1. 第9張教的min-max heap有教如何插入和刪除,那原本的min-max heap要怎麼建呢?就是給你一堆數字如何去建min-max heap?

(Ans):

一個數字一個數字逐次insert即可。

2. Radix Sort，以撲克牌爲例，不管是先排花色或先排點數，都會產生子牌組。P. 352說：LSD比MSD簡單，因爲LSD不需對各個子牌組分別做排序；它的 "overhead"比較少。請問是什麼意思？

(Ans):

MSD需要對各個子排組作比較性排序而LSD不需要，請參考 http://knight.fcu.edu.tw/~d9046876/ds/d_64.htm

3. (課本p.316 figure 6.44)late-early=0 時爲critical path的路徑之一，不過我還是搞不清楚其中late 和 early分別代表甚麼？

(Ans):

ES - earliest start time: the earliest time at which the activity can start given that its precedent activities must be completed first.

EF - earliest finish time, equal to the earliest start time for the activity plus the time required to complete the activity.

LF - latest finish time: the latest time at which the activity can be completed without delaying the project.

LS - latest start time, equal to the latest finish time minus the time required to complete the activity.

4. (page.353)老師上課有說到對電腦來說LSD是比較簡單的作法，我看了一下課文，裡面是說However, since an LSD sort does not require the maintenance of independent subpiles, it is easier to implement.不過我不太懂課文這邊的解釋，不知道有沒有更清楚實際一點的解釋。

(Ans):

同第二題，以撲克牌來說，LSD不需要對子牌組另外做比較性的排序(如insertion sort)，可以以同樣的演算法(bucket sort)做到底所以較簡單。

5. 在hash的主要作用是把一個字串經過一個function對應到一個數字而以此作爲一個index,但爲甚麼一個一對一的function很難找,是因爲那一個function要1,2,3....這樣連續的限制嗎?

(Ans):

不是的，只要對於所有的輸入來說有一組唯一的hash值即可做到一對一的

hash function，就很容易讓不同的數字hash到相同的位置導致非一對一的結果。

6. Kruskal's algorithm 的time complexity爲 O(|E|log|E|),如何知道他的時間複雜度， 還有建立一個花費最少的擴張樹(MST),Prim's algorithm和Kruskal's algorithm 哪個是花費最少的擴張樹， 擴張樹定義任兩個節點間爲相連,也就是說存在一條路徑,爲什麼這條路徑不一定是兩點之間的最短路徑?

(Ans):

問題一：請參看Program 6.7: while loop最多跑 |E| 次，每一次的loop，第一行要跑log|E|次，第二行也是log|E|次，所以整個是O(|E|log|E|)。

問題二：Kruskal's algorithm = O(e^2); Prim's algorithm = O(e log e)。

問題三：因爲最短路徑有可能已經被移除了。

7. tree的高度=ln(n+1) FLOOR, 那爲什麼有時候又會說是log(n)，不太了解。

(Ans):

Tree的高度是整數值，而ln出來的值並不一定是整數值，所以取floor；而在算complexity時，floor(ln(n+1))和log(n)的order是一樣的，所以直接用log(n)即可。

8. 我想問HASHING投影片的LINEAR PROBLING 的DELETE部分在DELETE29時 他在FILL時跳過0 而直接跳到45 是表示說原本MOD 17的資訊有被存下來 然後再SEARCH嗎?所以不管隔多遠 只要前面有比較靠近本來應該存放的位子的話 就照順序往前移 所以要對空位後面的每個數字都做SEARCH?

(Ans):

可以一個一個往前掃過去，每看一個就檢查它是否應該出現在本來的位置，再往前移；如果這種動作很常出現，也可以儲存這個資訊，用空間來換取時間。

9. 那LINEAR PROBLING 如果SLOT都滿了要怎麼處理 還是原本就會避開不讓SLOT個數小於要儲存的資訊的數目呢?

(Ans):

可以用linear probing或是chaining的方法來做處理，或是用後面dynamic hashing的概念來動態配置大小（請參看課本）。

10. 爲什麼在delete heap的root時 要從最後一個節點遞補上去?

(Ans):

這樣才能確保所用的tree是一個complete binary tree，而當一個binary tree是

complete binary tree時，可以用array來做implementation，在呼叫任一個節點時，可以快速的知道它的parent node和child nodes（用lemma5.3）。

11. 請問第七章講議中，complexity of sort那個表格裡stability的意思是什麼？

(Ans):

Stability的意思是指在sorting的過程中，兩個以上相同大小的elements是否經過該sorting method 而使elements的sorting後的次序與原本的次序不一樣。(次序一樣稱stable, 不一樣稱unstable)

Example:

Sequence : $\{3_1, 2, 4, 1, 11, 3_2\}$

Sorting method 1 => $\{1, 2, 3_1, 3_2, 4, 11\}$: stable

Sorting method 2 => $\{1, 2, 3_2, 3_1, 4, 11\}$: unstable

12. AOE 求出 critical path 是由最早發生時間和最晚發生時間得出。不過我在課本中對於如何因此求出 critical path 不是很懂；去看其他的書，是寫說「若是該頂點的最早和最晚時間相同，此頂點即爲 critical path 的頂點」，所以第一個想請問，課本上到底是如何求出 critical path？如果求法與我所看到的書是一樣的，那爲什麼最早時間等於最晚時間的點，就會是 critical path通過的點？

(Ans):

想用AOE的概念求出critical path首先要知道earliest 和latest time的定義。

earliest time of v_i :

是指從v_i會發生的最早時間點(從起點v_0算起)

latest time of v_i :

是指在不增加整個project完成時間的情況下，允許v_i可最晚發生的時間點

在計算earliest time和latest time方面主要分爲兩個部分：forward stage和backward stage.

forward stage

利用公式6.3: $earliest[j] = \max\{earliest[i] + DurationOf < i, j >\}$ 計算earliest time. 因爲v_j得由in-degree的所有方向均完成後才可能發生，故此公式的意義在於『從在v_j之前和v_j有直接相連的點集合中找到v_i，使得{earliest[i] + <i,j> 間發生的時間長度}爲最大值』

backward stage

利用公式6.4: $latest[j] = \min\{latest[i] - DurationOf < j, i >\}$ 計算latest time. 因爲v_j得由out-degree的所有方向反推最晚要發生的時間點，故此公式的意義在於『從在v_j之後和v_j有直接相連的點集合中找到v_i，使得{latest[i] -

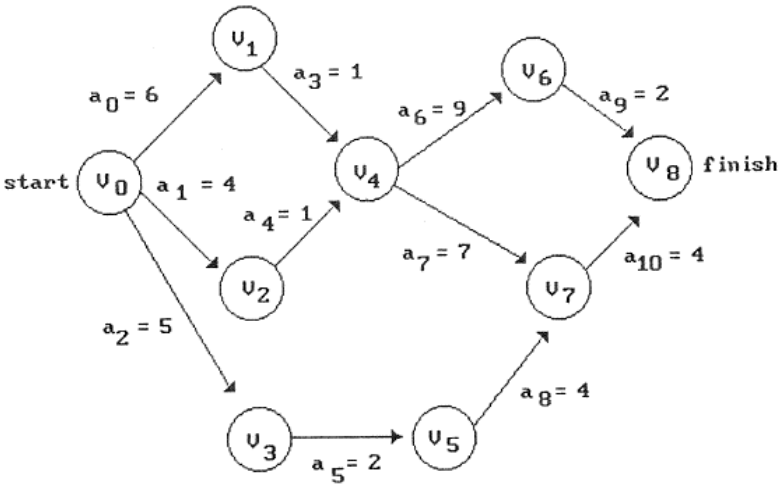
<i,j> 間發生的時間長度}爲最小值』

故從earliest time和latest time的定義來看，若一個activity會發生的最早時間點和會發生的最遲時間點相同的話，則說明該activity一定得在那個時間點上完成才不會影響在完成整個project的過程，其在過程中佔著舉足輕重的地位，我們稱作爲critical activity. 在將critical activities連接成可完成project的paths, 稱critical paths. (隨著critical activities組合的不同，可有多條的critical paths)

以下爲課本上找critical paths的一個實例：

Critical paths: v_0->v_1->v_4->v_7->v_8

v_0->v_1->v_4->v_6->v_8



Earliest	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	Stack
initial	0	0	0	0	0	0	0	0	0	[0]
output v_0	0	6	4	5	0	0	0	0	0	[3, 2, 1]
output v_3	0	6	4	5	0	7	0	0	0	[5, 2, 1]
output v_5	0	6	4	5	0	7	0	11	0	[2, 1]
output v_2	0	6	4	5	5	7	0	11	0	[1]
output v_1	0	6	4	5	7	7	0	11	0	[4]
output v_4	0	6	4	5	7	7	16	14	0	[7, 6]
output v_7	0	6	4	5	7	7	16	14	16	[6]
output v_6	0	6	4	5	7	7	16	14	18	[8]
output v_8										

Latest	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	Stack
initial	18	18	18	18	18	18	18	18	18	[8]
output v_8	18	18	18	18	18	18	16	14	18	[7, 6]
output v_7	18	18	18	18	7	10	16	14	18	[5, 6]
output v_5	18	18	18	18	7	10	16	14	18	[3, 6]
output v_3	3	18	18	8	7	10	16	14	18	[6]
output v_6	3	18	18	8	7	10	16	14	18	[4]
output v_4	3	6	6	8	7	10	16	14	18	[2, 1]
output v_2	2	6	6	8	7	10	16	14	18	[1]
output v_1	0	6	6	8	7	10	16	14	18	[0]

13. 在讀hash function的時候，發現linear probing 怎麼樣也不會比chaining 來的好用，不知道是不是有什麼特別的情況，讓我們可以用linear probing 比較好的??

(Ans):

Linear probing是一種collisions發生時的處理方法，其作法在將新加進來的element放到下一個open slot. Chaining 則是一種liked list的data structure,允許elements 可直接串接到hashing function對應到的bucket. 若比較static 和dynamic slots的data structure, 則static會有搜尋整個儲存elements的space O(buckets數*slots數)的worst case, 而dynamic的worst case則是O(slots數).

14. overflow是什麼意思？我現在將它理解成當存在一個bucket滿的時候，若我們繼續塞東西給此bucket，就會爆到，這就是overflow。這樣解釋對嗎？

(Ans):

嗯，大致上可這麼說。overflow的定義是指there is no space in the bucket for the new pair (key, element). (page 18, chap8 course slides). 即該bucket的slots中都有elements時便無法再儲存新進來的element的情況稱overflow.

15. 爲什麼heap sort是不穩定的？

(Ans):

以max heap作sorting爲例，它是將sequence先依序填到一個complete binary tree上，然後將maximum調整到root，然後再與最後的那個node交換；如此便可視爲將sequence中的一個element排序好。但若有兩個以上具相同值的elements, 在做max heap adjusting時有可能造成原本次序在前面的element先被adjust到root, 原本次序在後面的element在它之後才被adjust。等到整個sequence都排序好之後，就會產生unstable的現象。(heap sort的作法請參考heap sort example, p48-53 , chap 7 course slides)

16. 對於quick sort 的worst case要用哪一種sort來代替比較好？

(Ans):

如果你的資料是 random 未排序的，且 quick sort 進行中，pivot 是 random 選擇的，則你並無法預測此 quick sort 是否會遇到 worst case。如果資料與 pivot 的選擇並不是如上所述，則應該視實際的應用情形選用適合的 sort 方法，大部分的演算法都有其各自的優點與缺點，無法說誰是最好的。

17. 在建mininum cost spanningtree時，共有三種演算法，是要針對不一樣的tree選取不一樣的作法還是根本就沒差呢？

(Ans):

每一種演算法都能建構出 minimum cost spanning tree，你可以擇一使用，然而在某些實際應用上 (例如考慮平行處理)，你可能需要使用特定的演算法 (有可能是除了課本上講的之外的作法)，如果你有興趣請參閱以下網址：
http://en.wikipedia.org/wiki/Minimum_spanning_tree

18. 爲什麼quick-sort是個不穩定的排序法？（課本的問題，但是我想不出）

(Ans):

假設有一序列 $a_1, a_2, \ldots, a_n$，假設 $a_i == a_j$，其中 $i < j$，則使用 quick-sort 從小到大排序後，並無法保證 a_i 會出現在 a_i 之前，因此 quick-sort 不是一個 stable 排序法。

19. 爲什麼hash搜尋資料比較快？既然它是利用hash function產生的數值來儲存，假設資料存在array[123]裡，我知道他的key值是123，要找這筆資料，不還是要用到binary search之類的搜尋方式嗎？還是說我可以直接知道那個資料的位置再下去找？不知道是不是我的觀念錯了。我還想問hash到底有什麼好處，不就是隨機array嗎？普通的array辦不到嗎？

(Ans):

既然知道 key 值是 123，就可以直接索引到陣列的第 123 個元素，並不需要 binary search。Hash 的好處在於它只要算出 key 後，幾乎是在 O(1) 的時間內就可以找到資料，而一般未排序的陣列可能需要線性搜尋，其時間複雜度是 O(n)，其中 n 是資料個數。

20. random probing可以有效減少cluster和overflow的發生率嗎?其原理爲何?

(Ans):

一發生溢位，使用雜湊函數（f（x）＋s（i））％ b以計算bucket位址，1≦i≦b－1，s（i）爲pseudorandom number產生器，可產生1～b－1，每一數字恰爲一次，理論上能將發生 overflow 的資料平均分配到 bucket 中，因此能減少 cluster 與 overflow的狀況，但是這兩種情況還是會發生。

21. 執行quick sort的時候，爲什麼要假設 list[left].key <= list[right+1].key?

(program 7.6)

(Ans):

Quick sort 是再選定 pivot後從數列左邊往右找到第一個比pivot小的，從右往左找比pivot大的，爲了避免在特殊情況下，找不到比 pivot 更大的數時程式進入無窮回圈，所以必須做此假設來卡住向右搜尋的動作。

22. insertion sort的時間複雜度爲O((k+1)n), k爲 LOO (left out of order)的數目，爲什麼要 k+1 而不是 k 呢?

(Ans):

“k” 是額外要做的工作 “1” 是原本就要執行的工作，以最簡單的例子看，若有 k=1 個 LOO 那麼所需的時間應該是 O((k+1)n) 而不是 O(kn)=O(n)