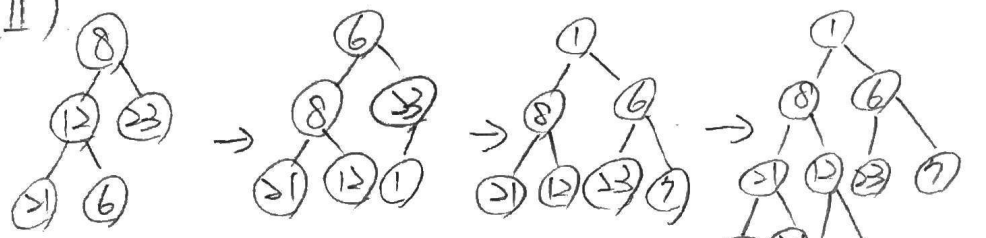
107台大資演

(I)

1. E
2. D
3. B
4. C 一般操作time. => heapify.
5. A
6. A perfect hashing
7. C

(II)



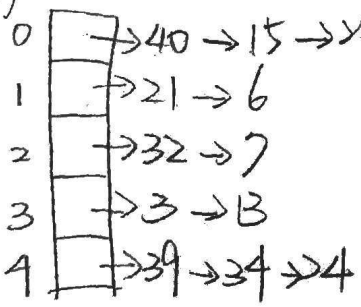
8. C
9. B

height = 4 | root at 1-level | height).

10. A

(III)

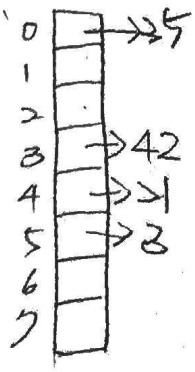
11.



(4×1+2×3+3×1)/5 = 3.6

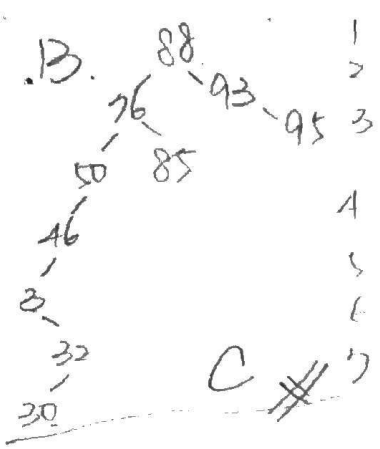
B

12.

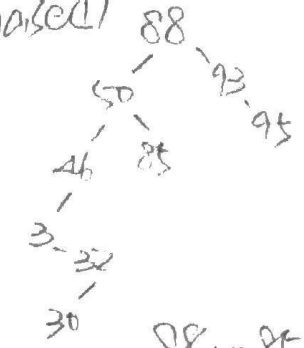


D

13.



remove 76

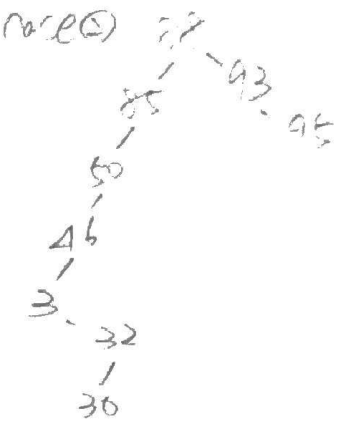


C#

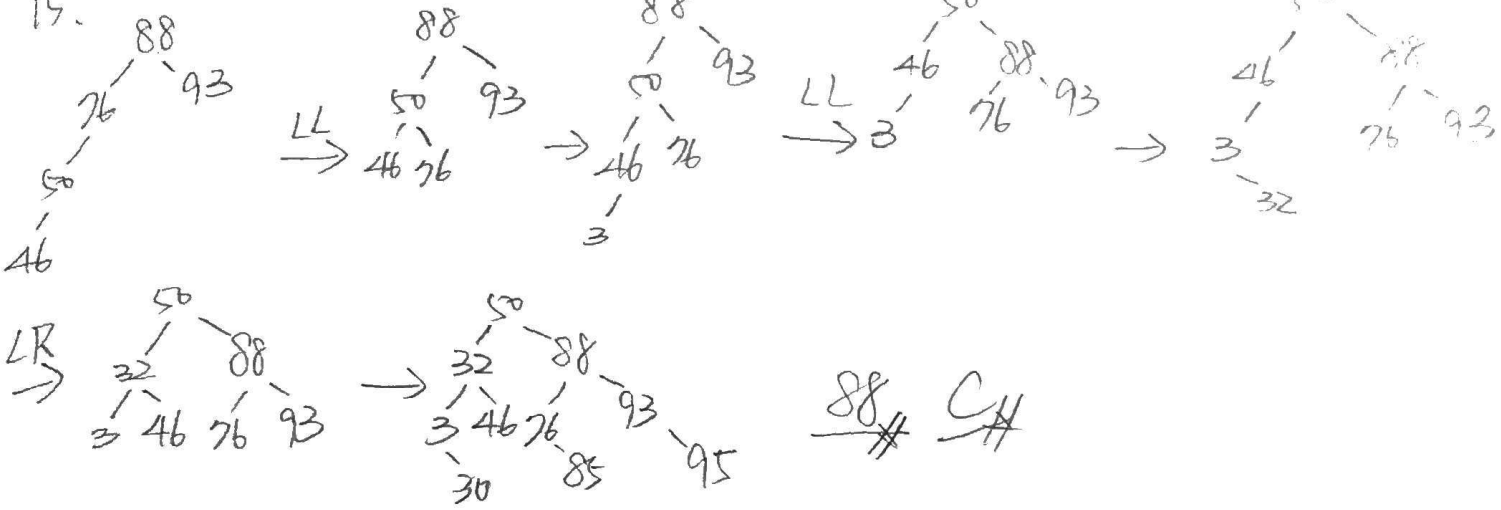
14.

88 or 85

B or C#



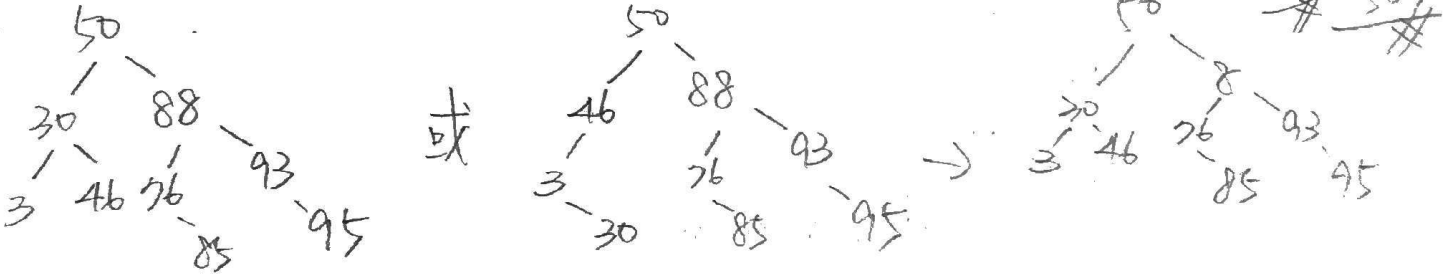
15.



88#

C#

16.



C# 50#

(V)

(a) (1) 假設有一序列為 $\langle a_1, a_2, \cdots a_{n-1}, a_n \rangle$

① 則使用 Greedy 方法選結果為 $|a_1 - a_n| + |a_2 - a_{n-1}| + \cdots |a_{n/2}| = M_1$

② 有可能存在 $|a_1 - a_j|$ $j \neq n$ 使值 $> |a_1 - a_n|$ 則 $\exists sum > M_1$

③ 所以 greedy 不適用

(V)

(a)

(2) ① 宣告2個變數，int round $\begin{cases} 奇數:找最大 \\ 偶數:找最小 \end{cases}$，temp 用於存放該輪選到的數字.
(initial = 1)

② 從A端取2數 比較依照 round 選擇較大/較小的數 assign給 temp
不要的數從B端 push 回去

③ 從A端再拿一個數比較，符合條件者更新為 temp，另一數則 push B端
重覆 $n-1$ 次

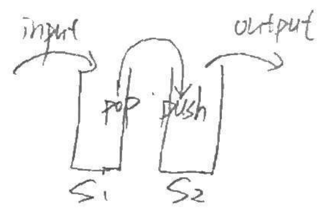
④ 回到 step 2，(round+1) % 2，做 n 輪

$T(n) = T(n) + O(n) = O(n^2)$ #

(V)

(b)

(1) case 1：用2個 stack 實現 queue，令 stack 為 S_1, S_2



① push：input 直接往 S_1 push

② pop：當 S_2 尚有值時 pop，若 S_2 沒有值則 pop S_1 all element 到 S_2
再 pop S_2

case 2：用2個 stack 實現 deque (2端為 A, B)，令 stack 為 S_1, S_2

① push A：將 input 直接 push to S_1
push B：將 input 直接 push to S_2

② pop A：從 S_1 pop 一個 element，若 S_1 為空則 pop all element from S_2
push to S_1，再 pop top element，最後 pop all element from S_1 push
回 S_2 ⇒ $O(2n)$

pop B：同上方法 (A·B 角色互換)

pop both：將 S_1, S_2 同時 pop 一個 element.

(V)

1b)

12) 定義成本: a. push = 1

b. pop = { 1 / n

總成本 = 推n個元素, pop出n個元素(每個可能要n) $\Rightarrow n^2$

平攤成本 = $n^2/n = O(n)$

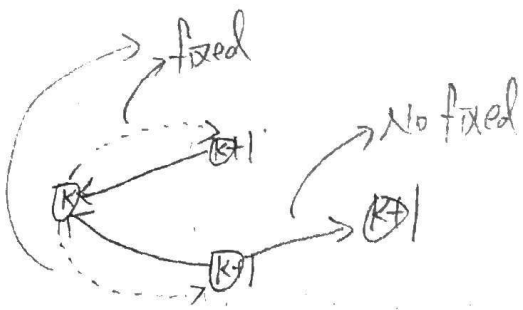
13) 定義成本 = a. push = 1

pop = { 1 / ≥n

總成本 = 推n個元素, pop出n個元素(每個可能要≥n time) $\Rightarrow \geq n^2$

平攤成本 = $\geq n^2/n = O(n)$

主要成本來源皆為pop(搬移成本)



(VI)

1a) XX向連通圖

① 從V_a桌當起桌做BFS, 終桌為V_b

② 中間桌考慮$V_i \to V_j$ { 若V_i到V_i沒有path: V_j的fix time更新成V_i+1

若V_i到V_i path沒斷: V_j的fix time = V_i的fix time

③ 對每條邊依BFS順序做relax

④ 若對任意桌V_k存在一條"新fixed path time < 舊fixed path time"

就更新

⑤ 類似求topological sort, time complexity = $O(|V|+E)$

(VI)

b)

i) 有向圖 $V_a \to V_b$

① 用BFS計算，以V_a為起點，直到可達V_b

② 所有點和邊至多走過一次 $\Rightarrow O(V+E)$ #

c) 每邊有權重，有向圖

① 用V_a當起始點，做Dijkstra algorithm，終點V_b

② 每次relax目前可達之weight最小邊，relax最多E次

③ 利用Fibonacci heap輔助 $\Rightarrow O(V\log V+E)$