# Numbering (numbering)

Given a forest on $N$ nodes, a numbering of it is an assigment of positive integers to each edge of the forest. A numbering is beautiful if, for every node, its edges have the numbers $1, 2, \ldots, d$ in some order (where $d$ is the degree of the node).

You are given $N$ positive integers $A_0, \ldots, A_{N-1}$. Determine if there exists a forest on $N$ nodes such that:

- for every $0 \leq i \leq N-1$, the degree of the node $i$ is $A_i$;
- it admits at least one beautiful numbering.

Additionally, if there exists a such a forest, construct an example.

## Implementation

You will have to submit a single `.cpp` source file.

> ☞ Among this task's attachments you will find a template `numbering.cpp` with a sample implementation.

You have to implement the following function:

C++
```
variant<bool, vector<pair<int, int>>> find_numbering(int N, vector<int> A);
```

- Integer $N$ represents the number of nodes.
- The array $A$, indexed from 0 to $N-1$, contains the values $A_0$, $A_1$, ..., $A_{N-1}$, where $A_i$ is the degree of the $i$-th node.
- The function should return either a boolean or an array of pairs of integers.
  - If no valid (satisfying the conditions of the statement) forest exists, the function should return `false`.
  - If a valid forest exists, you have two options:
    - To be awarded the full score, the procedure should return an array of pairs of integers, representing the edges of a valid forest.
    - To be awarded a partial score, the procedure should return `true` or any array of integers not describing a valid forest.

The grader will call the function `find_numbering` and will print the following to the output file:

- If the return value is `false`, it will print a single line with the string `NO`.
- If the return value is `true`, it will print a single line with the string `YES`.
- If the return value is an array of pairs of integers of length $M$, it will print a line with the string `YES`, followed by one line with $M$, followed by $M$ lines with the pairs of the array.

## Sample Grader

The task's directory contains a simplified version of the jury grader, which you can use to test your solution locally. The simplified grader reads the input data from `stdin`, calls the functions that you must implement, and finally writes the output to `stdout`.

The input is made up of 2 lines, containing:

- Line 1: the integer $N$.
- Line 2: $A_0, A_1, \ldots, A_{N-1}$.

The output is made up of multiple lines, containing the values returned by the function `find_numbering`.

## Constraints

- $2 \leq N \leq 10^5$.
- $0 \leq A_i \leq N - 1$.

## Scoring

Your program will be tested on a set of test cases grouped by subtask. The score associated to a subtask will be the minimum of the scores obtained in each of the test cases.

- **Subtask 1 [ 0 points]**: Sample test cases.
- **Subtask 2 [16 points]**: $A_i \leq 2$.
- **Subtask 3 [12 points]**: $A_i \leq 3$.
- **Subtask 4 [16 points]**: Let $\texttt{count}(i)$ be the number of occurrences of $i$ in $A$. You are guaranteed that $\texttt{count}(i) \geq \texttt{count}(i+1) + \texttt{count}(i+2) + \ldots$ for all $1 \leq i \leq N - 1$.
- **Subtask 5 [10 points]**: $N \leq 12$.
- **Subtask 6 [24 points]**: $N \leq 500$.
- **Subtask 7 [22 points]**: No additional constraints.

For each test case in which a valid forest exists, your solution:

- gets full points if it returns a valid forest.
- gets 50% of the points if it returns `true` or an array that does not describe a valid forest.
- gets 0 points otherwise.

For each test case in which a valid forest does not exist, your solution:

- gets full points if it returns `false`.
- gets 0 points otherwise.

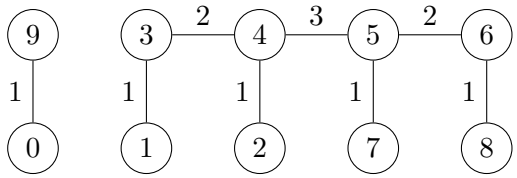
## Examples

| stdin | stdout |
|---|---|
| 4<br>1 1 2 1 | NO |
| 10<br>1 1 1 2 3 3 2 1 1 1 | YES<br>8<br>0 9<br>1 3<br>2 4<br>3 4<br>4 5<br>5 6<br>5 7<br>6 8 |

## Explanation

In the **first sample case**, we want a valid forest with 4 nodes: 3 with degree 1 and 1 with degree 2. We can show that this is not possible. Suppose such a forest exists, then there should be an edge with number 2 out of the node with degree 2. This edge connects to another node that should have degree at least 2. However, such a node does not exist, since all other nodes have degree 1.

In the **second sample case**, we want a valid forest with 10 nodes: 6 with degree 1, 2 with degree 2 and 2 with degree 3. Such a forest exists and the output is depicted below:




Notice that nodes 4 and 5 have three edges labeled $1, 2$ and 3.
Furthermore nodes 3 and 6 have two edges labeled 1 and 2.
Finally nodes $0, 1, 2, 7, 8$ and 9 have one edge labeled 1.