ABC 153 解説

writer: kyopro_friends

2020 年 1 月 26 日

For International Readers: English editorial starts on page 7.

A: Serval vs Monster

求める答えは H/A の切り上げです。ceil 関数などを使ってもよいですし、$(H+A-1)/A$ の切り捨てにより求めることもできます。

C 言語での実装例は次のとおりです。

```
#include<stdio.h>
int main(){
    int h,a;
    scanf("%d%d",&h,&a);
    printf("%d\n",(h+a-1)/a);
}
```

このほか、「H が 0 以下になるまで A を引く」というループ処理で解くこともできます。

B: Common raccoon vs Monster

全ての必殺技を 1 度ずつ使ってモンスターの体力を 0 以下にできれば良いです。つまり、$H - (A_1 + A_2 + \ldots + A_N)$ が 0 以下のとき Yes、そうでないとき No となります。

C 言語での実装例は次のとおりです。

```
int main(){
    int H,N;
    scanf("%d%d",&H,&N);
    for(int i=0;i<N;i++){
        int A;
        scanf("%d",&A);
        H-=A;
    }
    if(H<=0)puts("Yes");
    else puts("No");
}
```

C: Fennec vs Monster

体力の高い方から K 体のモンスターに必殺技を使い、残りのモンスターに攻撃するのが最適です。モンスターを体力の順にソートすることで $O(N \log N)$ で解くことができます。

D: Caracal vs Monster

体力が H のモンスター 1 体に勝つために必要な攻撃回数を $f(H)$ とします。このとき

$$f(H) = \begin{cases} 2 \times f(\lfloor H/2 \rfloor) + 1 & (H > 1) \\ 1 & (H = 1) \end{cases}$$

となります。この定義に従って再帰的に計算することで答えを求めることができます。

E: Crested Ibis vs Monster

次のような DP を考えることで解くことができます。

$$DP[i] = \text{モンスターの体力を } i \text{ 減らすため消耗する魔力の最小値}$$

同じ魔法を複数回使うことができるので、DP 配列の更新順序には気をつけてください。答えは $\min_{H \leq i < H + \max A_i} DP[i]$ になります (モンスターの体力を $H + \max A_i$ 以上減らすには、H 以上減らしてからさらに魔法を使う必要があるので無駄です)。この部分は、「$i \geq H$ であるような部分について予めまとめた状態で DP 配列を持つ」という実装の工夫により簡略化することもできます。計算量は $O(NH)$ です。

なお、答えは最大で $H * \max A_i$ 程度になるので、次のような DP では解けないことに注意してください。

$$DP[i] = \text{魔力を } i \text{ 消耗したときに減らせるモンスターの体力の最大値}$$

F: Silver Fox vs Monster

モンスターたちはあらかじめ座標の昇順にソートされているとしてよいです。座標が小さい方を”左”、大きい方を”右”という向きで呼ぶことにします。

最も左にいるモンスターを倒すためときには、できるだけ右で爆弾を使い、他のモンスターを多く巻き込むのが良いです。そのようにして左側から貪欲に倒していくことが最適になります。

尺取り法や二分探索などを用いて、各モンスター i について「$X_j - X_i \leq 2D$ となる最大の j」を計算しておくことで、モンスター i を左端として攻撃するときにどこまで右側のモンスターを巻き込むことができるかわかります。あとは、累積和などを用いて既に減らした体力量を管理することで答えを求めることができます。

最初に座標順にソートする部分がボトルネックとなり、計算量は $O(N \log N)$ です。

A: Serval vs Monster

The desired answer is H/A rounded up. You can use ceil functions etc., or it can be also calculated by rounding down $(H + A - 1)/A$.

The following is a sample code in C language.

```
#include<stdio.h>
int main(){
    int h,a;
    scanf("%d%d",&h,&a);
    printf("%d\n",(h+a-1)/a);
}
```

Otherwise, this problem can also be solved by loop operations of "subtracting by A until H becomes 0 or below."

B: Common raccoon vs Monster

It is enough if the monster's health become 0 or below by using all the Special Move once each. In other words, the answer is Yes if $H - (A_1 + A_2 + \ldots + A_N)$ is 0 or below, and No otherwise.

The following is a sample code in C language.

```
int main(){
    int H,N;
    scanf("%d%d",&H,&N);
    for(int i=0;i<N;i++){
        int A;
        scanf("%d",&A);
        H-=A;
    }
    if(H<=0)puts("Yes");
    else puts("No");
}
```

C: Fennec vs Monster

It is optimal to use Special Move for K monsters with the largest health, and to Attack to the other monsters. By sorting the monsters by the monsters' healths, it can be solved in a total of $O(N \log N)$ time.

D: Caracal vs Monster

Let $f(H)$ be the number of attacks needed to defeat one monster of health H. Then the following relation holds:

$$f(H) = \begin{cases} 2 \times f(\lfloor H/2 \rfloor) + 1 & (H > 1) \\ 1 & (H = 1) \end{cases}$$

The answer can be found by calculating recursively following the definition above.

E: Crested Ibis vs Monster

This problem cane be solved by considering the following DP.

$DP[i]$ = The minimum total Magic Points that have to be consumed to decrease the monster's health by i

Since the same spell can be cast multiple times, be careful of the order of refreshing the DP array. The answer is $\min_{H \leq i < H + \max A_i} DP[i]$ (in order to decrease the monster's health by $H + \max A_i$ or more, an additional spell has to be cast after it is decreased by H or more, so it's meaningless). This part can be simplified by ingenious implementation of "holding a DP array in which the part of $i \geq H$ are put together. The total time complexity is $O(NH)$.

Note that it cannot be solved by the following DP, since the answer can be $H * \max A_i$ at most.

$DP[i]$ = The maximum total health that can be decreased when the i Magic Points are consumed

F: Silver Fox vs Monster

You can assume that the monsters are sorted in the increasing order of coordinates. We denote the direction of smaller coordinates by being "left," and the larger coordinates by being "right."

In order to defeat the leftmost monster, it is good to use the bomb as in the right as possible and get involved as many monsters as possible. It is optimal to defeat greedily from the left in such a way.

By precalculating "the maximum j such that $X_j - X_i \leq 2D$" for each monster i utilizing sliding windows or binary searching etc., it can be determined how many monsters in the right can be involved when monster i is attacked at the left end. All the left is managing the amount of health already decreased, using cumulative sums etc., then the answer can be found.

The bottleneck is the first step, sorting by the coordinates, so the total time complexity is $O(N \log N)$.