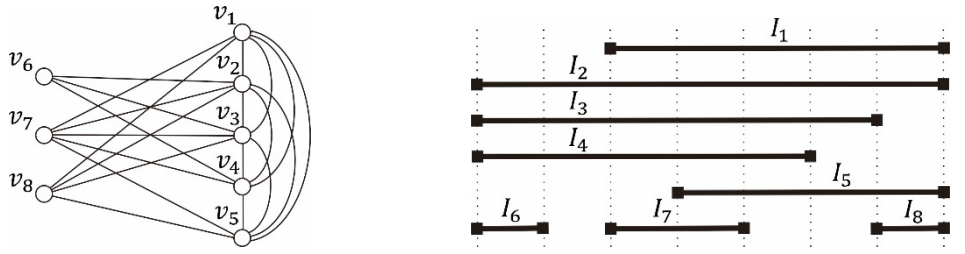
# Problem E
## Network Vulnerability
Time Limit: 3 Seconds

Network vulnerability refers to how much the network performance reduces in various cases of disruptions, such as natural disasters, element failures, or adversarial attacks. A network is frequently represented as a graph, in which the vertices and edges correspond to nodes and links, respectively. So the terms network and graph are used interchangeably here. The connectivity, which is defined to be the minimum number of vertices whose deletion results in a disconnected graph or a one-vertex graph, is recognized as the most important measure to evaluate the vulnerability of a network.

Nonetheless, the connectivity only partly reflects the resistance of a graph to the deletion of vertices. Accordingly, there have been many studies proposing other metrics to account for the network vulnerability, among which the toughness and the scattering number appear to be the most popular. The underlying notion of the toughness and scattering number is the maximum number of connected components resulting from removing $k$ vertices from a graph. Let $c_k(G)$ denote the maximum number of connected components obtained by removing exactly $k$ vertices from a graph $G$. It is unfortunate that given a graph $G$ and a positive integer $k$, the problem of determining $c_k(G)$ is computationally intractable.

For some graph classes like interval graphs, however, $c_k(G)$ can be computed in polynomial time. An interval graph is the intersection graph of a family of (closed) intervals on the real line, where two vertices are connected with an edge if and only if their corresponding intervals intersect. The family is usually called an interval representation for the graph. See Figure E.1 for an example of an interval graph and its interval representation.




(a) an interval graph (b) an interval representation of the graph of (a)

Figure E.1: An interval graph and its interval representation.

Given an interval representation of an interval graph $G$ with $n$ vertices, your job is to write an efficient running program for computing $c_k(G)$ for all $k$ included in $\{0, 1, \ldots, n-1\}$. If the interval representation shown in Figure E.1(b), for example, is given, then $c_0(G)$, $c_1(G)$, $c_2(G)$, $c_3(G)$, $c_4(G)$, $c_5(G)$, $c_6(G)$, $c_7(G)$ are 1, 1, 1, 2, 2, 3, 2, 1, respectively.

### Input
Your program is to read from standard input. The first line contains a positive integer $n$ representing the number of intervals, where $n \leq 2{,}000$. In the following $n$ lines, each contains a pair of left and right endpoints of an interval. You may assume that the endpoints, left or right, of intervals are integers between $-100{,}000{,}000$ and $100{,}000{,}000$, inclusive.

**Output**

Your program is to write to standard output. Print exactly one line which contains the sequence $c_0(G)$, $c_1(G)$, ..., $c_{n-1}(G)$ of $n$ numbers separated by a single space.

The following shows sample input and output for four test cases.

**Sample Input 1**

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

**Output for the Sample Input 1**

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

**Sample Input 2**

```
3
-2 -2
-2 7
-2 7
```

**Output for the Sample Input 2**

```
1 1 1
```

**Sample Input 3**

```
4
-1 -1
3 4
5 6
7 8
```

**Output for the Sample Input 3**

```
4 3 2 1
```

**Sample Input 4**

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

**Output for the Sample Input 4**

```
2 3 3 2 1
```