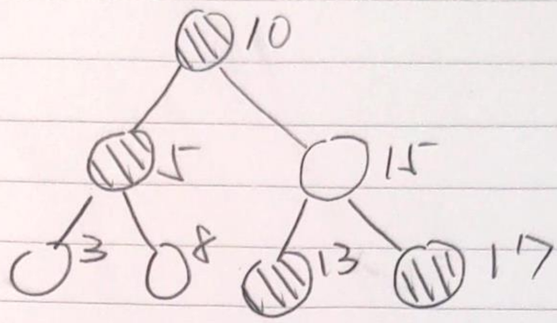
14

Refs: Linear Sorting Slide P.11

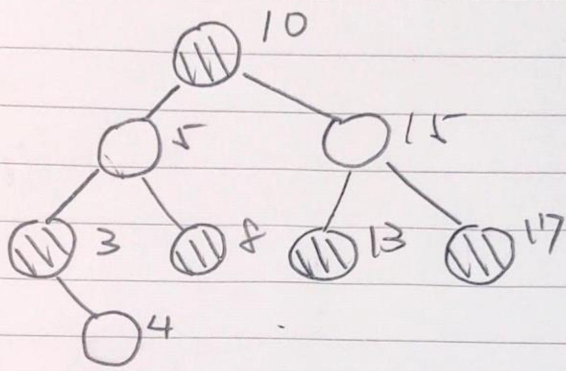
```
ALT-COUNTING-SORT(A, B, k)
let C[0..k] be a new array
for i = 0 to k
    C[i] = 0
for j=1 to A.length
    C[A[j]] = C[A[j]] + 1

for i = k-1 downto 0
    C[i] = C[i]+C[i+1]
for j = 1 to A.length
    B[ A.length-C[A[j]]+1 ] = A[j]
    C[A[j]] -= 1
```

Refs: RBTree slides P. 25-27

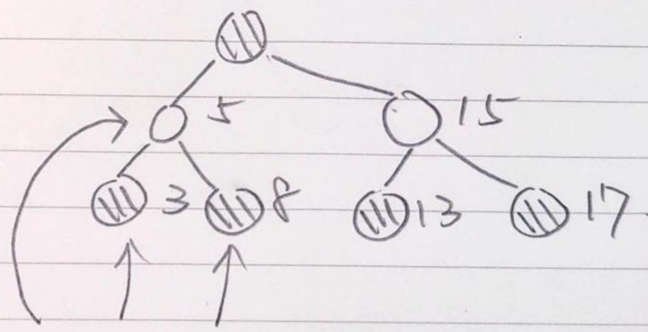
15. Before



Insert 4.



Delete 4.



These 3 has different colors.

So it can be different.

TITLE/NO.

M T W T F S S

Refs：None

16、 Good：課堂活動有參與感

Change：後半學期能像前半有每週即時回饋的練習