B2. Character Swap (Hard Version)time limit per test1 secondmemory limit per test256 megabytesinputstandard inputoutputstandard output

This problem is different from the easy version. In this version Ujan makes at most 2n swaps. In addition, k≤1000,n≤50

and it is necessary to print swaps themselves. You can hack this problem if you solve it. But you can hack the previous problem only if you solve both problems.

After struggling and failing many times, Ujan decided to try to clean up his house again. He decided to get his strings in order first.

Ujan has two distinct strings s and t of length n consisting of only of lowercase English characters. He wants to make them equal. Since Ujan is lazy, he will perform the following operation at most 2n times: he takes two positions i and j (1≤i,j≤n, the values i and j can be equal or different), and swaps the characters si and tj

.

Ujan’s goal is to make the strings s and t equal. He does not need to minimize the number of performed operations: any sequence of operations of length 2n

or shorter is suitable.Input

The first line contains a single integer k (1≤k≤1000

), the number of test cases.

For each of the test cases, the first line contains a single integer n (2≤n≤50), the length of the strings s and t

.

Each of the next two lines contains the strings s and t, each having length exactly n

. The strings consist only of lowercase English letters. It is guaranteed that strings are different.Output

For each test case, output “Yes” if Ujan can make the two strings equal with at most 2n

operations and “No” otherwise. You can print each letter in any case (upper or lower).

In the case of “Yes” print m (1≤m≤2n) on the next line, where m is the number of swap operations to make the strings equal. Then print m lines, each line should contain two integers i,j (1≤i,j≤n) meaning that Ujan swaps si and tj during the corresponding operation. You do not need to minimize the number of operations. Any sequence of length not more than 2n

is suitable.ExampleInputCopy

4

5

souse

houhe

3

cat

dog

2

aa

az

3

abc

bca

OutputCopy

Yes

1

1 4

No

No

Yes

3

1 2

3 1

2 3

官方题解在此，我承认我菜到扣脚

We claim that you can make the strings equal if and only if the total number of each character in both of the strings s and t

is even.

Proof that this is a necessary condition. If we can make the strings equal, then for each position, the characters of s and t

will be the same. Therefore, each character must appear an even number of times in both strings together.

Algorithm, if all characters appear even number of times. Iterate over the index i from 1 to n. If si≠ti

, then one of the following cases holds:

* There is an index j>i

such that si=sj. Then simply swap sj with ti and then the strings will have the same character at position i. There is an index j>i such that si=tj. Then first swap tj with sj, and then swap sj with ti. Again, the strings will have the same character at position i

* .

Since we make at most 2 swaps for each position, this algorithm produces two equal strings and makes at most 2n swaps in total. The complexity of the algorithm: O(n).

This problem was introduced by [user:MikeMirzayanov] inspired by the easier version of the problem (with a single swap).

#include<iostream>

#include<cstdio>

#include<algorithm>

#include<cstring>

#include<map>

#include<vector>

using namespace std;

int main()

{

int q;

scanf("%d",&q);

while(q--)

{

int move1[200];int move2[200];

int cnt=0;

int flag=1;

int n;

scanf("%d",&n);

string a;

string b;

cin>>a;

getchar();

cin>>b;

for(int i=0;i<a.length();i++)

{

if(a[i]!=b[i])

{

for(int j=i+1;j<a.length();j++)

{ if(a[j]==b[j]) continue;

if(a[i]==a[j])

{

swap(a[j],b[i]);

move1[++cnt]=j+1;

move2[cnt]=i+1;

break;

}

}

if(a[i]!=b[i])

{

for(int j=i+1;j<a.length();j++)

{ if(a[j]==b[j]) continue;

if(a[i]==b[j])

{

swap(a[j],b[j]);

move1[++cnt]=j+1;

move2[cnt]=j+1;

swap(a[j],b[i]);

move1[++cnt]=j+1;

move2[cnt]=i+1;

break;

}

}

}

}

}

for(int i=0;i<n;i++) {if(a[i]!=b[i]) flag=0;}

if(!flag){printf("No\n");}

else {printf("Yes\n");

printf("%d\n",cnt);

for(int i=1;i<=cnt;i++)

{

printf("%d %d\n",move1[i],move2[i]);

}

}

}

return 0;

}