UNIVERSITY OF MICHIGAN - SHANGHAI JIAO TONG UNIVERSITY
JOINT INSTITUTE

Homework #4

Name & ID: 周雅凯 518021911039

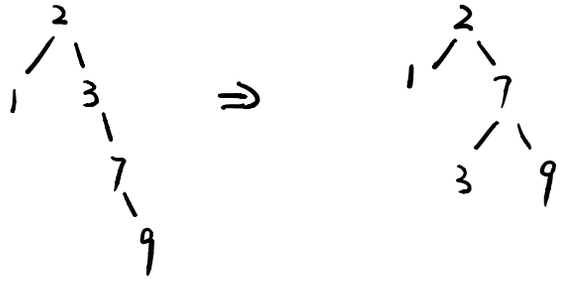
Due Date: *Dec 15th, 2020*

Question 1

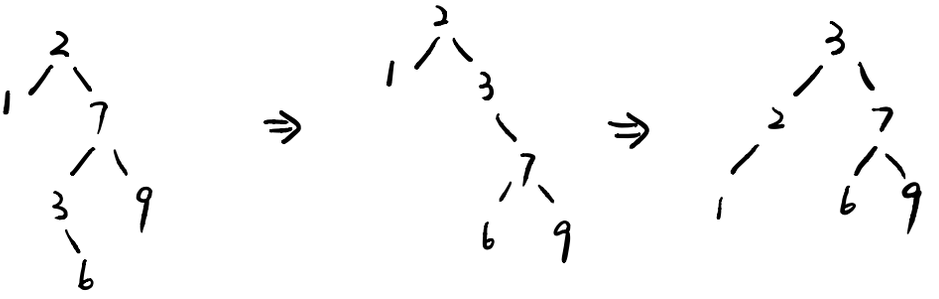
1. Suppose that we insert a sequence of keys 2, 1, 3, 7, 9 into an initially empty AVL tree. Draw the resulting AVL tree.
2. Suppose that we further insert key 6 into the AVL tree you get in Problem (1). Draw the resulting AVL tree.
3. Suppose that we further insert keys 4, 5 into the AVL tree you get in Problem (2). Draw the resulting AVL tree.
4. Suppose that we further insert keys 8, 10, 11, into the AVL tree you get in Problem (3). Draw the resulting AVL tree.
5. Write down the balance factor for each node in the final AVL tree you get in Problem (4).

Answer.

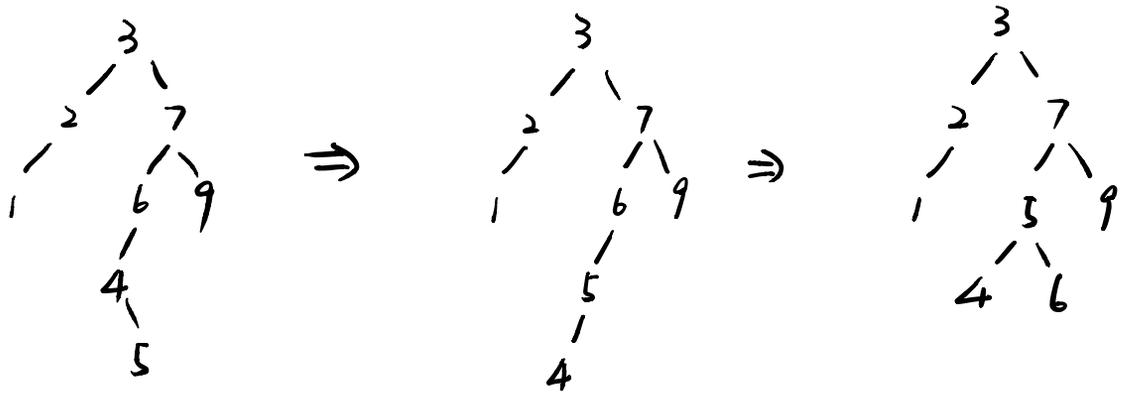
1.



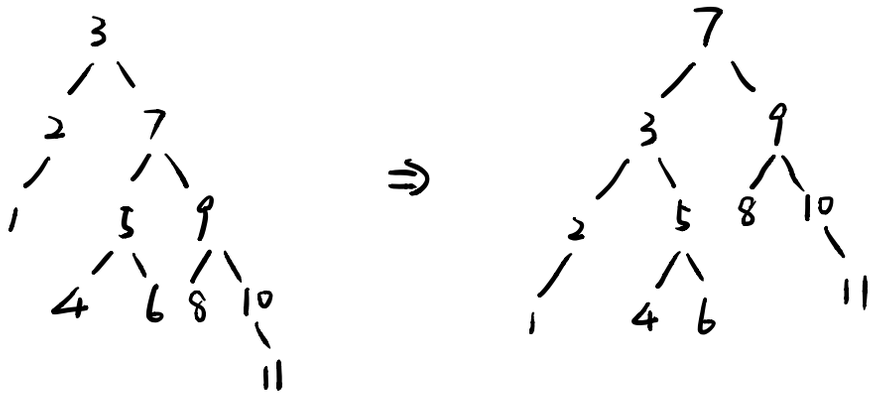
2.



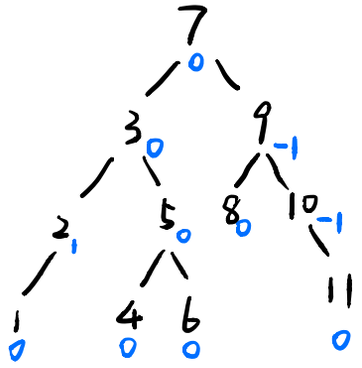
3.


3
2
7
1
6
9
4
5
⇒
3
2
7
1
6
9
5
4
⇒
3
2
7
1
5
9
4
6

4.


3
2
7
1
5
9
4
6
8
10
11
⇒
7
3
9
2
5
8
10
1
4
6
11

5.


7
0
3
0
9
-1
2
1
5
0
8
0
10
-1
1
0
4
0
6
0
11
0

Question 2

Starting with vertex 7, plot the MST for the graph above using Prim's algorithm. Please show the intermediate steps (including 8 MST graphs).

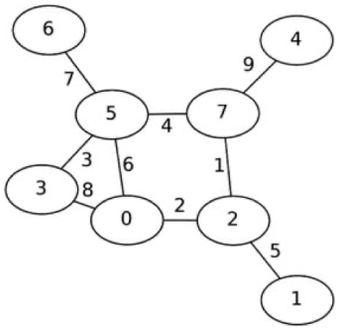
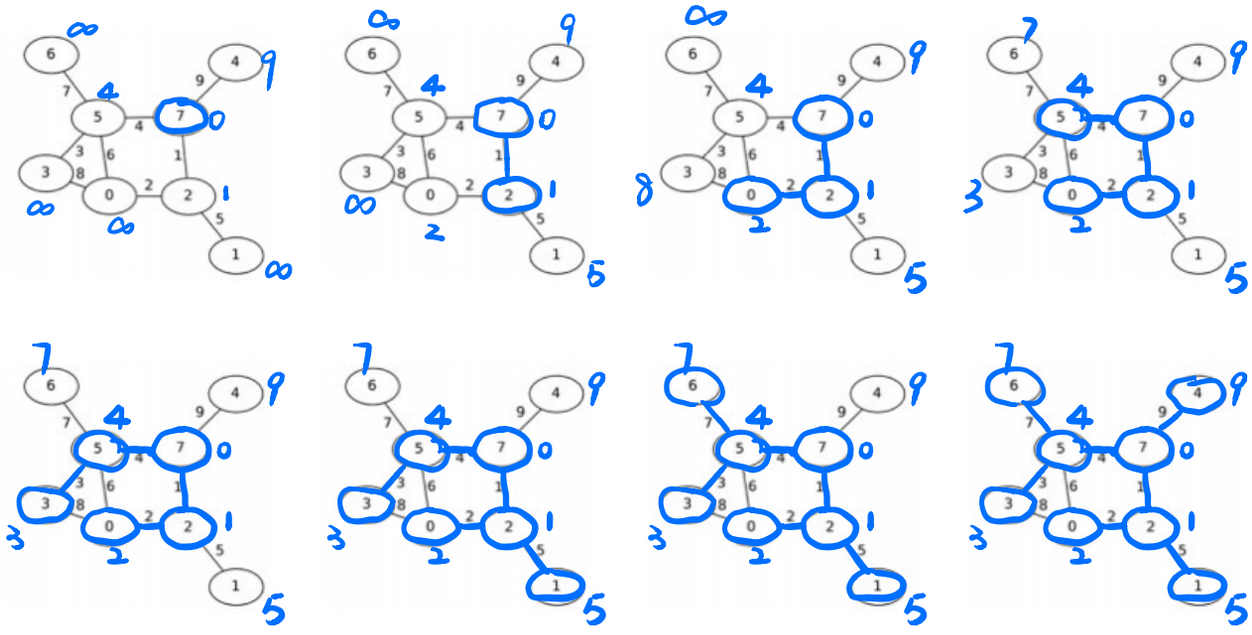


Figure 1: graph

Answer.



Question 3. Safe the Blue Tiger!

The blue tiger is stuck in a $m \times n$ grid, where there is non-negative number in each cell of the grid. Notice that the blue tiger **can only** move down or left in this game. In this homework, we are going to apply what we learnt and find a path from the top right of the grid to the bottom left of the grid. We should sum up all the numbers along the path and our goal is to minimize the sum. We need the time complexity to be $\mathcal{O}(n \times m)$ and the space complexity to be $\mathcal{O}(\min(m, n))$. You only need to write down the psudeo code.

Sample: Assume the grid of size 3×4 is represented as

8 8 1 1
8 8 1 8
1 1 1 8

The minimum sum is 6, by design the path as the one that cross all the 1s in the grid.

Template:

```
int solution(const int grid[][]){
    //To do
}
```

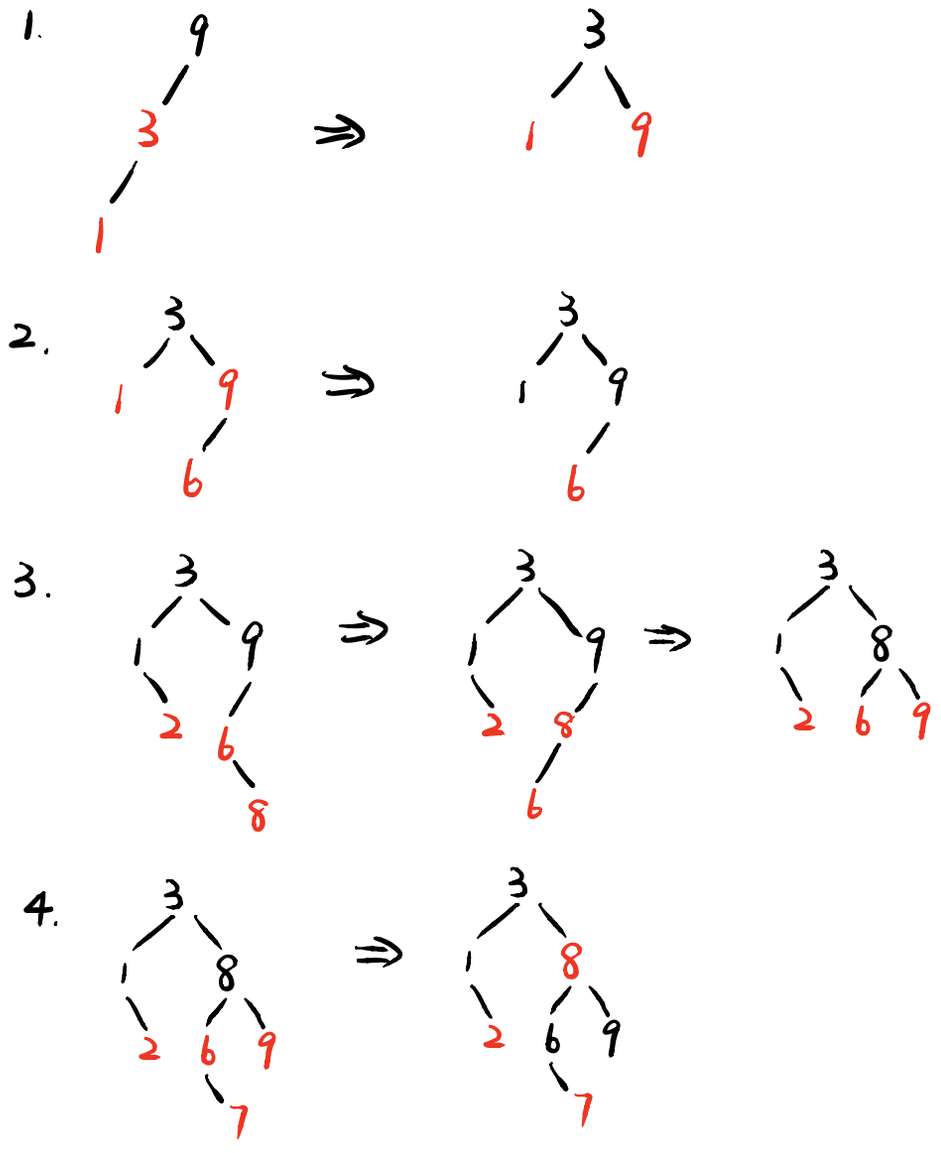
Answer.

```
int solution (const int grid[][]) {
    int row =(sizeof(grid)/sizeof (int))/(sizeof (grid[0])/sizeof (int));
    int column = sizeof(grid[0])/ sizeof(int);
    if ( row ==0 || column ==0) {
        return 0;
    }
    vector<int>   dp(column);    // we assume column ≤ row
    dp[column-1] = grid[0][column-1];
    for(int i= column -2; i>=0; --i) {
        dp[i]=dp[i+1]+ grid[0][i];
    }
    for (int i=1; i<row; ++i) {
        dp[column-1] += grid[i][colum-1];
        for (int j= column -2; j>=0; --j){
            dp[j]= min( dp[j], dp[j+1])+ grid[i][j];
        }
    }
    return dp[0];
}
```

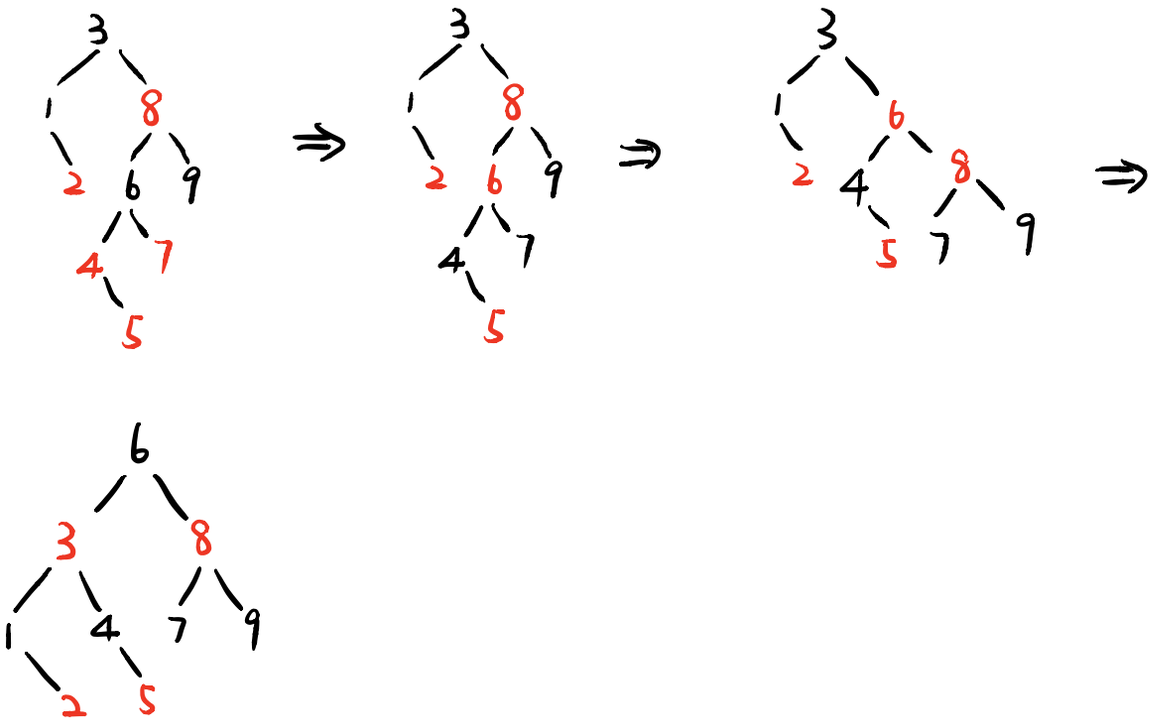
Question 4. RB Tree

1. Suppose that we insert a sequence of keys 9, 3, 1 into an initially empty red-black tree. Draw the resulting red-black tree.
2. Suppose that we further insert key 6 into the red-black tree you get in Problem (1-a). Draw the resulting red-black tree.
3. Suppose that we further insert keys 2, 8 into the red-black tree you get in Problem (1-b). Draw the resulting red-black tree.
4. Suppose that we further insert key 7 into the red-black tree you get in Problem (1-c). Draw the resulting red-black tree.
5. Suppose that we further insert keys 4, 5 into the red-black tree you get in Problem (1-d). Draw the resulting red-black tree.

Answer:.



5.



Exercise 5. Involution master

There is a master of involution here at JI and s/he is among us! Our master is trading on stock market. He his rich with infinite amount of cash. However, this stock market has only 1 company listed and our master can only hold at most 1 share at any given time. Therefore, he must sell his share before he can buy again. On top of that, our master can make at most two transactions (2 buys and 2 sells). Despite all the difficulties, our master can foresee into the future and know the stock price until Armageddon (the day that stock market closes). Assume that our master stores the stock prices of the week that he foresees into an array called *price*. The ith element in the *price* array is the stock price at the ith day. Please come up with an dynamic programming algorithm that help our master make the most wise trading decisions which maximizes the profit and outputs the maximum profit. Additional objective: the involution master is inpatient. He needs your algorithm to finish in $\mathcal{O}(n)$ time with n being the total number of days and in $\mathcal{O}(1)$ space.
Here are some examples.

Example 1:
Input: prices = [3,3,5,0,0,3,1,4]
Output: 6
Explanation: Buy on day 4 (price = 0) and sell on day 6 (price = 3), profit = 3-0 = 3. Then buy on day 7 (price = 1) and sell on day 8 (price = 4), profit = 4-1 = 3.

Example 2:
Input: prices = [1,2,3,4,5]
Output: 4
Explanation: Buy on day 1 (price = 1) and sell on day 5 (price = 5), profit = 5-1 = 4. Note that you cannot buy on day 1, buy on day 2 and sell them later, as you are engaging multiple transactions at the same time. You must sell before buying again.

Example 3:
Input: prices = [7,6,4,3,1]
Output: 0
Explanation: In this case, no transaction is done, i.e. max profit = 0.

Example 4:
Input: prices = [1]
Output: 0

Template:

```
class Solution {
public:
    int maxProfit(vector<int>& prices) {
        //To do

    }
}
```

Help passing on the torch of involution!

Answer:.

```
int maxProfit (vector<int>& prices) {
    size_t n = prices.size();
    if (n==0) {
        return 0;
    }
    int fstbuy = -prices[0];
    int secbuy = INT_MIN;
    int fstsell = secsell = 0;
    for (size_t i = 1; i<n; ++i) {
        secsell = max(secsell, secbuy+prices[i]);
        secbuy = max(secbuy, fstsell-prices[i]);
        fstsell = max(fstsell, fstbuy+prices[i]);
        fstbuy = max(fstbuy, -prices[i]);
    }
    return max(fstsell, secsell);
}
```