**Damaging Actions**

Jan2022-Medium-Q1-Damaging Actions <Dynamic Programming> <Nachiket Kanore>

**1. Topic** Dynamic Programming

**2. Difficulty Level :** Medium

**3. Question / Problem Statement**

Ayush has N cards in his hands. All the cards in his hands are action cards. Each of them has a cost (some number of actions you have to spend on using the card) and an effect (the amount of damage it deals to the opponent).

Following is the description of the cards: the arrays: actions and damage.

Each of these arrays has exactly N elements.

For each index i, you have a card that costs actions[i] actions and makes damage[i] damage. You can only play each card once.

You can spend atmost 3 actions on playing your cards this turn.

Find maximum total amount of damage you can perform.

**Note**

N will be between 1 and 50, inclusive.

actions will have exactly N elements.

Each element of actions will be between 1 and 3, inclusive.

damage will have exactly N elements.

Each element of damage will be between 1 and 99, inclusive.

**Function Description**  
In the provided code snippet, implement the provided **damagingActions(...)** method using the variables to print the pattern for each line from 1 to i. You can write your code in the space below the phrase **“WRITE YOUR LOGIC HERE”**.   
  
There will be multiple test cases running so the Input and Output should match exactly as provided.  
The base Output variable **result** is set to a default value of **-404** which can be modified. Additionally, you can add or remove these output variables.

**Input Format**

N

**actions** array consisting N elements  
**damage** array consisting N elements

**Sample Input**

*8*

*{1, 2, 1, 3, 1, 2, 1, 1} – actions*

*{1, 1, 1, 1, 1, 1, 1, 1} – damage*

**Constraints**

1 <= N <= 50

1 <= actions[i] <= 3

1 <= damage[i] <= 99  
  
**Output Format**  
Integer answer  
  
**Sample Output**

3

**Explanation**  
***We can deal three damage by playing any three of the five cards that cost 1 action each.***

**8. Solution Steps:**

[Basic Steps: Pseudo algorithm and Pseudo flow that will allow us to write code in most of the Languages.]

1. Examine all possible sets of one, two, and three cards, for each of them check whether you have enough actions to play it, and if you do, look at the total damage you would deal.
2. Among all those options, the one that deals the most damage is our answer.

**9. Running Solution in C, C++ or Java**

[Running code that will run accurately on our System or Profoundly used IDE.]

#include <bits/stdc++.h>

typedef long long ll;

using namespace std;

int damagingActions(int N, vector<int> actions, vector<int> damage) {

int ans = 0;

for(int i = 0; i < N; i++) {

ans = max(ans, damage[i]);

}

for(int i = 0; i < N; i++) {

for(int j = i+1; j < N; j++) {

if(actions[i] + actions[j] > 3) continue;

ans = max(ans, damage[i] + damage[j]);

}

}

for(int i = 0; i < N; i++) {

for(int j = i+1; j < N; j++) {

for(int k = j+1; k < N; k++) {

if(actions[i] + actions[j] + actions[k] > 3) continue;

ans = max(ans, damage[i] + damage[j] + damage[k]);

}

}

}

return ans;

}

int main() {

int N = 8;

vector<int> actions = {1, 2, 1, 3, 1, 2, 1, 1};

vector<int> damage = {1, 1, 1, 1, 1, 1, 1, 1};

cout << damagingActions(N, actions, damage) << '\n';

}

Input:

2

1 3

2 1

Output:

2

**10. Test Cases [ Qty: 12 ]**

[ Test Cases: Minimum 12 which are composed of Easy, Complex, Negative and Boundary Value Cases (BVC). We should be able to insert these in Our System. Realistic Values ]

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case No** | **Input** | **Output** | **Score** |
| 1 | 3  2 2 1  5 6 1 | 7 | 0 |
| 2 | 3  1 2 3  1 4 9 | 9 | 0 |
| 3 | 8  1 2 1 3 1 2 1 1  1 1 1 1 1 1 1 1 | 3 | 1 |
| 4 | 3  1 2 3  1 7 9 | 9 | 1 |
| 5 | 3  1 2 3  3 7 9 | 10 | 1 |
| 6 | 36  2 3 2 3 2 2 3 2 2 3 1 2 3 1 1 1 2 2 3 3 1 2 3 2 1 1 3 3 2 2 1 1 3 2 3 3  77 44 72 74 98 26 31 94 5 17 43 13 47 75 50 66 70 33 13 84 30 90 43 28 61 96 15 99 54 78 46 87 37 99 31 72 | 258 | 1 |
| 7 | 27  3 3 1 2 2 3 2 2 3 3 1 3 1 3 1 2 2 2 3 3 1 2 1 1 3 3 1  60 60 19 37 41 54 39 37 59 58 20 59 19 59 21 41 37 41 54 58 17 36 18 19 60 59 17 | 62 | 1 |
| 8 | 7  1 1 1 1 3 3 2  14 35 57 1 84 16 54 | 111 | 1 |
| 9 | 3  1 2 1  11 29 16 | 45 | 1 |
| 10 | 50  3 3 1 3 1 1 1 3 2 3 1 3 1 3 1 2 1 1 1 2 1 3 2 1 3 3 3 1 3 3 2 3 3 1 3 2 2 3 1 1 2 2 2 2 3 2 3 1 1 3  96 90 31 96 32 32 34 90 65 96 30 96 33 93 32 59 34 29 30 65 30 91 62 28 94 90 95 31 93 92 64 94 95 33 95 59 63 91 34 33 61 59 65 61 94 62 93 28 33 96 | 102 | 1 |
| 11 | 3  3 3 2  53 99 88 | 99 | 1 |
| 12 | 9  1 3 2 2 2 3 2 1 1  47 7 45 75 8 51 82 35 33 | 129 | 1 |

**Format of Coding Question**

**Nomenclature: Question ID:** Jun2019-Easy-Q1-Multiple Graphs <specimen name>

**1. Topic** and relation of the question to some theory (This is what we will be introducing from now on.)

**2. Difficulty Level :** (Easy, Medium, High) as per the depth of Problem statement and solution difficulty.

**3. Question / Problem Statement**

[ Here the Question should be a short and sweet story that can sufficiently explain problem to be solved. This should not be a clumsy story. It should be simply understandable. Candidate should be able to Identify Topic by reading ]

**Note (if any):**

**4. Function Description:**

[ Here we provide the function name in **camelCase(...).** Here we define what the function will return or print. ]

**5. Input Format:**

[ Here the Candidate will be explained what is acceptable in the Input and how one should accept Input ]

**6. Sample Input:**

[ Here the Candidate will be provided with a running Sample Input ]

**Constraints (if any):**

**7. Output Format:**

[ Here the Candidate will be explained what is need to be as Output and how one should present Output ]

**8. Sample Output:**

[ Here the Candidate will be provided with a running Sample Output ]

**9. Explanation** of Sample Input and Output:

[ Here the Candidate will be guided and explained with how Sample Input and Output has arrived to what is shown. ]

**10. Solution Steps:**

[Basic Steps: Pseudo algorithm and Pseudo flow that will allow us to write code in most of the Languages.]

**11. Running Solution in C, C++ or Java**

[Running code that will run accurately on our System or Profoundly used IDE.]

**12. Test Cases [ Qty: 12 ]**

[ Test Cases: Minimum 12 which are composed of Easy, Complex, Negative and Boundary Value Cases (BVC). We should be able to insert these in Our System. Realistic Values ]

**13. Code stubs (if any):**  
[ Each code stub will have name as per function description ]

**Requirements:**

1. Come up with Better and easier Question Format. Less Story, hinting to closeness of answer. Story should be established with least or no mathematics. Story cannot be more than 2 sentences.

2. Coding Question should become a fun element yet should be able to judge deep serious intent of programming capacity of the Candidate and provide apparatus (How we can better calculate Candidate's Coding and Programming Skills... Use of Data Structures and ability to write optimum solution.)

4. Question should be EEOC and any discriminatory policy compliant:

eg.

a. Cricket is not common globally... Questions cannot be related to unfamiliar subject and topics.

b. Harmful Words: Bomb, Drugs... so on... Are not allowed.

c. Male and female OR Female and Male... Superiority and Inferiority Discrimination not to be a part of this.

5. Inspiration: Questions can be like:

***Easy***

Write a code to find Fibonacci series until 5th number. (It should be as simple as these)

***Situational Difficult Questions:***

Write a Code that can find 5 star rated songs from a set of Playlists which are repeated in each Playlist.

***New and Advanced Difficulty Questions.***

Best topics:

1. Algorithms:

<https://www.geeksforgeeks.org/fundamentals-of-algorithms/>

2. Advanced Data Structures:

[https://www.geeksforgeeks.org/data-structures/#AdvancedDataStructure](https://www.geeksforgeeks.org/data-structures/" \l "AdvancedDataStructure)

3. Latest Trends:

a. Find data from Spatial Matrix.

b. Solvable Analytics.

c. Solvable Big Data.

d. Solvable Smart Algos.

4. Data Science:

<https://www.testdome.com/tests/data-science-test/65>

<https://www.geeksforgeeks.org/tag/data-science/>