Dynamic Programming Assignment

1. Falling glass

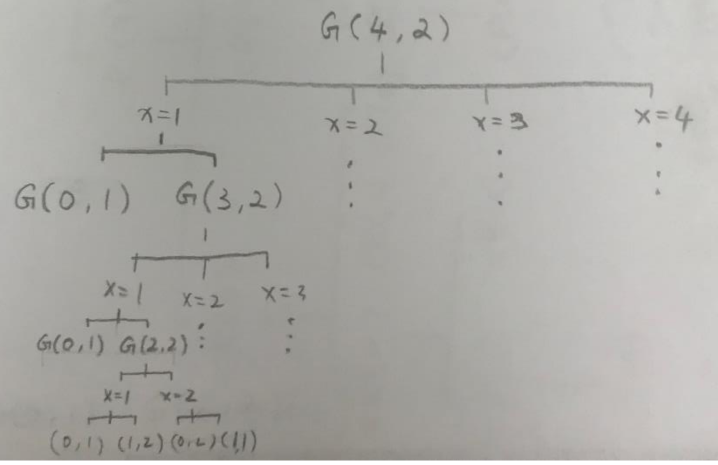
a) Optimal substructure / recurrence

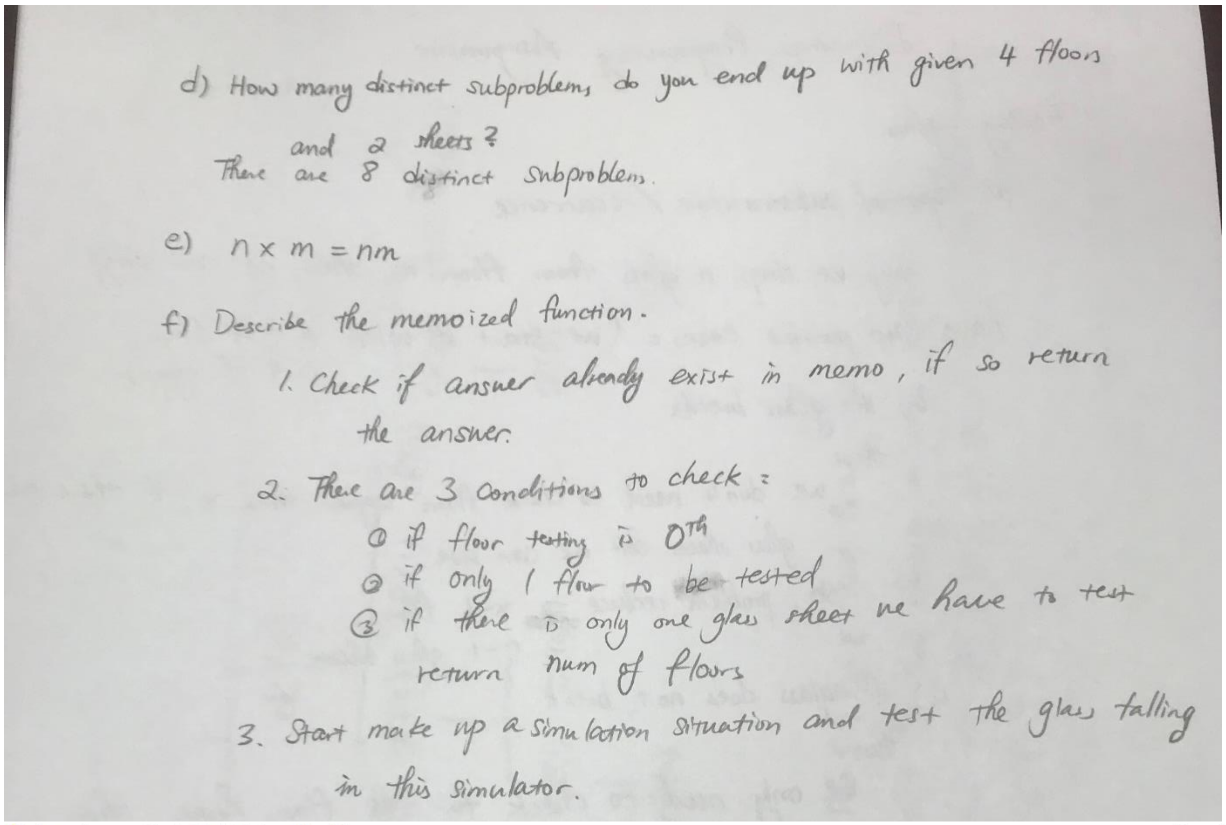
Say we drop a glass from floor x, then we can only have two possible cases : (we start off with k # of floors, n # of eggs)

i) if glass breaks
then
we don't need to check floors upper than x, if there are glass sheets left we can use.
So, problem reduce to x−1 floors, n−1 glass sheets.

ii) if glass does not break
then
we only need to check for the floors higher than x. if there are any glass sheets left.
So, problem reduces to k−x floors,
since glass sheet doesn't break, the # of glass sheets remains the same : n glass sheets.

b) recurrence tree for given (floors = 4, sheets = 2)



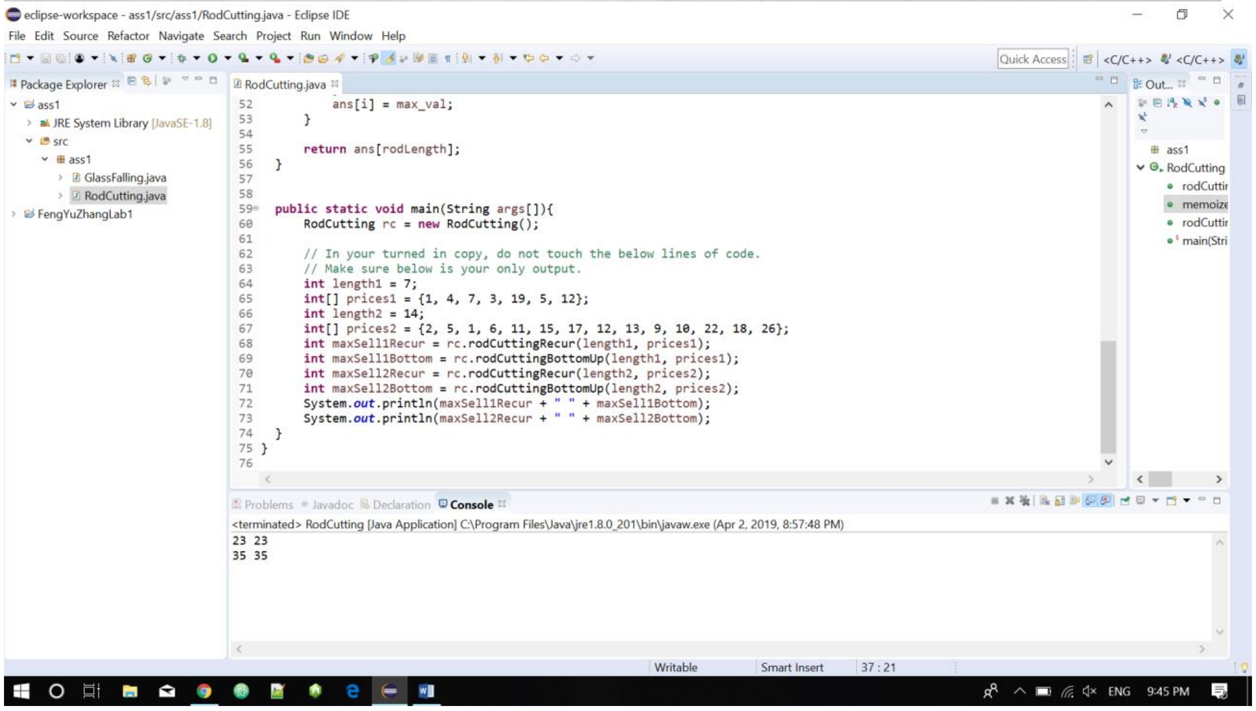


d) How many distinct subproblems do you end up with given 4 floors and 2 sheets?
There are 8 distinct subproblems.

e) $n \times m = nm$

f) Describe the memoized function.

1. Check if answer already exist in memo, if so return the answer.
2. There are 3 Conditions to check:
   ① if floor testing is 0^{th}
   ② if only 1 floor to be tested
   ③ if there is only one glass sheet we have to test
   return num of floors
3. Start make up a simulation situation and test the glass falling in this simulator.



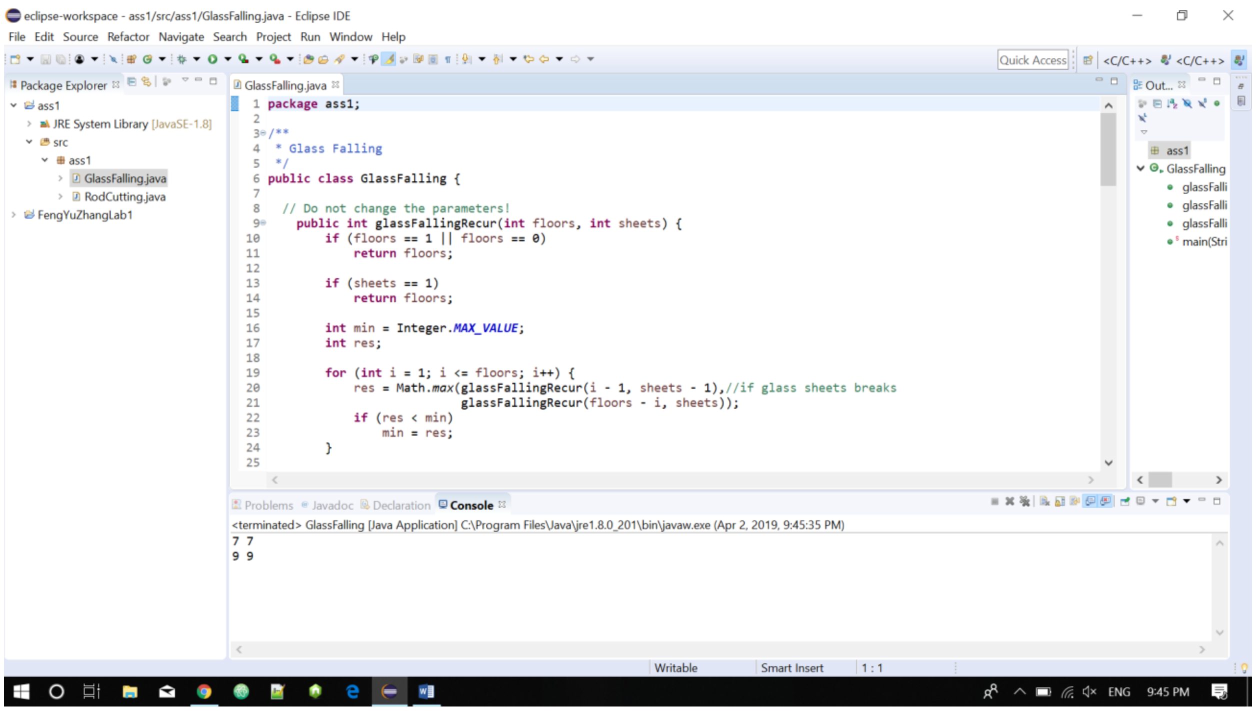
```java
            ans[i] = max_val;
        }

        return ans[rodLength];
    }


    public static void main(String args[]){
        RodCutting rc = new RodCutting();

        // In your turned in copy, do not touch the below lines of code.
        // Make sure below is your only output.
        int length1 = 7;
        int[] prices1 = {1, 4, 7, 3, 19, 5, 12};
        int length2 = 14;
        int[] prices2 = {2, 5, 1, 6, 11, 15, 17, 12, 13, 9, 10, 22, 18, 26};
        int maxSell1Recur = rc.rodCuttingRecur(length1, prices1);
        int maxSell1Bottom = rc.rodCuttingBottomUp(length1, prices1);
        int maxSell2Recur = rc.rodCuttingRecur(length2, prices2);
        int maxSell2Bottom = rc.rodCuttingBottomUp(length2, prices2);
        System.out.println(maxSell1Recur + " " + maxSell1Bottom);
        System.out.println(maxSell2Recur + " " + maxSell2Bottom);
    }
}
```

```
<terminated> RodCutting [Java Application] C:\Program Files\Java\jre1.8.0_201\bin\javaw.exe (Apr 2, 2019, 8:57:48 PM)
23 23
35 35
```



```
package ass1;

/**
 * Glass Falling
 */
public class GlassFalling {

  // Do not change the parameters!
    public int glassFallingRecur(int floors, int sheets) {
        if (floors == 1 || floors == 0)
            return floors;

        if (sheets == 1)
            return floors;

        int min = Integer.MAX_VALUE;
        int res;

        for (int i = 1; i <= floors; i++) {
            res = Math.max(glassFallingRecur(i - 1, sheets - 1),//if glass sheets breaks
                           glassFallingRecur(floors - i, sheets));
            if (res < min)
                min = res;
        }

```

```
<terminated> GlassFalling [Java Application] C:\Program Files\Java\jre1.8.0_201\bin\javaw.exe (Apr 2, 2019, 9:45:35 PM)
7 7
9 9
```