Competitive Programming
Science Honors Program (SHP)
Session 5

Christian Yongwhan Lim

Saturday, October 31, 2026

Guest Today!

Carl Zhang

- Founding Engineer at Probook
- Judge, 2026 ICPC North America Mid-Central Regional

Overview

- Dynamic Programming
- Divide and Conquer

0-1 Knapsack

Divide and Conquer

Divide and Conquer

- **Divide** the problem into multiple subproblems,

Divide and Conquer

- **Divide** the problem into multiple subproblems,

- **Conquer** each subproblem,

Divide and Conquer

- **Divide** the problem into multiple subproblems,
- **Conquer** each subproblem,
- **Combine** the result!

Merge Sort

- Sort an array of integers in ascending order.

Binary Exponentiation

- Calculate $\mathbf{a^n}$ fast!

Binary Exponentiation: Main Idea

$$a^n = \begin{cases} 1 & \text{if } n == 0 \\ \left(a^{\frac{n}{2}}\right)^2 & \text{if } n > 0 \text{ and } n \text{ even} \\ \left(a^{\frac{n-1}{2}}\right)^2 \cdot a & \text{if } n > 0 \text{ and } n \text{ odd} \end{cases}$$

Binary Exponentiation: Implementation

```
long long binpow(long long a, long long b) {
    if (b == 0)
        return 1;
    long long res = binpow(a, b / 2);
    res *= res
    if (b % 2)
        return res * res * a; // res * a
    else
        return res * res; // res
}
```

Merge Sort

- Sort an array of integers in ascending order.

- Time: O(n log n)
- Space: O(n)

Merge Sort: Implementation

```
// vector<int> v, aux;
void merge_sort(int left, int right) {
  if (left==right) return;
  int middle=(left+right)/2;
  merge_sort(left,middle), merge_sort(middle+1,right);
```

- Time: O(n log n)
- Space: O(n)

Merge Sort: Implementation

```
// vector<int> v, aux;
void merge_sort(int left, int right) {
  if (left==right) return;
  int middle=(left+right)/2;
  merge_sort(left,middle), merge_sort(middle+1,right);
  aux.clear();
  int i=left, j=middle+1;
  while (i<=middle || j<=right)
    if (j>right || (i<=middle && v[i]<v[j]))
      aux.push_back(v[i]), i++;
    else aux.push_back(v[j]), j++;
  for (int i=left; i<=right; i++)
    v[i]=aux[i-left];
}
```

Permutation Transformation (1490D: Div. 3 D)

D. Permutation Transformation

time limit per test: 2 seconds

memory limit per test: 256 megabytes

input: standard input

output: standard output

Permutation Transformation (1490D: Div. 3 D)

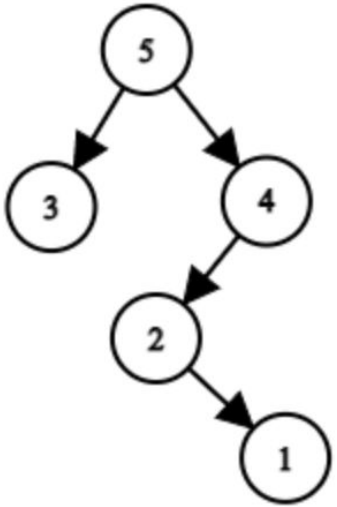
A permutation — is a sequence of length n integers from 1 to n, in which all the numbers occur exactly once. For example, $[1]$, $[3, 5, 2, 1, 4]$, $[1, 3, 2]$ — permutations, and $[2, 3, 2]$, $[4, 3, 1]$, $[0]$ — no.

Polycarp was recently gifted a permutation $a[1 \ldots n]$ of length n. Polycarp likes trees more than permutations, so he wants to transform permutation a into a rooted binary tree. He transforms an array of different integers into a tree as follows:

- the maximum element of the array becomes the root of the tree;
- all elements to the left of the maximum — form a left subtree (which is built according to the same rules but applied to the left part of the array), but if there are no elements to the left of the maximum, then the root has no left child;
- all elements to the right of the maximum — form a right subtree (which is built according to the same rules but applied to the right side of the array), but if there are no elements to the right of the maximum, then the root has no right child.

Permutation Transformation (1490D: Div. 3 D)

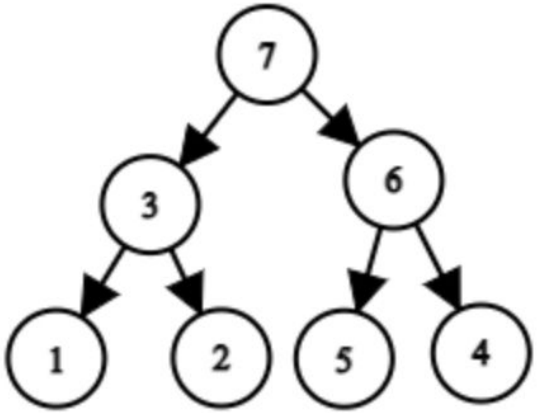
For example, if he builds a tree by permutation $a = [3, 5, 2, 1, 4]$, then the root will be the element $a_2 = 5$, and the left subtree will be the tree that will be built for the subarray $a[1 \dots 1] = [3]$, and the right one — for the subarray $a[3 \dots 5] = [2, 1, 4]$. As a result, the following tree will be built:



The tree corresponding to the permutation $a = [3, 5, 2, 1, 4]$.

Permutation Transformation (1490D: Div. 3 D)

Another example: let the permutation be $a = [1, 3, 2, 7, 5, 6, 4]$. In this case, the tree looks like this:



The tree corresponding to the permutation $a = [1, 3, 2, 7, 5, 6, 4]$.

Let us denote by d_v the depth of the vertex a_v, that is, the number of edges on the path from the root to the vertex numbered a_v. Note that the root depth is zero. Given the permutation a, for each vertex, find the value of d_v.

Permutation Transformation (1490D: Div. 3 D)

Input

The first line contains one integer t ($1 \le t \le 100$) — the number of test cases. Then t test cases follow.

The first line of each test case contains an integer n ($1 \le n \le 100$) — the length of the permutation.

This is followed by n numbers $a_1, a_2, \ldots, a_n$ — permutation a.

Output

For each test case, output n values — $d_1, d_2, \ldots, d_n$.

Permutation Transformation (1490D: Div. 3 D)

Example

input

Copy

```
3
5
3 5 2 1 4
1
1
4
4 3 1 2
```

output

Copy

```
1 0 2 3 1
0
0 1 3 2
```

Permutation Transformation (1490D: Div. 3 D): header

```
#include <bits/stdc++.h>
using namespace std;

#include <ext/pb_ds/assoc_container.hpp>
using namespace __gnu_pbds;
template <class T>
using Tree = tree<T, null_type, less<T>,
rb_tree_tag, tree_order_statistics_node_update>;

typedef long long ll;
```

Permutation Transformation (1490D: Div. 3 D): helper

```
vector<int> a,d;

void eval(int l, int r, int x) {
  int mx=-1, cur=-1;
  for (int i=l; i<=r; i++)
    if(mx<a[i]) mx=a[i], cur=i;
  d[cur]=x;
  if(cur>l) eval(l,cur-1,x+1);
  if(cur<r) eval(cur+1,r,x+1);
}
```

Permutation Transformation (1490D: Div. 3 D): main

```
int main() {
  ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);
  int t; cin>>t;
  while(t--) {
    int n; cin>>n;
    a.assign(n,-1), d.assign(n,-1);
    for (int i=0; i<n; i++) cin>>a[i];
    eval(0,n-1,0);
    for (auto &e : d) cout<<e<<" ";
    cout<<"\n";
  }
  return 0;
}
```


BREAK

Group Exercise

- You may work in groups with those sitting near you.
- Each person must submit their own solutions.
- You have **75 minutes**.
- Afterward, you will present your solution!


THANK YOU