Final

Graded

Student
鄭博允

Total Points
158 / 230 pts

Question 1

1 — 8 / 15 pts

- – 0 pts Correct
- – 3 pts Minor mistake in algorithm
- – 7 pts Major mistake in algorithm / algorithm incomplete
- – 4 pts Minor mistake that makes $h(s_1) = h(s_2) \rightarrow s_1 \bowtie s_2$ not guaranteed.
- ✔ – 7 pts Major mistake that makes $h(s_1) = h(s_2) \rightarrow s_1 \bowtie s_2$ not guaranteed.
- – 1 pt $h(s_1) = h(s_2) \leftarrow s_1 \bowtie s_2$ not guaranteed.
- – 3 pts Algorithm doesn't satisfy (b)
- – 4 pts Time complexity is worse than $O(n)$.
- – 15 pts Completely wrong or empty.

1 How about aaaa vs bb?

Question 2

2 — 0 / 15 pts

- – 0 pts Correct
- – 2 pts very minor mistake
- – 2 pts didn't analysis the time complexity
- – 0 pts minor mistake in hash function
- – 5 pts minor mistake
- – 0 pts major mistake in hash function
- – 10 pts major mistake
- ✔ – 15 pts wrong answer / empty

Question 3

3 **5** / 10 pts

- **− 0 pts** Correct
- **− 3 pts** Correct example with wrong explanation
- ✔ **− 5 pts** Correct example without explanation
- **− 5 pts** Wrong or no example of s_w
- **− 10 pts** Wrong or blank

Question 4

4 **8** / 10 pts

- ✔ **+ 1 pt** (a) Yes
- ✔ **+ 2 pts** (a) Correct Example or Explanation
- ✔ **+ 3 pts** (b) Put as many red nodes as possible in the right subtree and no red node in the left subtree.
- ✔ **+ 3 pts** (b) Derive correct formula
- **+ 1 pt** (b) Cases for height is odd and even.
- **− 1 pt** (a) Minor mistake in explanation.
- ✔ **− 1 pt** (b) Minor mistake in deriving.
- **− 2 pts** (b) Major mistake in deriving.
- **+ 0 pts** Blank / Wrong

Question 5

5 **10** / 10 pts

- ✔ **− 0 pts** Correct
- **− 0 pts** Wrote (T, y) instead of (T, x)
- **− 0 pts** The parameter is not the parent node
- **− 2 pts** One mistake
- **− 10 pts** Incorrect or blank

Question 6

6 **25** / 25 pts

✔ **+ 5 pts** (1) correct

✔ **+ 5 pts** (2) correct

✔ **+ 5 pts** (3) correct

✔ **+ 5 pts** (4) correct

✔ **+ 5 pts** (5) correct

+ 0 pts viewed

Question 7

7 Resolved **15** / 20 pts

✔ **+ 5 pts** without recursive with correct thoughts

+ 15 pts correct pseudocode

✔ **+ 10 pts** minor mistake

+ 5 pts major mistake with some correct thoughts

+ 0 pts wrong / blank

– 3 pts wrong spelling or tiny error

2 original k != current k so k - m isn't correct

Regrade Request **Submitted on:** Jun 19

助教不好意思
我不是很懂這題的錯誤出在哪
original k != current k 照理來說應該沒有影響吧
演算法應該只要回傳 k-th smallest element 就好
中間的 k 怎麼變化應該沒有差

因為你的 m 是從 l 開始算，所以你每一輪的 k 的變更項是 k - (m - l) 相關的值，而不是 k = k - m

Reviewed on: Jun 19

Question 8

8 **15** / 15 pts

✔ **– 0 pts** Correct

– 7 pts Minor mistake

– 12 pts Major mistake

– 15 pts Blank / No answer

Question 9

9 **0** / 15 pts

+ 15 pts Correct

+ 15 pts Using the array method mentioned in the class + union-by-size method.
Note that you need to store the elements each set has (you can use linked-list, variable-length array) since you cannot traverse the whole array, that would cause the time of UNION to become O(N) instead of O(set size), thus the overall time complexity is O(N^2) and cannot be improved by union-by-size.

+ 15 pts Using the tree method + **path compression** + weighted union, note that you need to use path compression to make sure that FIND-SET is O(1) if you call it more than n times.

+ 5 pts Mentioned weighted union method

− 3 pts Note that you need to put all the things into an array to apply the quickselect since linked lists do not supply random access (or you need to explain why you can avoid random access in that algorithm).

− 5 pts You need to mention the weighted union method in your linked list method

− 5 pts UNION needs to finish in O(small set) time, sorting the items in linked-list make it become O(small set*log(big set)) time, which would make the overall time complexity $O(m + n \lg^2 n)$

✔ **+ 0 pts** Blank

+ 0 pts Wrong answer/only mention the original disjoint set/print-k-smallest function

Question 10

10 **0** / 10 pts

− 0 pts Correct

− 5 pts need more detail about the answer.

− 10 pts Blank

✔ **− 10 pts** answer not reasonable

Question 11

11 **10** / 10 pts

+ 0 pts Empty / Wrong

✔ **+ 1 pt** Correct answer of (a)

✔ **+ 4 pts** Correct explanation of (a)

+ 2 pts Explanation of (a) with minor mistake

✔ **+ 5 pts** Correct explanation of (b)

+ 3 pts Explanation of (b) with minor mistake

Question 12

12

Resolved 9 / 15 pts

– 0 pts Correct

– 2 pts only show final result with (a)

– 1 pt only show final result with (b)

– 2 pts one mistake in (a)

– 1 pt one mistake in (b)

– 2 pts 1-based hash table

– 15 pts wrong/blank

– 6 pts Point adjustment

Regrade Request **Submitted on: Jun 19**

助教不好意思
我想確認一下 point adjustment 是什麼意思
這題被扣分是因為什麼原因 答案錯誤還是格式錯誤還是其他

這裡是 (a) 的 28, 88, 59 三個數字填錯了，系統之所以會顯示那個原因是因為個人操作失誤，造成不便還請見諒

Reviewed on: Jun 19

Question 13

13

25 / 25 pts

– 0 pts Correct

– 3 pts minor mistake

– 3 pts minor mistake 2

– 10 pts multiple keys in one place

– 25 pts wrong/blank

+ 3 pts reasonable effort

Question 14

14 **20** / 25 pts

- **– 0 pts** Correct
- **– 5 pts** Operation a incorrect
- **– 5 pts** Operation b incorrect
- **– 5 pts** Operation c incorrect
- **– 5 pts** Operation d incorrect
- ✔ **– 5 pts** Operation e incorrect
- **– 25 pts** Did not answer
- **– 12.5 pts** Did not provide each step though having correct answer

Question 15

15 feedback **8** / 10 pts

- **– 0 pts** 很完整的建設性建議
- ✔ **– 2 pts** 建議內容比較少一些
- **– 4 pts** 建議只有非常少量文字
- **– 10 pts** 空白

Student ID	Name
B11902038	鄭博允

Final Examination Problem Sheet

TIME: 06/06/2023, 13:20–16:20

Exam Policies

- This is an open-book exam. You can use any hard copy you bring. **No electronic device is allowed** to be used to assist in taking the exam (e.g., tablet, laptop, smartphone), nor can you use the network to consult any online resources. Any form of cheating, lying or plagiarism will not be tolerated. Students can get zero scores and/or get negative scores and/or fail the class and/or be kicked out of school and/or receive other punishments for those kinds of misconduct.
- Both English and Chinese (if suited) are allowed for answering the questions. We do not accept any other languages.
- Unless explicitly asked, you do **NOT** need to justify why your algorithm runs in the desired time/space complexity, but you are welcomed to briefly explain the key ideas to help the TAs grade your solution.
- There are 15 problems. The ones marked with * are supposedly simple(r); the ones marked with ** are supposedly regular; the ones marked with *** are supposedly difficult. **The full credit is 225 points + 5 bonus points.**
- Please **write your name and student ID** on the top of the first page of the exam paper. When you submit the exam papers, make sure that they are **sorted by the page number** in ascending order and all pages are present. Please write directly on the answer area following the description of each problem on the exam papers. You may use additional blank papers for calculation. However, write your final answers only on the exam papers, **and no extra page can be submitted. Failure to comply with these instructions will induce up to 40 points of penalty in total.**
- You should keep your answers as *concise* as possible to facilitate grading. In particular, please use proper *pseudo code* and/or sufficiently-understandable words to describe your algorithm to the TAs clearly!

String Matching

For problem 1 and problem 2

Given a string S of length n. The set of characters to form S is Σ, where the size of the set is $k = |\Sigma|$. Let $\Sigma = \{c_1, c_2, \ldots, c_k\}$. Define $f(s, c)$ to represent the frequency of character c in string s, i.e., the number of times c appears in s. Note that k is not a constant and can change.
We can now define a "frequency match" relation $\bowtie$ of two strings s_1 and s_2:

$$s_1 \bowtie s_2 \textit{ if and only if } f(s_1, c_i) = f(s_2, c_i),\ \forall i \in \mathbb{N} \text{ and } i \in \{1 \le i \le k\}. \tag{1}$$

1. (**) (15 points) Modify the Rabin-Karp hash function, and derive a new one $h(s)$ for a given string s, so that it can be used for the new "frequency match" operation. The function should fulfill the following requirements:
   (a) For two given strings s_1 and s_2, $h(s_1) = h(s_2)$ if and only $s_1 \bowtie s_2$.
   (b) $h(s[l_1..l_2]) + h(s[l_2 + 1..r]) = h(s[l_1..r])$, $l_1 \le l_2 \le r$.
   (c) Calculate $h(s)$ for a string s of length n takes $O(n)$-time. Note that the running time should not depend on k.

$S_1 \bowtie S_2$ 可以看成是 S_1 和 S_2 的組成元素一樣，但順序不一定相同

假設 Σ 內的每個元素都對應到一個值，$C_i = i$

i.e., $C_1 = 1$, $C_2 = 2$, $C_3 = 3$ ... $C_k = k$

我們遍歷整個 S，假設 S 的長度是 n

每遇到一個字元 C_i 就把 hashing value 加上 i^2

舉例來說 "ada" $= 1^2 + 4^2 + 1^2 = 18$

"bbb" $= 2^2 + 2^2 + 2^2 = 12$

因為我們只有遍歷整個 S，所以 time complexity $= O(n)$

2. (***) (15 points) Given a string S of length n, a set of q queries will be made. In each query, a set of four index numbers l_1, r_1, l_2, r_2 are given. The query asks if $s[l_1..r_1] \bowtie s[l_2..r_2]$. Develop an algorithm in pseudo code, such that it takes $O(n+q)$-time to perform initialization and to respond to all queries. Note that the running time should not depend on k. Show that your algorithm indeed complies with the time complexity requirement. For the simplicity of the analysis, you can assume that spurious hit (when applicable) never occurs.

3. (*) (10 points) A student develops MODIFIED-COMPUTE-PREFIX-FUNCTION and claims that it would run faster than the original COMPUTE-PREFIX-FUNCTION for the KMP string matching algorithm. The only modifications are in lines 6-7. It might be faster, but the grading TA discovers that in some cases MODIFIED-COMPUTE-PREFIX-FUNCTION no longer gives the correct output in $\pi[]$. Let Σ denote the set of characters used for the string, which include the numerical digits $0-9$. Please give one string s_c that would still generate correct output, and one string s_w that would generate incorrect output. Clearly explain why s_c would still be correct but s_w would not be.

   The correctness of the output is determined by the original definition of the prefix function:

$$\pi[q] = \max\{k : k < q \text{ and } P_k \sqsupset P_q\}^1 \tag{2}$$

```
MODIFIED-COMPUTE-PREFIX-FUNCTION(P)
 1  m = P.length
 2  let π[1..m] be a new array
 3  π[1] = 0
 4  k = 0
 5  for q = 2 to m
 6       if P[k + 1] ≠ P[q]
 7            k = 0
 8       if P[k + 1] == P[q]
 9            k = k + 1
10       π[q] = k
11  return π
```

[1] In case some students did not pay attention to the lecture, $P_k \sqsupset P_q$ means P_k is a suffix of P_q.

S_c = 1 2 3 1 2 3

the output will be: 0 0 0 1 2 3

which is correct

S_w = 1 2 3 1 2 4 1 2 3 1 2 3

the output will be: 0 0 0 1 2 0

Red-Black Tree

4. (*) (10 points) Given a red-black tree of height h. The rank r of a certain key k means that k is the r-th smallest key out of all keys in the tree. Then, we can define rank ratio $\rho = \frac{r}{n}$, where n is the number of keys in the tree.

   (a) Can the key of the root node be the median of all keys in the tree? Please explain.

   (b) What is the smallst possible rank ratio of the key of the root node? Please give the rank ratio in terms of h and explain your answer.

(a)

Yes,

since RB tree is a binary search tree, which means all nodes in the left subtrees of root is smaller than root and all nodes in the right subtrees is larger than root.

To make the key of the root node be the median, that is, there's half of nodes smaller than root and another half larger than root, therefore we can make the size of left subtrees of root equals to the size of right to satisfy the condition, which is absolutely possible.

(b)

For a RB tree of height h, it's black height must $\geq \frac{h}{2}$, that is, the smallest size of root's leftsubtree is when all nodes inside is black and has $2^{\frac{h}{2}}-1$ nodes.

Therefore, there must be at least $2^{\frac{h}{2}}-1$ nodes smaller than root and the rank of root $r \geq 2^{\frac{h}{2}}$.

For a RB tree of height h, the right subtree is at most $2^{(h-1)}-1$ nodes.

In the end, we have the ratio of root is at least

$$\rho \geq \frac{2^{\frac{h}{2}}}{2^{\frac{h}{2}} + 2^{(h-1)} - 1}$$

5. (*) (10 points) Write the pseudo code of RIGHT-ROTATE(T, x) to perform right rotation on a red-black tree in the blank area below. The operation is shown in Figure 1. Below you can find the pseudo code of LEFT-ROTATE(T, x).

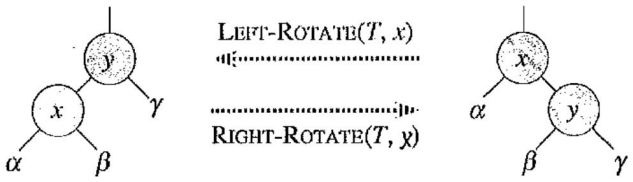


Figure 1: Rotation in red-black trees

```
LEFT-ROTATE(T,x)
1  y = x.right
2  x.right = y.left
3  if y.left ≠ T.nil
4       y.left.p = x
5  y.p = x.p
6  if x.p == T.nil
7       T.root = y
8  elseif x == x.p.left
9       x.p.left = y
10 else x.p.right = y
11 y.left = x
12 x.p = y
```

```
Right - Rotate (T, x)
y = x.left
x.left = y.right
if y.right ≠ T.nil
    y.right.p = x
y.p = x.p
if x.p == T.nil
    T.root = y
else if x == x.p.left
    x.p.left = y
else
    x.p.right = y
y.right = x
x.p = y
```

6. (**) (25 points) Figure 2 shows a red-black tree where uppercase letters are used as keys and lexicograhic order is used for comparison between keys. Perform the following insertions in the given order. Show the outcome of the tree after completing each insertion. Note that you do not have to show the intermediate steps during the insertions. Only the final result after each insertion is needed and graded. (1) insert T; (2) insert Y; (3) insert U; (4) insert Z; (5) insert A.

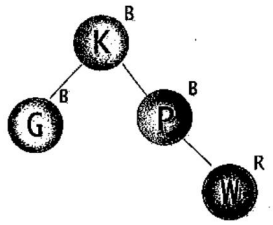
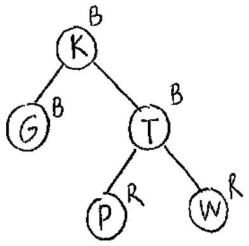
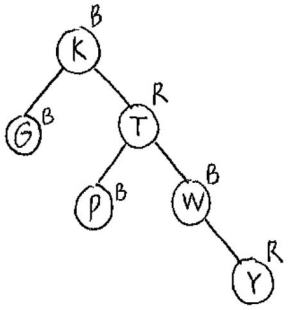


Figure 2: A red-black tree. The upperright corner of each node shows the color of the node in red (R) or in black (B). The key of each node is shown in the circle.

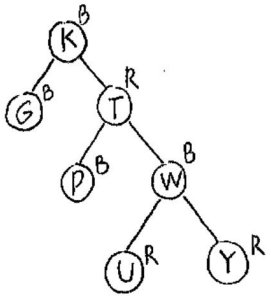
(1) Insert T



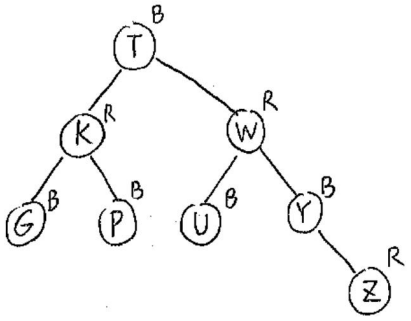
(2) Insert Y



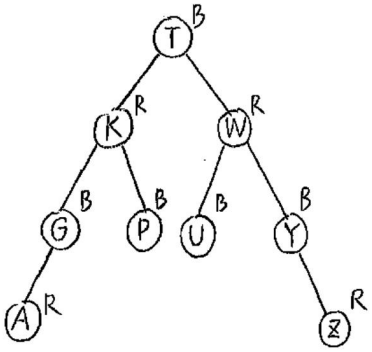
(3) Insert U



(4) Insert Z



(5) Insert A



Disjoint Sets and Sorting

7. (**) (15 points + 5 bonus points) `quickselect` is an algorithm very similar to `quicksort`. But instead of returning the sorted array, `quickselect` returns the k-th smallest element in the input array, where k is a given parameter. A pivot is still selected, and partitioning the input array into those smaller than the pivot and the others is also performed. Then, it would attempt to find the k-th smallest element from one of the two partitions by further partitioning it with a new pivot.

   Modify the `quicksort` C function below to perform `quickselect` with the following requirements:

   (a) Use the function prototype `void quickselect(int l, int u, int k);` such a function call will look for the k-th smallest element from the input array `x[]` between indices `l` and `u`.

   (b) On average, it should be able to find the k-th smallest element in $O(n)$-time, where n is the size of the input array.

   (c) (bonus 5 points) Modify the function to be iterative instead of recursive. If your implementation is recursive, you can still get full points of this problem but without the bonus.

```c
//Lomuto quicksort, as given in the Beautiful Code book
void quicksort(int l, int u) {
    int i, m;
    if (l>=u) return;
    // swapping x[l] and x[randint(l,u)]
    swap(l, randint(l, u));
    m = l;
    for (i=l+1; i<=u; i++)
        if (x[i] < x[l])
            swap(++m, i);
    swap(l, m)
    quicksort(l, m-1);
    quicksort(m+1, u);
}
```

```c
int quickselect (int l, int u, int k){
    int i, m;
    while (l < u){
        swap (l, radint(l,u));
        m = l;
        for (i = l+1; i ≤ u; i++)
            if (x[i] < x[l])
                swap(++m, i);
        swap (l, m);
        if (m-l+1 == k) return x[m];
        elseif (m-l+1 < k)
            k = k-
            l = m+1;
        else
            u = m-1;
    }
    return -1;
}
```

8. (**) (15 points) Analyze your `quickselect` implementation from the previous problem and show that *in the best case* it runs in $O(n)$-time.

best case :

選中的 pivot 就是我們要找的 k-th smallest element,

這樣我們只要跑一次 for 迴圈 (從 l+1 到 u) 即可

因此會是 O(n)

9. (**) (15 points) In this problem, we ask you to develop an implementation of disjoint set data structure supporting the PRINT-K-SMALLEST(k, i) operation (explained below). You can use one of the three implementations introduced in the class as the starting point and revise it. No pseudo code is needed, but brief explanation is required.

   Let n be the number of elements in the sets, i.e., the number of executed MAKE-SET operations. Now, in addition to MAKE-SET, FIND-SET, and UNION operations, we also need to support PRINT-K-SMALLEST(k, i) which prints out the k-th smallest key in a set S_i with index i, utilizing your `quickselect` implementation in the previous problem. k can have different values in different PRINT-K-SMALLEST(k, i) operations. Similarly, on average, PRINT-K-SMALLEST(k, i) should take $O(n)$-time.

   Your implementation should run a sequence of m operations consisting of MAKE-SET, FIND-SET, UNION (but without PRINT-K-SMALLEST(k, i)) in $O(m + n \log n)$-time.

10. (*) (10 points) Give an example that a linear-time sorting algorithm is *slower* than a comparison-based sorting algorithm and explain why.

長度 n

假設要排序的數列裡每個數字都不同

i.e. size of $\Sigma = k = n$

這樣一來 counting sort 就必需花 $n+k = 2n$ 的時間

而如果我們用 quick sort 而且 pivot 剛好選到 best case，

那實際上只要跑 n 的時間

Hashing

11. (*) (10 points) Answer these two questions related to hashing:

    (a) With a load factor α close to 1, explain which of the collision resolution schemes for the hashing table can perform better in terms of running time with an unsuccessful search, i.e., the key to be searched is not in the table. The answer can be one of the following: {*open addressing, chaining, both*}.

    (b) Explain how we can delete a key in the hash table when using *linear probing*, but can still make sure that those already inserted can reliably be found.

(a)

chaining can performing better,

如果是 chaining，我們最多找定對應的格子下 chain 的所有 keys 就結束

如果是 open addressing，我們會一直 probing 直到整個 table 跑完才結束

(b)

我們可以在 delete 之後在空出來的那格做標記，

e.g. 放入 -1

這樣我們如果要 find 一個 key，看到 -1 就往下一個 probe 就好；

如果要 insert，可以將 -1 視為空格子。

12. (**) (15 points) Consider inserting the keys $10, 22, 31, 4, 15, 28, 17, 88, 59, 30$ into an initially empty hash table of length $m = 13$ using the following methods. Show the result after completing each of the last five insertions, i.e., for each method, five answers are required. You do not need to show the intermediate steps for each insertion.

    (a) Double hashing with $h(k, i) = (h_1(k) + i \cdot h_2(k)) \mod m$ where $h_1(k) = k$ and $h_2(k) = 1 + (k \mod (m-1))$

    (b) Chaining $h(k) = k \bmod m$

(a)

	idx: 0	1	2	3	4	5	6	7	8	9	10	11	12
insert 28 →			15		4	31	28			22	10		
insert 17 →			15	17	4	31	28			22	10		
insert 88 →			15	17	4	31	28	88		22	10		
insert 59 →		59	15	17	4	31	28	88		22	10		
insert 30 →		59	15	17	4	31	28	88		22	10	30	

(b)

idx: 0	1	2	3	4	5	6	7	8	9	10	11	12
		15		4	31		59④		22	10		
		↓		↓						↓		
		28①		17②						88③		
				↓								
				30⑤								

① insert 28 in idx 2

② insert 17 in idx 4

③ insert 88 in idx 10

④ insert 59 in idx 7

⑤ insert 30 in idx 4

13. (**) (25 points) Consider the directoryless dynamic hashing scheme introduced in the class. Assume the keys are nonnegative integers. The hash function, given key k and the number of bit i, is defined as $h(k,i) = \text{LSB}(k,i)$. For example, the binary representation of $k = 11$ is $1011_{(2)}$; thus, $h(k,3)$ is the least significant three bits of k, $011_{(2)}$. Given the directoryless hash table shown in Figure 4 with variables $r = 1, q = 0$, and two keys 15 and 8 are already inserted into the table. Assume each table entry holds one key and additional insertion to the same entry is considered overflown. Show the content of the table and the values of the two variable after each of the following insertions in the given order. Note there is no need to show the intermediate steps. Only the final result after each insertion is graded. (1) insert 12; (2) insert 3; (3) insert 7; (4) insert 4; (5) insert 11.

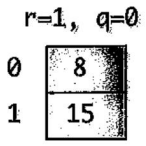
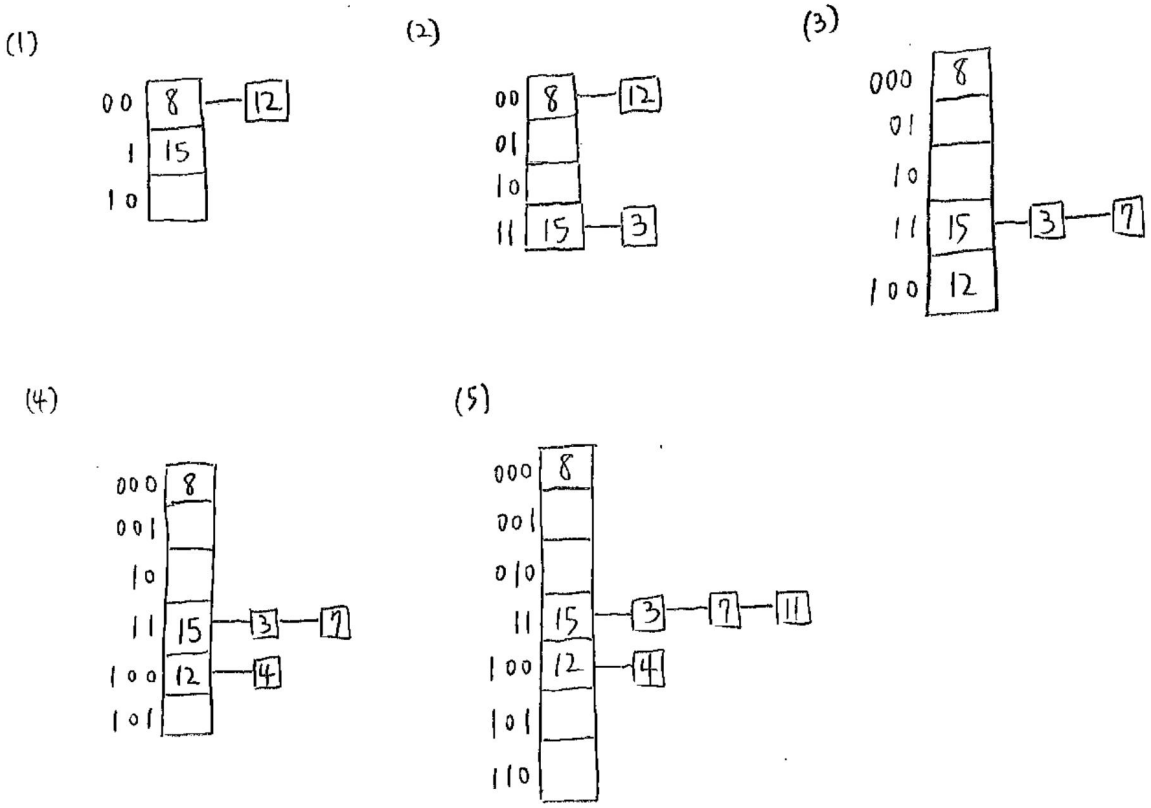


Figure 3: A directoryless hash table with 8 and 15 as inserted keys.



B-Tree

14. (**) (25 points) Perform the following operations on the B-tree with minimum degree $t = 2$ in the figure in the given order: (a) insert 53; (b) insert 40; (c) insert 44; (d) insert 10; (e) delete 35. Show the content of the B-tree after each operation. No intermediate steps are needed.

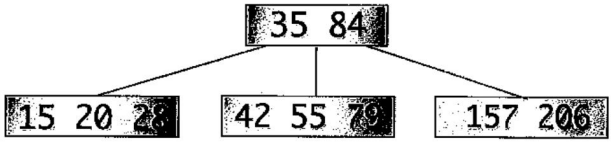
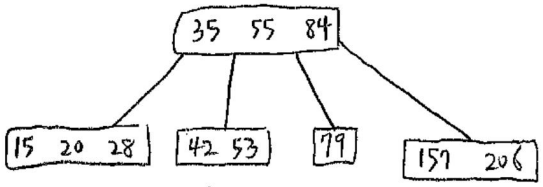
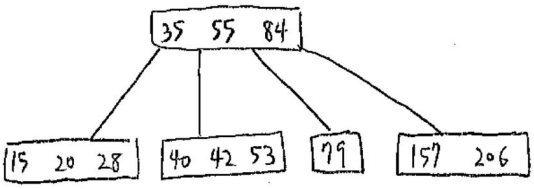


Figure 4: A B-tree

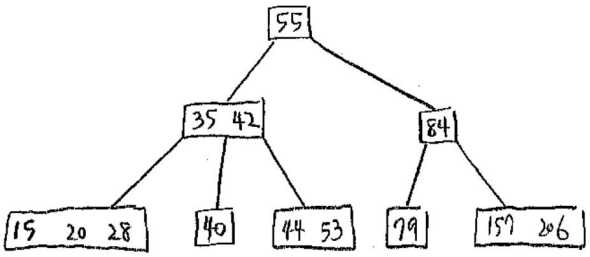
(a)



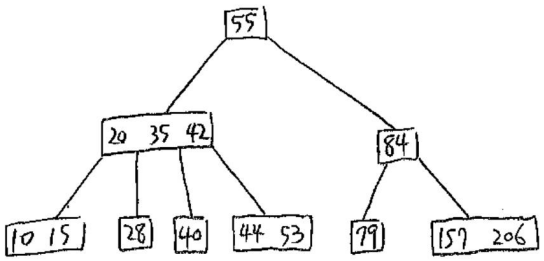
(b)



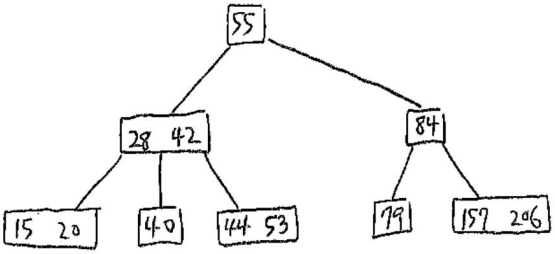
(c)



(d)



(e)



Feedback

15. (*) (10 points) Please give some *constructive* suggestions to the course this semester. In particular, what we did right this semester, and what we should change in the future? Your feedback is particularly important for us to improve the course. We hope most of you can kindly write a few sentences at least (plus it counts the same points as the other problems). Thank you!

很好玩

學到很多東西

如果我這學期沒修 NASA 的話

應該會覺得滿輕鬆的

但 NASA + DSA 真的會死人 ^^