План защиты

Автор задачи и разработчик: Мария Жогова

Основные тезисы из условия:

1. можно сражаться против атак с силой не больше m;
2. нельзя сражаться против атак с силой больше m;
3. «пропускать» можно ровно x атак подряд, при этом на момент окончания последней атаки необходимо быть «дома».

В первой подгруппе подойдет любое решение, находящее ответ. В частности, можно написать рекурсивный перебор, который каждый раз рассматривает два варианта: встретить текущую атаку или пропустить следующие x, если это не запрещено условием. Если рекурсивный перебор дойдет до последней атаки, ответ существует, можно его запомнить, чтобы в конце вывести минимальный из всех найденных. Иначе ответа нет. Псевдокод приведен ниже:

```
go(skipped, i):
    if i > n:
        запомнить ответ skipped
        return
    if a[i] <= m:
        go(skipped, i + 1) // не пропускаем
    if i + x - 1 <= n:
        go(skipped + [i], i + x) // пропускаем
```

Заметим, что силы атак не важны, важно только больше они, чем m, или нет. Все атаки, которые имеют силу больше m, необходимо переждать на островах. Таким образом, подгруппа $x = 1$ требует просто вывести номера всех атак, сила которых $> m$.

Остальные группы являются подсказками к полному решению. Так, например, придумав решение для $x = 2$, может быть проще придумать решение для произвольного x. Ограничение $x = 2$ означает, что атаки можно пропускать только по две подряд, в частности, нельзя пропустить последнюю без предпоследней. Если последняя атака имеет $a_n > m$, то ее необходимо пропустить, значит перед $n - 1$-й атакой придется просить убежище. Иначе нет смысла пропускать ее, и можно перейти к рассмотрению предыдущей.

Проблема может возникнуть, когда такой алгоритм дойдет до a_1, и окажется, что ее надо пропустить, а a_2 уже покрыта другим отрезком времени в убежище. В таком случае сделаем следующее:

1. покроем первые две атаки одним отрезком;
2. если вторая была покрыта отрезком $[2, 3]$, то третью атаку надо пропустить — сдвинем отрезок вправо на 1, получим $[3, 4]$;
3. если при этом четвертая была покрыта $[4, 5]$, сдвинем его вправо на 1, и так далее рекурсивно.

Наша рекурсия займет $\mathcal{O}(n)$ времени и либо выставит все отрезки на непересекающиеся позиции, либо окажется, что ни одна атака не была встречена, и Оматикайя провели все время в полетах между островами и обратно, тогда последний отрезок $[n - 1, n]$ нельзя сдвинуть вправо на 1. Но в таком случае массив a имеет вид $[> m, ?, > m, ?, \ldots, ?, > m]$, и все его элементы $> m$ в принципе нельзя покрыть отрезками длины 2.

Это подводит нас к полному решению. Во-первых, можно доказать, что такой жадный алгоритм действительно находит минимальное количество раз, которое клану придется просить убежище. Это стандартная задача. Каждый элемент $> m$ должен быть покрыт каким-то отрезком длины x. Если существует какое-то другое покрытие, можно в нем сдвинуть несовпадающие с нашим покрытием отрезки влево и получить наше покрытие, не увеличив число отрезков.

Собственно, для написания решения формально доказывать этот факт не требовалось, корректность жадного алгоритма в данной задаче кажется достаточно интуитивно понятной, чтобы от нее можно было отталкиваться.

Во-вторых, надо понять, как разбираться со случаями, когда, выбирая самый левый из покрывающих очередную атаку отрезков, мы приходим к тому, что не можем покрыть несколько первых атак. Но на самом деле полностью аналогичное случаю $x = 2$ рассуждение работает и для $x > 2$. Таким образом, псевдокод алгоритма получается довольно простым:

```
for i = n ... 1:
    if a[i] > m и (i не покрыто последним отрезком):
        segments.push_front((i - x + 1, i))
    i -= 1

for t = 0 ... |segments|:
    if segments[t] не пересекает предыдущий: // и левую границу массива
        break
    сдвинуть segments[t] вправо

if segments.back().right > n: // последний отрезок вышел за границу
    вывести -1
else:
    вывести segments
```