Day 98 coding Statement :

There are N breakfasts in the restaurant "Fat Hut" where the Arun works. The ith breakfast has an attractiveness A_i and cost C_i.

Arun has noticed that nobody takes the jth breakfast if there exists at least one breakfast i such that $A_i \geq A_j$ and $C_i < C_j$.

In other words, if a breakfast is less attractive and more expensive than any of the other dishes, then nobody is
interested in that breakfast.

Arun will be happy if all the N breakfasts have a chance to be taken. Unfortunately, Arun has no power over prices.
On the other hand, he can change the attractiveness of some breakfasts by some real number. However, after the changes,
the attractiveness of the ith breakfast must lie in the interval $[L_i, R_i]$.

He would also like to change the attractiveness of the minimum number of breakfasts. Help the Chef do it.

```java
import java.util.*;
class Main {

public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
try {
int t=0 ;
if (sc.hasNextInt())
t= sc.nextInt();
while(t-->0){
int n = sc.nextInt();
D[] dishes = new D[n];
for (int i = 0; i < n; i++) {
long a= sc.nextLong();
long c= sc.nextLong();
long l= sc.nextLong();
long r= sc.nextLong();
dishes[i]=new D(a,c,l,r);
}
int res = solution(dishes, n);
System.out.println(res);
}
}catch(Exception e){
 e.printStackTrace();
 return;

}
```

```java
        finally {
        sc.close();
        }
        }
         private static int solution(D[] dishes, int n) {
        Arrays.sort(dishes,0,n,(a,b) -> Long.compare(a.cost, b.cost));
        TreeSet<Long> v = new TreeSet<Long>();
         long max = 0;
         boolean fail = false;
         for(int i = 0; i < n; ++i){
         long l=dishes[i].l;
         long r=dishes[i].r;
         long a=dishes[i].attr;
         if(r <= max){
         fail = true;
         break;
         }
         while(!v.isEmpty() && v.last() >= r)
         v.remove(v.last());
         if(a > max){
         if(v.isEmpty() || a> v.last())
         v.add(a);
         else {
         long pos = v.ceiling(a);
         v.remove(pos);
         v.add(a);

         }
         }
         max = Math.max(max, l);
         }
         if(!fail)
         return n - v.size();
        return -1;
        }
        }
        class D {
        long attr;
        long cost;
        long l;
        long r;
        public D(long attr, long cost, long lattr, long rattr) {
        this.attr = attr;
        this.cost = cost;
        this.l = lattr;
        this.r = rattr;
        }
        }
```