Homework 2

Problem 5

1. Let M_i denote the maximum sum of the non-decreasing subsequence up to S_i for $i = 1, 2, \ldots, |S|$. The recursion formula is written as

$$M_i = \begin{cases} S_i \text{ if } i = 1 \text{ or } S_j > S_i \ \forall j < i \\ \max\{M_j + S_i \mid j < i \text{ and } S_j \leq S_i\} \end{cases}$$

   First, let's prove the correctness of the recursion formula by induction.

   - **Inductive hypothesis**: M_n is correctly calculated with the formula.
   - **Basis**: when $n = 1$, $M_1 = S_1$, which is always correct.
   - **Inductive step**: If $S_j > S_{n+1} \ \forall j < n + 1$, then M_{n+1} is S_{n+1} itself because it is the start of a non-decreasing subsequence. Otherwise, if $\exists j < n + 1$ s.t. $S_j \leq S_{n+1}$, then a new non-decreasing subsequence is generated by concatenating S_{n+1} to the end of any non-decreasing subsequences up to S_j, of which the maximum sum is M_j. Therefore, M_{n+1} is the maximum of $\{M_j + S_{n+1} \mid j < n + 1 \text{ and } S_j \leq S_{n+1}\}$.
   - **Conclusion**: the recursion formula is correct.
   - **Time complexity**: To calculate each M_i for $i = 2, \ldots, |S|$, we have to loop through $j = 1, 2, \ldots, i - 1$ and keep track of the maximum if $S_j \leq S_i$. Therefore, the time complexity is $1 + \sum_{i=2}^{n}(i - 1) = O(n^2)$.

2. a. The $n \times m$ grid is filled in column by column from left to right. First, a configuration of a column is defined as a unique way of placing objects in that column. For a column with n rows, there are exactly 2^n unique configurations, numbered from 1 to 2^n. First, S, M, and I are defined as follows:
   - S_j: the number of valid ways to put objects into an $n \times j$ grid.
   - $M_{i,j}$: the number of valid ways to put objects into an $n \times j$ grid given the configuration of the j-th column is i.
   - $I_{i,j}$: the set of valid configurations of $(j - 1)$-th column given the configuration of j-th column is i.

   With these notations, the subproblem and the transition function can be expressed as follows:

   Subproblem: $S_j = \sum_{i=1}^{2^n} M_{i,j}$

   Transition function: $M_{i,j} = \begin{cases} 0 \text{ if configuration } i \text{ is not valid} \\ 1 \text{ if configuration } i \text{ is valid and } j = 1 \\ \sum_{i \in I_{i,j}} M_{i,j-1} \text{ if configuration } i \text{ is valid and } j > 1 \end{cases}$

   b. The steps in the analysis of time complexity:

   - Given the configuration of $(j - 1)$-th and j-th column, it takes $O(n)$ to check if it is valid.
   - To calculate $M_{i,j}$, we have to loop through 2^n possible configurations of $(j - 1)$-th column. Time complexity: $O(2^n \times n)$.
   - To calculate S_j, we have to loop through all 2^n configurations of j-th column. Time complexity:

$O(2^n \times 2^n \times n)$.

- The solution is S_m, namely, there are m S_j's to compute. Therefore, the overall time complexity: $O(2^n \times 2^n \times n \times m) = O(4^n \times n \times m)$.

Problem 6

1. Three.
2. Define variables as follows: j be the largest index $< i$ s.t. balloon j is not removed from the sequence of balloons, and x be the number of shots. Pseudocode is given as follows:
    1. Sort n balloons by y^h in increasing order. If y^h are the same, then sort by y^l in non-increasing order. $\Rightarrow O(n \log n)$
    2. $j = 0 \Rightarrow \Theta(1)$
    3. **for** balloon $i = 1, \ldots, n-1$: $\Rightarrow \Theta(n)$
        1. **if** $y_i^l \leq y_j^l$:
            1. Remove balloon i from the sequence
        2. **else**:
            1. $j = i$
    4. n' = the number of balloons left $\Rightarrow \Theta(1)$
    5. $x = 0$ and $i = 0 \Rightarrow \Theta(1)$
    6. **while** $i \neq n'$ (there is any balloon left):
        1. $x = x + 1$ and shoot at $y = y_i^h$
        2. **while** $y \in [y_i^l, y_i^h]$:
            1. $i = i + 1$
    7. **return** x

    - **Time complexity**: The while loop in step 6 is entered at most n times since at least one balloon is shot in each loop. Therefore, step 6.1 takes $O(n)$. Likewise, the while loop in step 6.2 is entered exactly n times. Therefore, step 6.2.1 takes $\Theta(n)$. Altogether, the time complexity of the given algorithm is upper bounded by the sorting in Step 1, which is $O(n \log n)$.
3. - Step 1 & 2: If balloon $i-1$ is completely contained in balloon i, then balloon i can be removed because whenever balloon $i-1$ is shot broken, balloon i is shot broken as well. The resulting sequence of balloons satisfying the following two properties: (1) y^h in increasing order, and (2) y^l in increasing order.
    - **Optimal substructure**: Given a sequence of sorted balloons. Define M_i be the optimal solution for the first i balloons. Define $p(i)$ be the largest index $i' < i$ s.t. balloon i' and balloon i do not overlap. The subproblem is given is follows:

$$M_i = \begin{cases} 1 \text{ if } i = 0 \text{ or } p(i) \text{ does not exist} \\ 1 + M_{p(i)} \text{ otherwise} \end{cases}$$

Those overlapping with balloon i can be broken in one shot, and those not overlapping with balloon i are counted in $M_{p(i)}$. If M_i is optimal, then $M_{p(i)}$ has to be optimal; otherwise, M_i can be even smaller.

- **Greedy choice property**: Given a sequence of i sorted balloons. A greedy algorithm is to shoot at the y^l of the last balloon left in the sequence. Suppose the optimal solution does not include this greedy choice. Then, shooting position of the first shot $y > y_i^l$ will leave j balloons unbroken s.t. $j \geq p(i)$. Since M is a non-decreasing sequence, we know $M_j \geq M_{p(i)}$. However, this is a contradiction to that the optimal solution does not include this greedy choice because $1 + M_{p(i)} \leq 1 + M_j$, i.e. the greedy choice is always at least as small as the optimal solution. **Conclusion**: The optimal solution includes this greedy choice. The algorithm given above chooses to shoot at the y^h from the beginning of the sequence, which is equivalent to shooting at the y^l from the end of the sequence.