Tuesday, April 1, 2025 6:34 AM

Solving Questions with Brainpower

2140. Solving Questions With Brainpower

Solved

Medium Topics Companies Hint

You are given a **0-indexed** 2D integer array `questions` where `questions[i] = [points_i, brainpower_i]`.

The array describes the questions of an exam, where you have to process the questions **in order** (i.e., starting from question `0`) and make a decision whether to **solve** or **skip** each question. Solving question `i` will **earn** you `points_i` points but you will be **unable** to solve each of the next `brainpower_i` questions. If you skip question `i`, you get to make the decision on the next question.

- For example, given `questions = [[3, 2], [4, 3], [4, 4], [2, 5]]`:
  - If question `0` is solved, you will earn `3` points but you will be unable to solve questions `1` and `2`.
  - If instead, question `0` is skipped and question `1` is solved, you will earn `4` points but you will be unable to solve questions `2` and `3`.

Return *the **maximum** points you can earn for the exam.*

Example 1:

```
Input: questions = [[3,2],[4,3],[4,4],[2,5]]
Output: 5
Explanation: The maximum points can be earned by solving questions 0 and 3.
- Solve question 0: Earn 3 points, will be unable to solve the next 2 questions
- Unable to solve questions 1 and 2
- Solve question 3: Earn 2 points
Total points earned: 3 + 2 = 5. There is no other way to earn 5 or more points.
```

Example 2:

```
Input: questions = [[1,1],[2,2],[3,3],[4,4],[5,5]]
Output: 7
Explanation: The maximum points can be earned by solving questions 1 and 4.
- Skip question 0
- Solve question 1: Earn 2 points, will be unable to solve the next 2 questions
- Unable to solve questions 2 and 3
- Solve question 4: Earn 5 points
Total points earned: 2 + 5 = 7. There is no other way to earn 7 or more points.
```

Constraints:

- `1 <= questions.length <= 10^5`
- `questions[i].length == 2`
- `1 <= points_i, brainpower_i <= 10^5`

```
int solve = points[i][0] + help(i + points[i][1])
int notsolve = help(i+1)
return max(solve, notsolve)
```

Top- Down DP with Memoization :-

Code

C++ Auto

```cpp
class Solution {
public:
    long long mostPoints(vector<vector<int>>& questions) {
        int n=questions.size();
        vector<long> dp(n, -1);
        return help(questions, 0, dp);
    }
    long long help(vector<vector<int>> &questions, int i, vector<long> &dp){
        int n=questions.size();
        if(i>=n)return 0;
        if(dp[i]!=-1)return dp[i];
        long solve=questions[i][0]+help(questions, i+questions[i][1]+1, dp);
        long notSolve=help(questions, i+1, dp);
        return dp[i]=max(solve, notSolve);
    }
};
```

T.C. O(n)

S.C. O(n)

Bottom Up DP :-

```cpp
class Solution {
public:
    long long mostPoints(vector<vector<int>>& questions) {
        // Get the number of questions
        int n = questions.size();
// Initialize DP array where dp[i] represents the maximum points
        // we can earn starting from the ith question
        // Using long long to handle potential large values
        vector<long long> dp(n,0);
// Base case: For the last question, we just take its points value
        dp[n-1] = questions[n-1][0];
// Fill the dp array from right to left (bottom-up approach)
        for(int i = n-2 ; i >= 0 ; i--) {
            int points = questions[i][0];
            int brainpower = questions[i][1];

// Option 1: Solve the current question
        // We add the points of current question and the maximum points we can get
        // after skipping the next 'brainpower' questions
        int next_available_index = min(i + brainpower + 1 , n);
        long long solve_points = points + (next_available_index < n ? dp[next_available_index] : 0);
// Option 2: Skip the current question
        // We take the maximum points we can get starting from the next question
        long long skip_points = dp[i+1];
// The optimal strategy is to take the maximum of the two options
        dp[i] = max(solve_points,skip_points);
}
// dp[0] now contains the maximum points possible starting from the first question (with our bottom-up
approach)
```

```
        return dp[0];
    }
};
```

From <https://leetcode.com/problems/solving-questions-with-brainpower/solutions/6601532/best-soluti-on-solving-questions-with-brainpower-a-dynamic-programming-masterclass/?envType=daily-question&envId=2025-04-01>