FROG SORT

PROBLEM CODE: FROGS

PROBLEM DESRIPTION

There are N frogs (numbered 1 through N) in a line. For each valid i, the i-th frog is initially at the position i, it has weight Wi, and whenever you hit its back, it jumps a distance Li to the right, i.e. its position increases by Li. The weights of the frogs are pairwise distinct.

You can hit the back of each frog any number of times (possibly zero, not necessarily the same for all frogs) in any order. The frogs do not intefere with each other, so there can be any number of frogs at the same time at each position.

Your task is to sort the frogs in the increasing order of weight using the smallest possible number of hits. In other words, after all the hits are performed, then for each pair of frogs (i,j) such that Wi<Wj, the position of the i-th frog should be strictly smaller than the position of the j-th frog. Find the smallest number of hits needed to achieve such a state.

INPUT

The first line of the input contains a single integer T denoting the number of test cases. The description of T test cases follows.
The first line of each test case contains a single integer N.
The second line contains N space-separated integers W1,W2,…,WN.
The third line contains N space-separated integers L1,L2,…,LN.

OUTPUT

For each test case, print a single line containing one integer — the smallest number of times you need to hit the backs of the frogs.

CONSTRAINTS

- $1 \le T \le 2 \cdot 10_4$
- $2 \le N \le 4$
- $1 \le W_i \le N$ for each valid i
- $1 \le L_i \le$ for each valid i
- no two frogs have the same weight

WEIGHTS	3	1	2
L	1	4	5

WEIGHTS	3	1	2
L	1	4	5

	1, 3	2
	4 1	5

WEIGHTS	3	1	2
L	1	4	5

	1, 3	2
	4 1	5

	1	2, 3
	4	5 1

WEIGHTS
L

3	1	2
1	4	5

	1, 3	2
	4 1	5

	1	2, 3
	4	5 1

	1	2	3
	4	5	1

WEIGHTS	3	1	2
L	1	4	5

	1, 3	2
	4 1	5

STEP 1

	1	2, 3
	4	5 1

STEP 2

	1	2	3
	4	5	1

STEP 3

Basically we have to start from Frogs to with lowest weight and put them in their suitable position.

It’s a GREEDY STARTERGY.

Basically we have to start from Frogs to with lowest weight and put them in their suitable position.

It’s a GREEDY STARTERGY.

Maintain a sorted array of weights of the frog. Assume the frog with lowest weight is at correct position.

And kick the frog next to it in terms of weight to right of its previous frog. And continue till the frog with largest weight.

Basically we have to start from Frogs to with lowest weight and put them in their suitable position.

It's a GREEDY STARTERGY.

Maintain a sorted array of weights of the frog. Assume the frog with lowest weight is at correct position.

And kick the frog next to it in terms of weight to right of its previous frog. And continue till the frog with largest weight.

2	**1**	**4**	**3**
4	**1**	**2**	**4**
0	1	2	3

Basically we have to start from Frogs to with lowest weight and put them in their suitable position.

It's a GREEDY STARTERGY.

Maintain a sorted array of weights of the frog. Assume the frog with lowest weight is at correct position.

And kick the frog next to it in terms of weight to right of its previous frog. And continue till the frog with largest weight.

2	**1**	**4**	**3**
4	**1**	**2**	**4**
0	1	2	3

2	1	4	3
4	1	2	4

0 1 2 3

	1	4	3	2
	1	2	4	4

2	1	4	3
4	1	2	4

0	1	2	3	4
	1	4	3	2
	1	2	4	4

					5	6	7
	1	4		2			3
	1	2		4			4

2	1	4	3
4	1	2	4

0	1	2	3	4
	1	4	3	2
	1	2	4	4

					5	6	7
	1	4		2			3
	1	2		4			4

	1			2, 4			3
	1			4 2			4

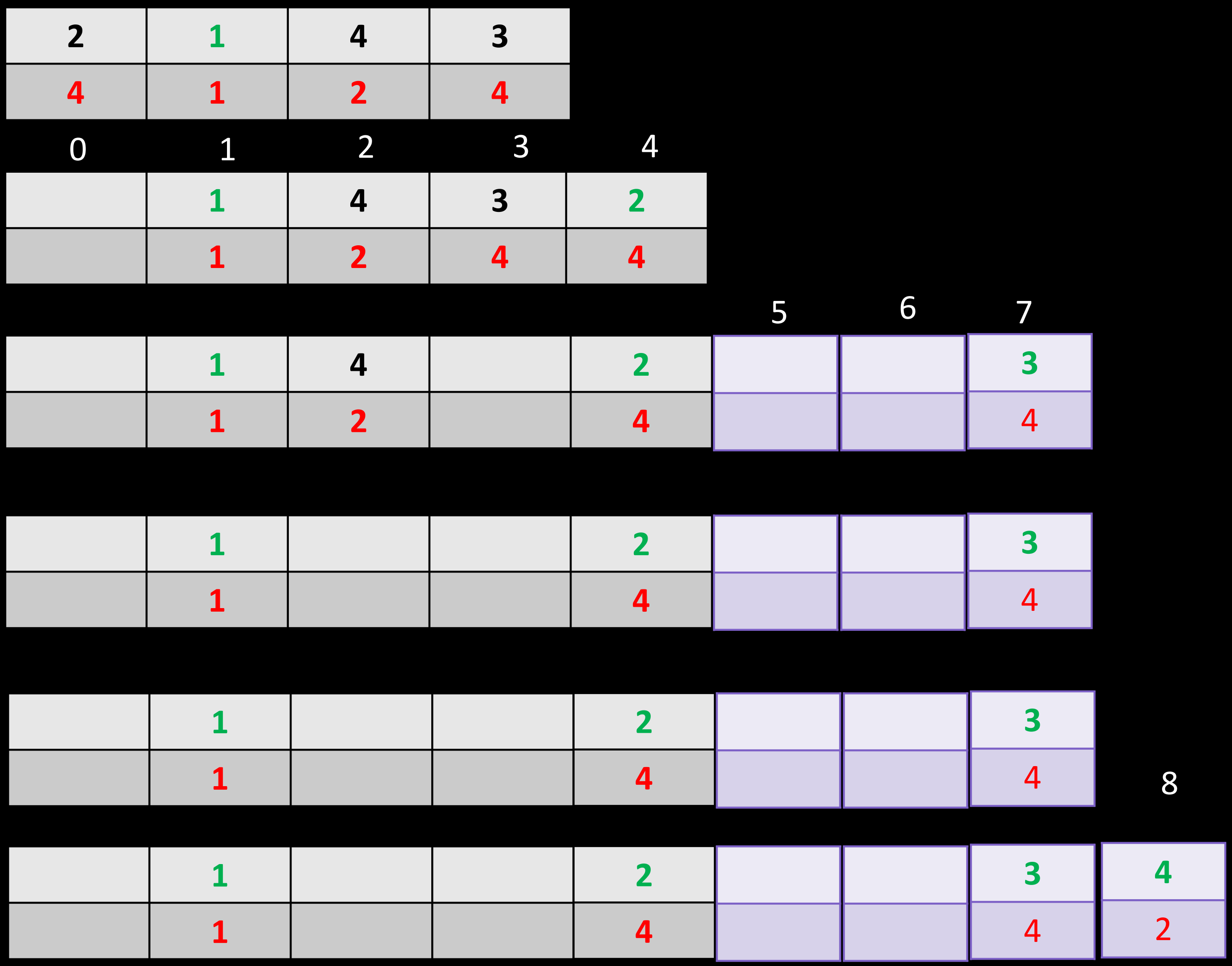
2	1	4	3
4	1	2	4

0	1	2	3	4
	1	4	3	2
	1	2	4	4

					5	6	7
	1	4		2			7
	1	2		4			4

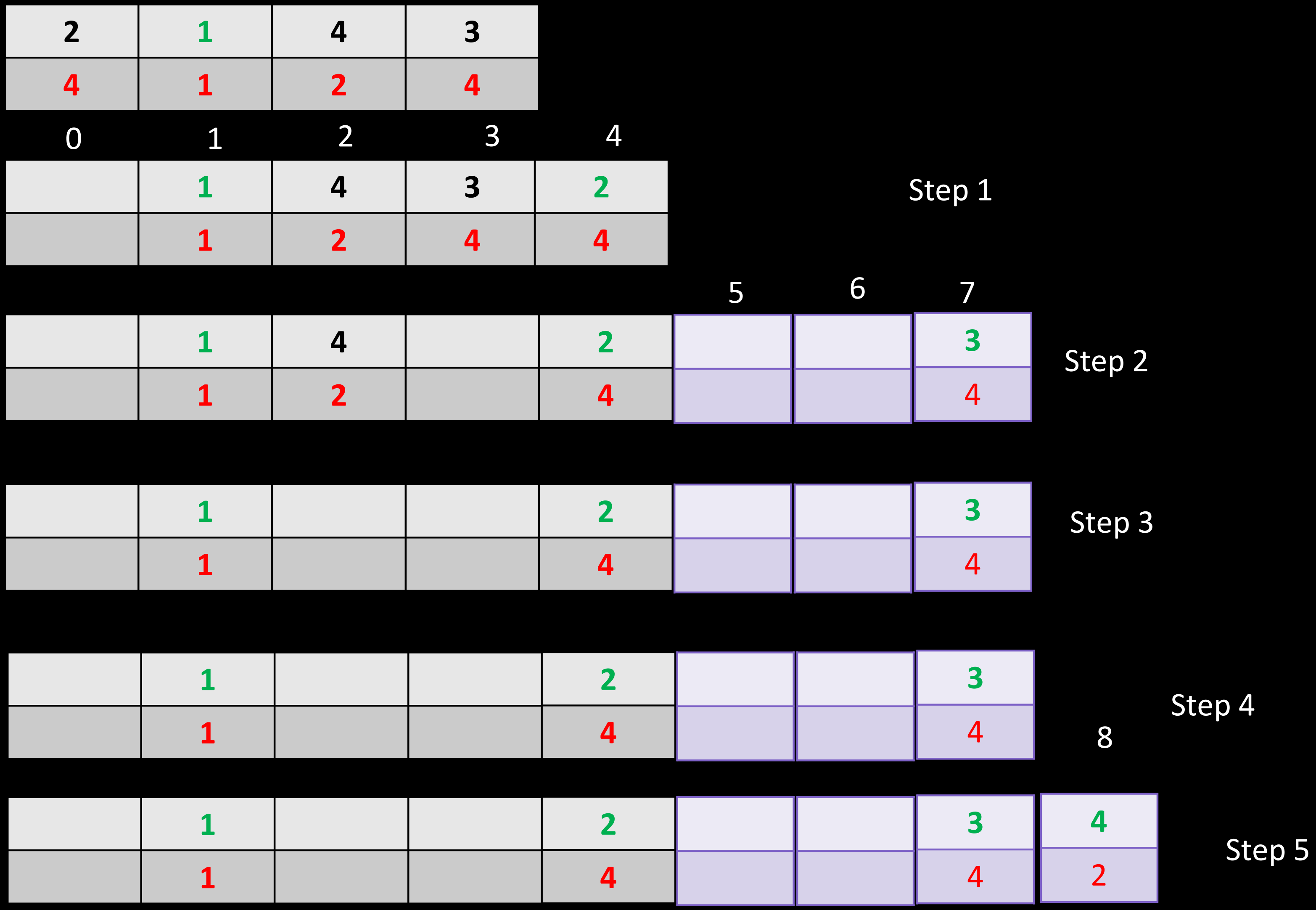
	1			2, 4			7
	1			4 2			4

	1			2		4	7
	1			4		2	4


2 1 4 3
4 1 2 4
0 1 2 3 4
1 4 3 2
1 2 4 4
5 6 7
1 4 2 3
1 2 4 4
1 2 3
1 4 4
1 2 3
1 4 4
8
1 2 3 4
1 4 4 2

Output:

5



2 1 4 3
4 1 2 4
0 1 2 3 4
1 4 3 2
1 2 4 4
Step 1
5 6 7
1 4 2 3
1 2 4 4
Step 2
1 2 3
1 4 4
Step 3
1 2 3
1 4 4
8
Step 4
1 2 3 4
1 4 4 2
Step 5

```
vector<int> w;       // containing weights
vector<int> l;        // contains the value of l

vector<int> pos(n); // pos[i] = i, 0<=i <= n-1, will store the
                      //current index of each frog

vector<int> b(n);
sort(b.begin(), b.end());    // to get a sorted version of w

        for (int j = 1; j< n; j++)
        {
            int index = getIndex(w, b[j]);
            int p = pos[getIndex(w, b[j-1])];
            int c = index;

            while(c<= p)
            {
                c += l[index];
                pos[index] = c;
                count ++;
            }
        }
Return count;
```

THANKS FOR WATCHING !