D. Graph Game

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

In computer science, there is a method called "Divide And Conquer By Node" to solve some hard problems about paths on a tree. Let's desribe how this method works by function:

$solve(t)$ (t is a tree):

1. Chose a node x (it's common to chose weight-center) in tree t. Let's call this step "Line A".
2. Deal with all paths that pass x.
3. Then delete x from tree t.
4. After that t becomes some subtrees.
5. Apply $solve$ on each subtree.

This ends when t has only one node because after deleting it, there's nothing.

Now, WJMZBMR has mistakenly believed that it's ok to chose any node in "Line A". So he'll chose a node at random. To make the situation worse, he thinks a "tree" should have the same number of edges and nodes! So this procedure becomes like that.

Let's define the variable $totalCost$. Initially the value of $totalCost$ equal to 0. So, $solve(t)$ (now t is a graph):

1. $totalCost = totalCost + (size\ of\ t)$. The operation "=" means assignment. $(Size\ of\ t)$ means the number of nodes in t.
2. Choose a node x in graph t at random (uniformly among all nodes of t).
3. Then delete x from graph t.
4. After that t becomes some connected components.
5. Apply $solve$ on each component.

He'll apply $solve$ on a connected graph with n nodes and n edges. He thinks it will work quickly, but it's very slow. So he wants to know the expectation of $totalCost$ of this procedure. Can you help him?

Input

The first line contains an integer n $(3 \le n \le 3000)$ — the number of nodes and edges in the graph. Each of the next n lines contains two space-separated integers a_i, b_i $(0 \le a_i, b_i \le n - 1)$ indicating an edge between nodes a_i and b_i.

Consider that the graph nodes are numbered from 0 to $(n - 1)$. It's guaranteed that there are no self-loops, no multiple edges in that graph. It's guaranteed that the graph is connected.

Output

Print a single real number — the expectation of $totalCost$. Your answer will be considered correct if its absolute or relative error does not exceed 10^{-6}.

Examples

input Copy

```
5
3 4
2 3
2 4
0 4
1 2
```

output

```
13.166666666666666
```

input

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

output

```
6.000000000000000
```

input

```
5
0 1
1 2
2 0
3 0
4 1
```

output

```
13.166666666666666
```

Note

Consider the second example. No matter what we choose first, the $totalCost$ will always be $3 + 2 + 1 = 6$.