Assignment#2 Report

Team name: Yunfan Li, Haichao Zhang, Jingyuan Xu

Course: CS325-001

Algorithm 1:

Pseudocode:

```
N, M, T
key[M], ball[T], box[N], newKey[M]
index ← 1
recursive (level, state) // level means current key, state means whether we use
                          // this key or not
    if level != M+1
        if state = 1
            newKey[index] = key[level]
            index ← index + 1
            res ← recursive (level+1, 1)
            index ← index - 1
            res ← recursive (level+1, 0)
            index ← index - 1
        else
            res ← recursive (level+1, 1)
            index ← index - 1
            res ← recursive (level+1, 0)
            index ← index - 1
    else //if level = M+1
        front ← calculate how many lockers should be unlocked before
                 the first key
        back ← calculate how many lockers should be unlocked after
                 the last key
        for i ← 1 to (index-1)
            k ←0
            for j ← newKey[i-1]+1 to newKey[i]
                openRight[N], openLeft[N]
                if box[i] = 1
                    openRight[k] ← j - newKey[i-1]
                    openLeft[k] ← newKey[i] - j +1
                    k ← k+1
            tmpMin ← INFINITY
            for j ← 1 to (k-2)
                if tmpMin > openRight[k] + openLeft[k+1]
                    tmpMin = openRight[k] + openLeft[k+1]
            min ← min + tmpMin
        res ← front + min + back
        return res
```

Time Analysis:
For the recursive process, we will find out all the possibility of keys combinations. So all sum of keys combination is 2^M. For every single keys combination, we search for every pair of keys, from left key plus 1 position to right key, calculate how many doors need to be opened in order to get all the balls between these two keys. This part of for-loop is N. So the total running time will be O(N2^M). The big-Omega appears when we do not have any balls in these lockers, so it is Omega(2^M).

Solutions:

```
Test1: 11
Test2: 14
Test3: 7
Test4: 14
Test5: 18
Test6: 1
Test7: 15
Test8: 8
```

Algorithm 2:

Pseudocode:

```
if do not have balls before the first key
    d[1][0] ← 0
    d[1][1] ← 1
else //have balls before the first key
    d[1][0] ← INFINITY
    d[1][1] ← key[1] - ball[1] + 1
for i ← 2 to M
    if have ball between key[i-1] and key[i]
        d[i][0] ← min (d[i-1][0]+(minimum of doors need to be opened
                  between key[i-1] and key[i])+1, d[i-1][1]+ (minimum of
                  doors need to be opened between key[i-1] and key[i]))
        for j ← key[i-1]+1 to key[i]
            for k ← j-1 down to key[i-1]
                if box[k] = 1
                    break;
            if k = key[i-1]
                leftNotOpen ← 0
                leftOpen ← 0
            else
                leftNotOpen ← k – key[i-1] + 1
                leftOpen ← k –key[i-1]
            for k ← j to key[i]
                if box[k] = 1
                    break;
            if k = key[i]+1
                r ← 1
            else
                r ← key[i] – k +1
            sum_leftNotOpen = min(sum_leftNotOpen, r+leftNotOpen)
            sum_leftOpen = min(sum_leftOpen, r+leftOpen)
        d[i][1] = min(d[i-1][0]+sum_leftNotOpen, d[i-1][1]+sum_leftOpen)
if have balls after the last key
    res ← min(d[M][0]+ball[T]-key[M]+1, d[M][1]+ball[T]-key[M])
else
    res ← min(d[M][0], d[M][1])
```

Time Analysis:
For the most outer for-loop is M, and inner for-loops is NM, so the total running time is O(NM^2). The big-Omega appears when there is no balls in these lockers, so it is Omega(M).

Solutions:
```
Test1: 96
Test2: 22
Test3: 68
Test4: 31
Test5: 103
Test6: 30
Test7: 87
Test8: 83
```