Sea2 – Soluţie (Berinde Radu)

Soluţia 1

O soluţie destul de uşor de implementat e una de complexitate **O(NlogN + M * D)**, unde **M** este numărul de vase care nu sunt distruse în momentul apariţiei (pentru care răspunsul nu e **-1**), iar **D** este numărul mediu de vase rămase pe mare. Această complexitate se obţine menţinând un vector de vase (cele rămase pe mare la un moment dat) sortate crescător după **x** (si deci automat descrescător după **y**). Putem afla dacă răspunsul pentru un vas este **-1** căutând binar primul vas cu **x** mai mare şi comparând coordonatele **y**. Când un vas nu este distrus, vedem câte vase trebuie să fie şterse din listă (va fi o subsecvenţă de vase continuă în stânga vasului găsit mai devreme) şi adăugăm noul vas. Soluţia are totuţi complexitatea $\mathbf{O(N^2)}$ pe cazul cel mai defavorabil şi primeşte în jur de 40 de puncte. Însă soluţia poate fi adusă la complexitatea **O(NlogN)** dacă înlocuim vectorul sortat cu un AVL. Pentru programatorii în C++, se puteau folosi arborii echilibraţi deja implementaţi în STL.

Soluţia 2

O soluţie în **O(NsqrtN)** (aprox. 70 de puncte) se poate obţine menţinând două structuri: una pentru a determina pentru fiecare **x**, coordonata **y** maximă pentru care un vas care ar apărea la poziţia **x**,**y** ar fi distrus în momentul apariţiei, iar cealaltă pentru a putea menţine numărul de vase rămase pe mare. Dacă vasul nu este distrus, cu o căutare binară putem afla până la ce **x** vor fi atacate celealte vase şi folosim a doua structură pentru a recalcula numărul de vase rămase. Soluţia se poate aduce la **O(NlogN)** folosind arbori de intervale.

Soluţie cu arbori echilibraţi în C++ (oare e scurtă sau ce?)

```
#include <stdio.h>
#include <set>
#include <utility>
using namespace std;

set< pair<int,int> > Ships;
set< pair<int,int> >::iterator It, It2, t;

int main()
{
    FILE *fi = fopen("sea2.in", "rt");
    FILE *fo = fopen("sea2.out", "wt");
    int x, y, nr;

    for (fscanf(fi, "%d", &nr); nr; nr--)
    {
        fscanf(fi, "%d %d", &x, &y);
        It = Ships.upper_bound(make_pair(x, y));
        if (It != Ships.end() && It->second > y)
        {
            fprintf(fo, "-1\n");
            continue;
        }
        if (Ships.size())
        {
            for (It2 = It; It2 != Ships.begin() && (--(t = It2))->second < y; It2 = t);
            Ships.erase(It2, It);
        }
        Ships.insert(make_pair(x, y));
        fprintf(fo, "%d\n", Ships.size());
    }
    fclose(fo);
    fclose(fi);
    return 0;
}
```