4 Rebuilding Bunnyland

4.1 Problem Statement

Wabbit thinks that Supreme Leader Armpit has ruined Bunnyland. Therefore, he has decided to rebuild Bunnyland as a safe refuge for his people.

Wabbit remembers that Bunnyland has N cities, numbered from 0 to $N-1$, and that there are D_i roads connected to the city i. He remembers that there are no self-loops or multiple edges, but is unsure whether you can move between any pair of cities using only the roads (the graph is connected). Hence, sometimes he will ask for a connected Bunnyland ($K = 1$), but sometimes he will not ($K = 0$).

Wabbit knows that it is impossible to rebuild Bunnyland exactly using only the information he has. However, he will be happy as long as his new Bunnyland satisfies all the constraints he remembers. Help him rebuild Bunnyland!

Formally, construct any graph with N nodes such that node i has degree D_i. **There should exist no self-loops or duplicate edges.** If $K = 1$, the graph should also be connected, else the connectivity doesn't matter.

4.2 Implementation Details

Your code should contain `#include "rebuild.h"`. You should not read from standard input or write to standard output. You should implement the following procedure:

```
bool rebuild(int N, vector<int> D, bool K)
```

- N: The number of cities in Bunnyland
- D: An array of length N representing the number of roads connected to city i.
- K: A boolean indicating if the graph constructed must be connected
- The procedure should return a boolean, indicating whether a reconstructed Bunnyland exists satisfying the constraints.
- The procedure will be called T times in each run.

If your function returns `true`, you are expected to construct one example of a graph satisfying the constraints. You may do so by calling the following procedure in `rebuild`:

```
void build_road(int X, int Y)
```

- X and Y are integers indicating that there is a road connecting nodes X and Y. They should satisfy $0 \leq X, Y < N$ and $X \neq Y$, and all calls to the function should be made with distinct unordered pair (X, Y).

4.3 Subtasks

For all calls to `rebuild`, it is guaranteed that:

- $1 \leq N$
- $0 \leq D_i < N$
- Sum of D_i (let's call it M) is even

For all runs of your program, it is guaranteed that:

- Sum of N is at most 100000 across all calls to `rebuild`
- Sum of M is at most 100000 across all calls to `rebuild`

Subtask	Score	N	K	Additional constraints
0	0	Sample Testcases		
1	13	$N \leq 4$	-	-
2	26	-	-	There is a construction $K_N \backslash T$
3	9	sum of N^2 is at most 200000	-	-
4	26	-	$K = 0$	-
5	26	-	$K = 1$	-

If you do not understand constraint for subtask 2, please read the footnote[1].

4.4 Sample Grader

A sample grader is provided as attachment. It takes in input in the following format:

- line 1: T
- line $2i$: N K
- line $2i + 1$: $D_1, D_2, \cdots, D_N$

The sample grader will not evaluate the correctness of your output. It will print out "Yes" for each test case that you provide a construction and "No" otherwise. If you provide a construction with M calls to `build_road`, it will then print M lines containing the edges you have added into the graph.

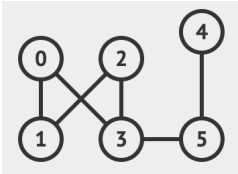
[1]The construction contains all edges in the complete graph with N nodes, except a set of $N - 1$ edges which forms a tree with N nodes. Note that it is not guaranteed that such a graph is connected.

4.5 Sample Testcases

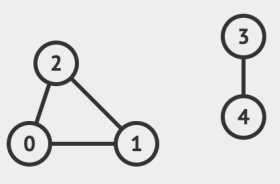
standard input	standard output
4 6 1 2 2 2 3 1 2 5 0 2 2 2 1 1 6 1 1 1 2 1 2 1 6 0 1 1 4 1 4 1	Yes 0 1 1 2 0 3 2 3 3 5 5 4 Yes 0 1 1 2 2 0 3 4 No No

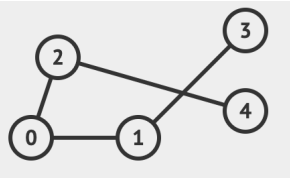
4.6 Sample Testcase Explanation

This is one possible testcase 1 solution (there may be others):



These are testcase 2 solutions (note that either one is fine as we did not ask for connected, and there may be other solutions not listed here):





This is not a testcase 3 solution because it is not connected. Testcase 3 has no solutions.

Testcase 4 has no solutions, connected or unconnected.