Indian Computing Olympiad

Training Material

Heaps→Race (CEOI 2003)

250,000 rockets racing in parallel tracks across space. Each rocket i has a starting position x_i and a constant velocity v_i (bounded by 100) with which it will move. Assume that the rockets are enumerated in order of their starting position.

Thus, initially the rockets are lined up as:

```
-----x---------------------------------------------------------->
   (x1,v1)

---------x---------------------------------------------------->
       (x2,v2)

----------------x---------------------------------------------->
             (x3,v3)

....

---------------------------x---------------------------------->
                  (x250000,v250000)
```

Constraints

- At any given time, no more two rockets are at the same position. This implies that all overtakings take place at distinct times.
- The race goes on forever. Eventually, faster rockets overtake slower rockets and beyond a point the rockets will become ordered in the order of their velocities.

Problems

1. Compute the number of overtakings (modulo 1000000, say) in the eventual race.
2. Print out the first 10,000 overtakings (in order).

Solution

Part 1

For starting position x_i, the rocket at that position will eventually overtake every rocket starting at x_j> x_i with velocity v_j<v_i. For each each velocity v_j maintain an array A_j of size 250000 which records for each position j the number of rockets of velocity v_i from position j to 250000 in the initial order. Then, to compute the number of rockets overtaken by the rocket (x_i,v_i), just add up A_j[i] for each v_j<v_i.

Naively, this takes time N^2. We can do it more efficiently as follows.

Maintain an array AtV[1..100] that records for each velocity v, the number of rockets AtV[v] that we have seen at velocity v.

Initialize AtV[1..100] to 0. Start from the last rocket. For rocket ri with position xi and velocity vi, increment AtV[vi]. Rocket ri will overtake all rockets to its right with smaller velocity. Inductively, AtV[] currently has the count all the velocities for all rockets to right of ri. So, add up AtV[1] + AtV[2] + ... + AtV[vi-1] to get number of overtakings that ri performs.

In this way, in one scan, we can add up the total number of overtakings. We scan N rockets and each step requires scanning the array AtV[] of size V, so this takes time N*V.

Another approach is to observe how the order of rockets changes. The final order is sorted in descending order terms of velocity. Each rocket r_i has a rank F_i in the final order and an initial rank S_i in the starting order.

How do we find the rank F_i for a rocket r_i? We have to sort the velocities, but this should be a stable sort so that rockets with same velocity retain their order. What we need to do is to count the number of swaps that between the original order and the stably sorted order.

We can use divide and conquer to get an O(N log N) solution to the number of swaps. Break up the total list into two halves, recursively compute the number of overtakings in each half and then compute the number of overtakings from left half to right half.

(This is essentially merge sort.)

Part 2

From the initial configuration, we can compute for each rocket the identity of the next rocket it will overtake.

Observe that a rocket at (x_i,v_i) can overtake a rocket at (x_j,v_j) only if x_j>x_i and v_j<v_i. In particular, among all rockets of this form, there is exactly one that it will overtake first. At any point, let order of the rockets be 1,2,..,N. Then the next overtaking must be an adjacent pair (i,i+1). (Why?)

Consider all pairs (r_i,r_i+1) such that the next rocket that r_i can overtake is r_i+1. For each such pair, there is a time t_i at which the overtaking will occur. It is possible that r_i will never overtake r_i+1 in which case we set t_i to infinity.

The next overtaking that will take place is the pair (r_i,r_i+1) with the shortest time t_i. We can thus maintain the pairs in a heap in terms of the time t_i at which the overtaking takes place.

At each step, we obtain the next overtaking by deleting the minimum element from the heap. This interchanges the pair (i,i+1) in the current order of the rockets and renumbers i+1 as i' and i as i'+1. We then compute the times for (i-1,i') and (i',i+2) and add the two new elements to the heap.

What about the old entries for (i-1,i) and (i+1,i+2) which no longer reflect consecutive entries in the list? Should these entries be deleted? Deleting arbitrary elements from a heap is not easy. However, we do not really need to delete these entries. They are valid overtakings and will take place at the time appointed.

The problem that may occur is that we make duplicate entries. For instance, suppose after the initial (i,i+1) overtaking, the next overtaking is (i-1,i') (where i' is the old i+1). Now the old (i-1,i) is again adjacent and we will introduce a duplicate entry for this pair in the heap.

However, notice that we are promised that all overtaking times are distinct. So, when we extract the minimum element from the heap, if it is the same as the previous element we got, we can be sure it is a duplicate and discard it.

Requires 250000 log 250000 operations to set up the heap and 10000*3*log 250000 operations to print out first 10000 entries.
