CS-GY 6033: Homework #3

Due on October 11, 2023

Professor Yi-Jen Chiang

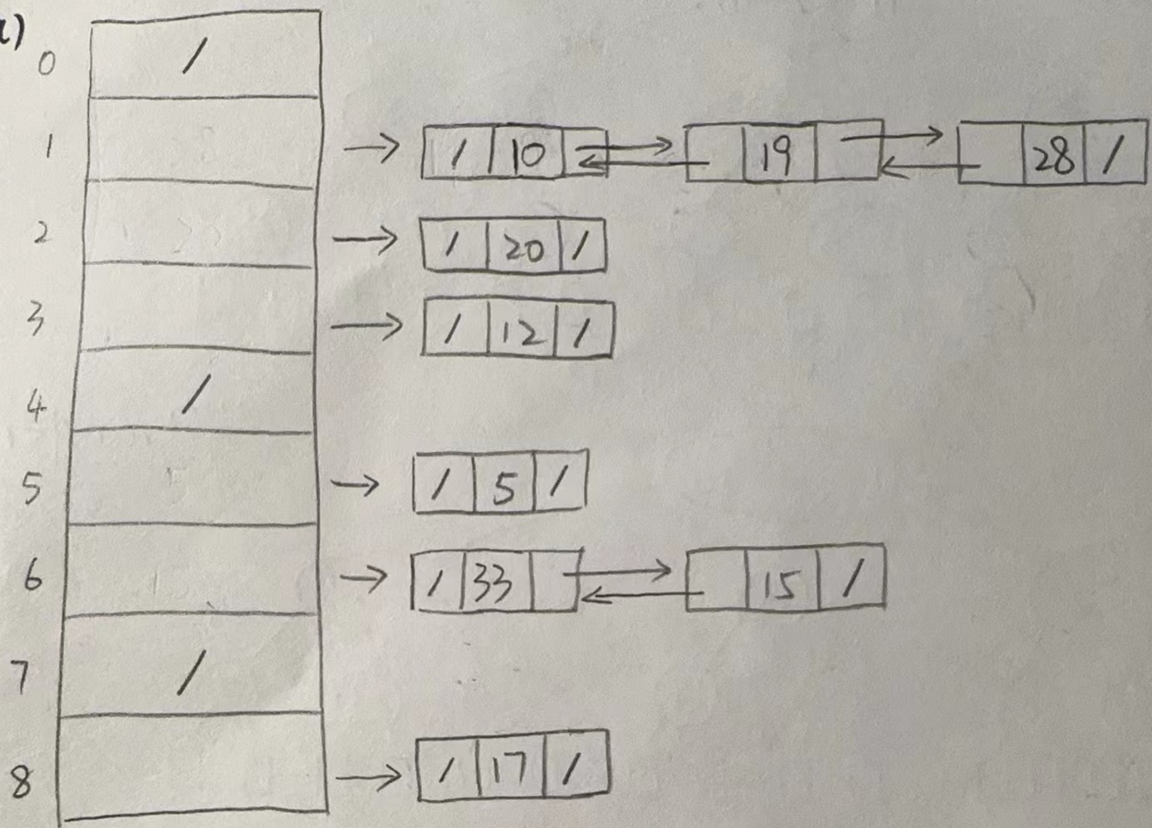
Runze Li
Nxxxxxxxx
rl50xx@nyu.edu

Tzu-Yi Chang
Nxxxxxxxx
tc39xx@nyu.edu

Jiayi Li
Nxxxxxxxx
jl156xx@nyu.edu

October 8, 2023

1. (a)



(b) linear probing: $h(k) = (k+i) \bmod 11$

0	22
1	88
2	–
3	–
4	4
5	15
6	28
7	17
8	59
9	31
10	10

quadratic probing:

$h(k) = (k + i + 3i^2) \bmod 11$

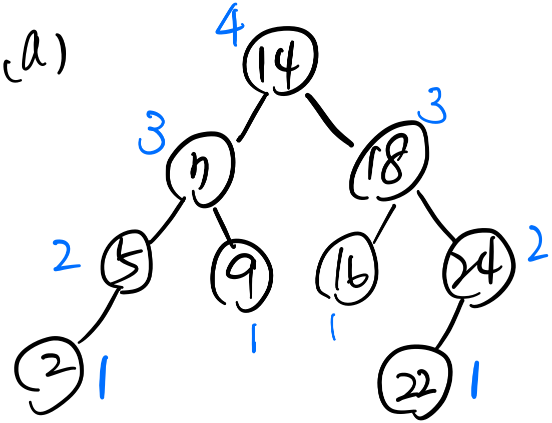
0	22
1	–
2	88
3	17
4	4
5	–
6	28
7	59
8	15
9	31
10	10

double hashing:

$h(k) = (k + (i + k \bmod 10)) \bmod 11$

0	22
1	–
2	59
3	17
4	4
5	15
6	28
7	88
8	–
9	31
10	10

2. (a)



The tree T satisfies that for every node v, the heights of the children of v differ by at most 1.

For example: $BF(14) = 3-3 = 0$

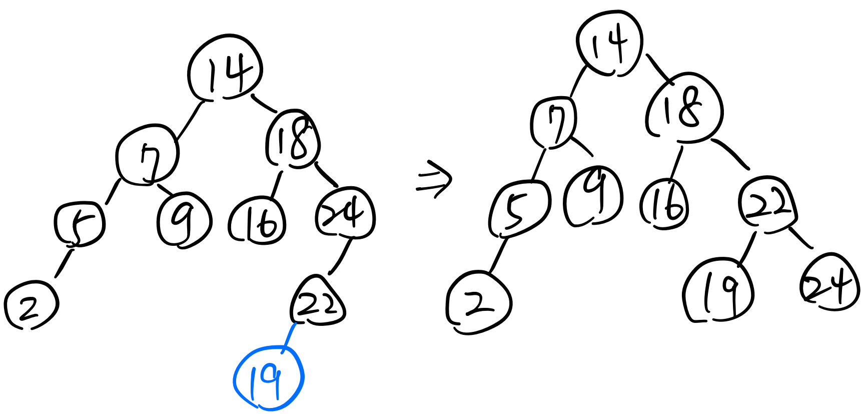
$BF(7) = 2-1 = 1$

$BF(24) = 1-0 = 1$

$\vdots$

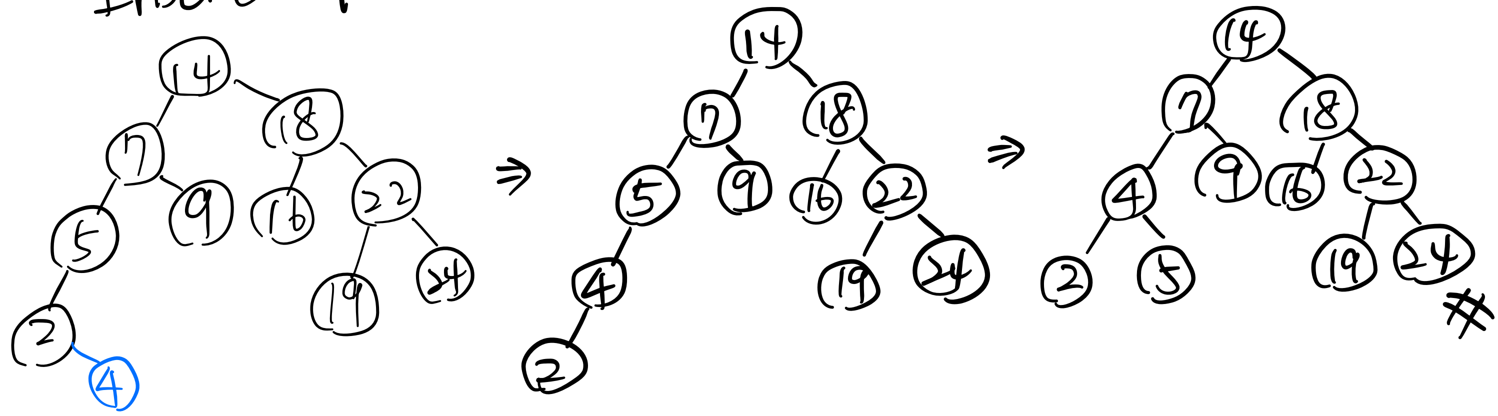
So, this is indeed a valid AVL-tree.

(b) Insert 19



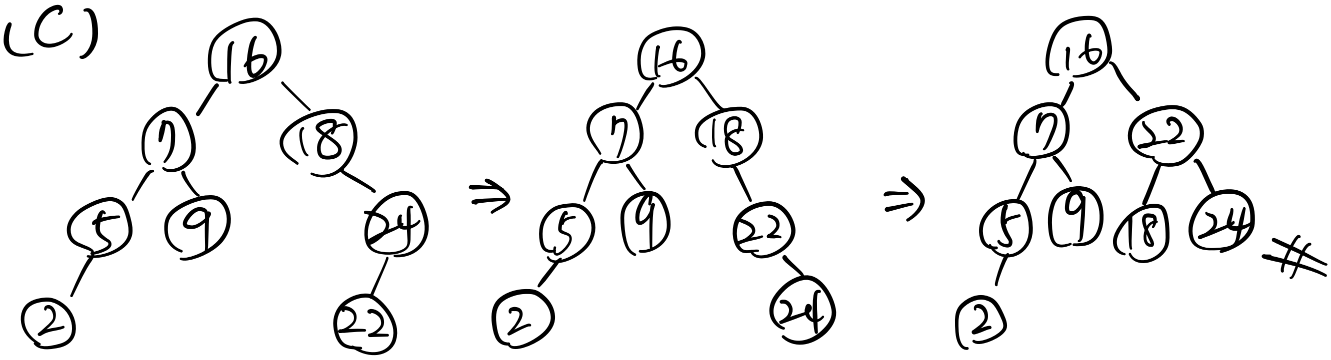
#

Insert 4



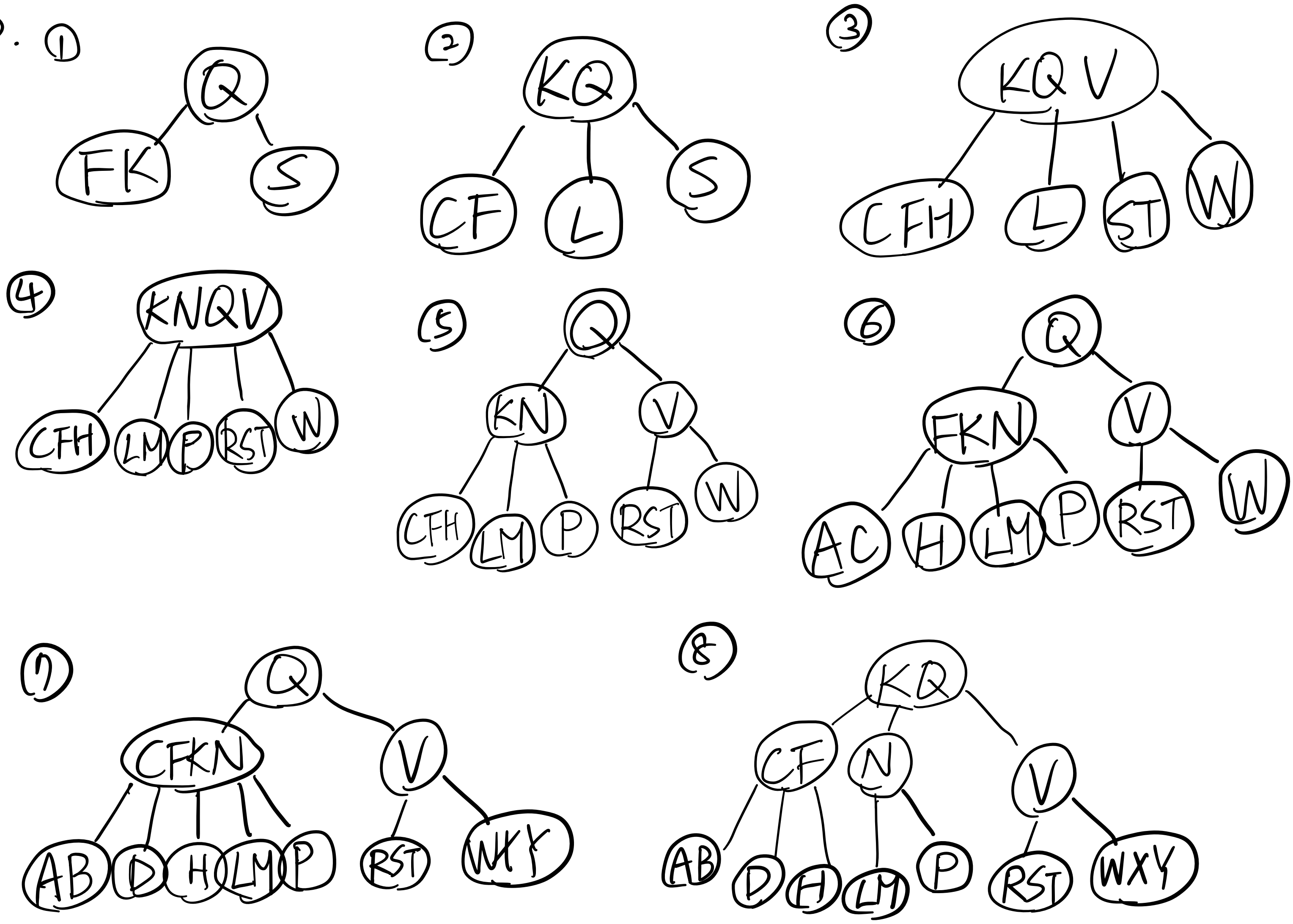
14 7 18 5 9 16 22 2 4 19 24
14 7 18 5 9 16 22 4 19 24 2
14 7 18 4 9 16 22 2 5 19 24
#

(C)



16 7 18 5 9 24 2 22
16 7 18 5 9 22 2 24
16 7 22 5 9 18 24 2
#

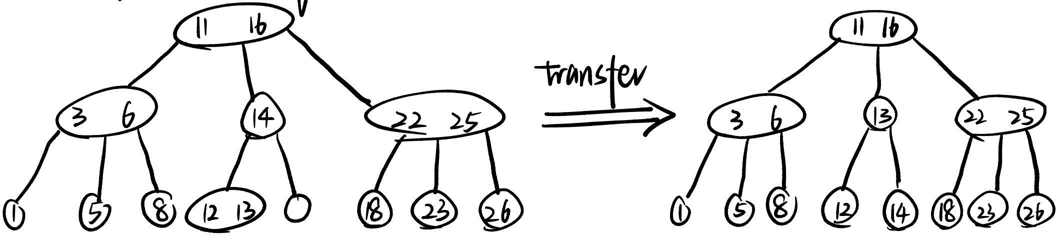
3.



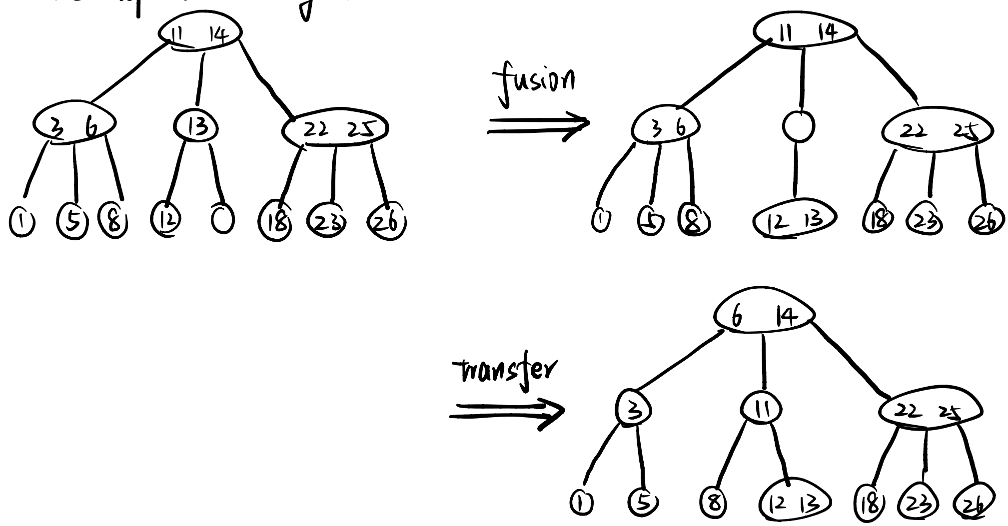
①
Q
FK S
②
KQ
CF L S
③
KQV
CFH L ST W
④
KNQV
CFH LM P RST W
⑤
Q
KN V
CFH LM P RST W
⑥
Q
FKN V
AC H LM P RST W
⑦
Q
CFKN V
AB D H LM P RST WXY
⑧
KQ
CF N V
AB D H LM P RST WXY

Q4.

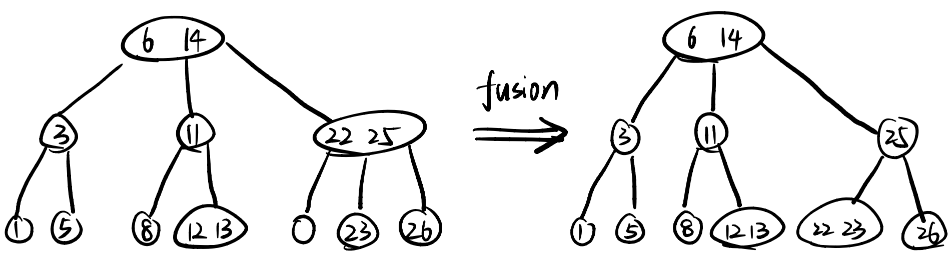
First step: delete key 17



Second step: delete key 16



Third step: delete key 18



5. a. Let X_i be a random variable for the number of probes in an unsuccessful search. Define A_j to be the event that the jth probe is to an occupied slot.

$$\Pr\{A_1\} = \frac{n}{m} \quad \Pr\{A_2 | A_1\} = \frac{n-1}{m-1} = \frac{\Pr\{A_2 \cap A_1\}}{\Pr\{A_1\}}$$

$$\because \Pr\{A_2|A_1\} = \frac{\Pr\{A_2 \cap A_1\}}{\Pr\{A_1\}} \quad \therefore \Pr\{A_2 \cap A_1\} = \Pr\{A_1\} \cdot \Pr\{A_2|A_1\} = \frac{n}{m} \cdot \frac{n-1}{m-1}$$

$$\begin{aligned}\Pr\{X_i > p\} &= \Pr\{A_1 \cap A_2 \cap A_3 \cap \cdots \cap A_p\} \\ &= \Pr\{A_1\} \cdot \Pr\{A_2|A_1\} \cdot \Pr\{A_3 | A_1 \cap A_2\} \cdots \Pr\{A_p | A_1 \cap \cdots \cap A_{p-1}\} \\ &= \underbrace{\frac{n}{m} \cdot \frac{n-1}{m-1} \cdot \cdots \cdot \frac{n-(p-1)}{m-(p-1)}}_{p \text{ terms}} \\ &\le \left(\frac{n}{m}\right)^p \\ &\le \left(\frac{1}{2}\right)^p = 2^{-p}\end{aligned}$$

b. $$\begin{aligned}\Pr\{X_i > 2\lg n\} &= \frac{n}{m} \cdot \frac{n-1}{m-1} \cdots \cdot \frac{n-(2\lg n-1)}{m-(2\lg n-1)} \\ &\le \left(\frac{n}{m}\right)^{2\lg n} \\ &\le \left(\frac{1}{2}\right)^{2\lg n} = 2^{-2\lg n} = \frac{1}{n^2} = O\left(\frac{1}{n^2}\right)\end{aligned}$$

c. Let the random variable $X = \max\{X_i : 1 \le i \le n\}$

$$\begin{aligned}\Pr\{X > 2\lg n\} &= \Pr\{X_1 > 2\lg n\} \cup \Pr\{X_2 > 2\lg n\} \cup \cdots \cup \Pr\{X_n > 2\lg n\} \\ &\le \sum_{i=1}^{n} \Pr\{X_i > 2\lg n\} \quad \text{(by Boole's inequality)} \\ &\le \sum_{i=1}^{n} \frac{1}{n^2} = n \cdot \frac{1}{n^2} = \frac{1}{n} = O\left(\frac{1}{n}\right)\end{aligned}$$

d. $$E[X] = \sum_x x \Pr\{X=x\} = \sum_{x \le t} x \Pr\{X=x\} + \sum_{x>t} x \Pr\{X=x\}.$$

$$\begin{aligned}E[X] &\le 2\lg n \Pr\{X \le 2\lg n\} + n \Pr\{X > \lg n\} \\ &= 2\lg n\left(1-\frac{1}{n}\right) + n\left(\frac{1}{n}\right) \\ &= 2\lg n - \frac{2\lg n}{n} + 1 \\ &= O(\lg n)\end{aligned}$$

Problem 6

Given n distinct, unsorted integers where their value range is >> n, design and analyze an algorithm to report all pairs of integers (p, q) among them such that $p = 3q - 2$. Your algorithm should run in $O(n)$ expected time and $O(n)$ worst-case space.

Solution

We can use **hash table** to solve this problem. For example, if we use C++ as programming language, we can use *unordered_map* to implement it. In the first loop, for each integer *num* in the array, we insert them into the hash table. In the second loop, for any integer q, we need to find if there exists an integer $p = 3q - 2$ in the hash table. If yes, we add the pair (p, q) to the resulting pairs. Finally we can find all pairs of integers (p, q) where $p = 3q - 2$.

The pseudo code **Find-Pairs** is as follows:

Algorithm 1 Find-Pairs

```
Input: arr, n                                         ▷ An array with n items
Output: pairs                          ▷ A list of pairs (p, q) where p = 3q-2
Initialize an empty list pairs
Initialize an empty Hash Table called hash_table
for each integer num in arr do
    Add num to hash_table                 ▷ Add the elements into Hash table
end for
for each integer q in arr do
    p ← 3q − 2
    if hash_table contains key p then                 ▷ Search in Hash table
        Add (p, q) to pairs
    end if
end for
Return pairs
```

In the algorithm above, we use two loops, whose expected time complexity is $O(n)$. In the first loop, there is an insertion operation in the hash table, which takes constant time, i.e. $O(1)$. In the second loop, there is a searching operation in the hash table, which has the time complexity of $O(1)$. So the expected time of the algorithm is $O(n)$.

In the worst case, every element in the input array is distinct and they are put in the hash table. Therefore, the required space in the hash table is proportional to the number of elements in the input array, which is n. As a result, the worst-case space complexity is $O(n)$.

In summary, this algorithm can run in $O(n)$ expected time and $O(n)$ worst-case space.