**A. Sereja and Dima -** [**https://codeforces.com/problemset/problem/381/A**](https://codeforces.com/problemset/problem/381/A)

Code :

#include<bits/stdc++.h>

using namespace std;

int main(){

    ios\_base::sync\_with\_stdio(false);

    cin.tie(NULL);

    int n;cin>>n;

    list<int> v;

    for(int i=0;i<n;i++){

        int x;cin>>x;

        v.push\_back(x);

    }

    int sereja=0,dima=0;

    bool s=false;

    for (int i = 0; i < n; i++)

    {

        if(s==false ){

            if(v.front()>v.back()){

                sereja+=v.front();

                v.pop\_front();

            }

            else{

                sereja+=v.back();

                v.pop\_back();

            }

            s=true;

        }

        else{

            if(v.front()>v.back()){

                dima+=v.front();

                v.pop\_front();

            }

            else{

                dima+=v.back();

                v.pop\_back();

            }

            s=false;

        }

    }

    cout<<sereja<<" "<<dima<<endl;

    return 0;

}

**A. Make it White -** [**https://codeforces.com/problemset/problem/1927/A**](https://codeforces.com/problemset/problem/1927/A)

#include<bits/stdc++.h>

using namespace std;

int main(){

    int t;cin>>t;

    while(t--){

        int n;cin>>n;

        string s;cin>>s;

        vector<int> v;

        for(int i=0;i<n;i++){

            if(s[i]=='B'){

                v.push\_back(i);

            }

        }

        sort(v.begin(),v.end());

        int ans = v.back()-v.front() + 1;

        cout<<ans<<endl;

    }

    return 0;

}

**A. Indian Summer -** [**https://codeforces.com/problemset/problem/44/A**](https://codeforces.com/problemset/problem/44/A)

#include <bits/stdc++.h>

using namespace std;

int main() {

    int n;cin >> n;

    set<pair<string, string>> st;

    for (int i = 0; i < n; ++i) {

        string species, color;

        cin >> species >> color;

        st.insert({species, color});

    }

    cout << st.size() <<endl ;

    return 0;

}

**B. Misha and Changing Handles -** [**https://codeforces.com/problemset/problem/501/B**](https://codeforces.com/problemset/problem/501/B)

#include <iostream>

#include <map>

#include <string>

using namespace std;

int main() {

int q;

cin >> q;

map<string, string> current\_to\_original;

map<string, string> original\_to\_current;

for (int i = 0; i < q; i++) {

string old\_handle, new\_handle;

cin >> old\_handle >> new\_handle;

string original;

if (current\_to\_original.count(old\_handle)) {

original = current\_to\_original[old\_handle];

current\_to\_original.erase(old\_handle);

} else {

original = old\_handle;

}

current\_to\_original[new\_handle] = original;

original\_to\_current[original] = new\_handle;

}

cout << original\_to\_current.size() << '\n';

for (auto it = original\_to\_current.begin(); it != original\_to\_current.end(); it++) {

cout << it->first << " " << it->second << '\n';

}

return 0;

}

**C. Word Game -** [**https://codeforces.com/problemset/problem/1722/C**](https://codeforces.com/problemset/problem/1722/C)

#include <bits/stdc++.h>

using namespace std;

int main(){

    int t; cin >> t;

    while(t--) {

        int n ;cin >> n;

        vector<set<string>> players(3);

        map<string, int> wordCount;

        for (int i = 0; i < 3; ++i) {

            for (int j = 0; j < n; ++j) {

                string word;

                cin >> word;

                players[i].insert(word);

                wordCount[word]++;

            }

        }

        vector<int> score(3, 0);

        for (int i = 0; i < 3; ++i) {

            for (const string& word : players[i]) {

                if (wordCount[word] == 1) score[i] += 3;

                else if (wordCount[word] == 2) score[i] += 1;

            }

        }

        cout << score[0] << " " << score[1] << " " << score[2] << endl ;

    }

    return 0;

}

**B. YetnotherrokenKeoard -** [**https://codeforces.com/problemset/problem/1907/B**](https://codeforces.com/problemset/problem/1907/B)

#include <bits/stdc++.h>

using namespace std;

int main() {

    ios\_base::sync\_with\_stdio(false);

    cin.tie(nullptr);

    int t; cin >> t;

    while (t--) {

        string input;cin >> input;

        deque<char> typed;

        deque<int> lower, upper;

        for (int i = 0; i < input.size(); ++i) {

            char ch = input[i];

            if (ch == 'b') {

                if (!lower.empty()) {

                    int idx = lower.back();

                    lower.pop\_back();

                    typed[idx] = '#';

                }

            } else if (ch == 'B') {

                if (!upper.empty()) {

                    int idx = upper.back();

                    upper.pop\_back();

                    typed[idx] = '#';

                }

            } else {

                typed.push\_back(ch);

                if (islower(ch)) lower.push\_back(typed.size() - 1);

                else if (isupper(ch)) upper.push\_back(typed.size() - 1);

            }

        }

        for (char c : typed) {

            if (c != '#') cout << c;

        }

        cout <<endl ;

    }

    return 0;

}

**C. Registration system -** [**https://codeforces.com/problemset/problem/4/C**](https://codeforces.com/problemset/problem/4/C)

#include<bits/stdc++.h>

using namespace std ;

int main(){

    int n;cin>>n;

    map<string, int>db;

    while (n--)

    {

        string name,newName;cin>>name;

        if(db.find(name)==db.end()){

            db[name]=0;

            cout<<"OK"<<endl;

        }else{

            db[name]++;

            newName=name+to\_string(db[name]);

            cout<<newName<<endl;

        }

    }

    return 0;

}

**Approach :**

**🧩 Problem Statement (in simple words)**

You are asked to design a **registration system** for usernames.

* When a user enters a new username:
  + If the username does **not exist**, accept it → print OK
  + If the username **already exists**, generate a new name by appending the smallest integer suffix i (starting from 1) such that name + i does **not exist**, and print that.

Then add the **new username** to the database as well.

**Example**

**Input:**

6

first

first

second

second

third

third

**Step-by-step processing:**

| **Request** | **Exists?** | **Output** | **Database after insertion** |
| --- | --- | --- | --- |
| first | No | OK | {first} |
| first | Yes | first1 | {first, first1} |
| second | No | OK | {first, first1, second} |
| second | Yes | second1 | {first, first1, second, second1} |
| third | No | OK | {first, first1, second, second1, third} |
| third | Yes | third1 | {first, first1, second, second1, third, third1} |

✅ Output:

OK

first1

OK

second1

OK

third1

**💡 How to Think**

We need a **fast lookup and insertion system** — that’s exactly what a **hash map** (unordered\_map in C++) provides.

We can store for each username:

* How many times it has appeared before.

Then, for each new request:

1. If name not in map → print OK, store name with count 0.
2. Else → increment the count → print name + count.

That’s it.

**🔧 Example Walkthrough**

Input:

4

abacaba

acaba

abacaba

acab

Processing:

| **Input** | **Exists?** | **Count Before** | **Output** | **Count After** |
| --- | --- | --- | --- | --- |
| abacaba | No | — | OK | 0 |
| acaba | No | — | OK | 0 |
| abacaba | Yes | 0 | abacaba1 | 1 |
| acab | No | — | OK | 0 |

✅ Output:

OK

OK

abacaba1

OK

**🧠 Algorithm (Step-by-step)**

1. Initialize an empty unordered\_map<string, int> db.
2. For each username s:
   * If s not in db:
     + Print OK
     + Set db[s] = 0
   * Else:
     + Increment db[s]
     + Print s + to\_string(db[s])
     + Insert that new name into db as well (to avoid future collision).

**⚙️ C++ Code Implementation**

#include <bits/stdc++.h>

using namespace std;

int main() {

ios::sync\_with\_stdio(false);

cin.tie(nullptr);

int n;

cin >> n;

unordered\_map<string, int> db; // stores name -> count

while (n--) {

string name;

cin >> name;

if (db.find(name) == db.end()) {

cout << "OK\n";

db[name] = 0;

} else {

db[name]++;

cout << name << db[name] << "\n";

db[name + to\_string(db[name])] = 0; // register new unique name

}

}

return 0;

}

**🧮 Complexity Analysis**

| **Operation** | **Complexity** |
| --- | --- |
| Lookup (hash map) | O(1) average |
| Insertion | O(1) average |
| String concatenation | O(1) (since names ≤ 32 chars) |
| For n requests | **O(n)** overall |

🕐 **Time Complexity:** O(n)  
💾 **Space Complexity:** O(n) — for storing all usernames in the hash map.

**✅ Example Run**

**Input**

6

first

first

second

second

third

third

**Output**

OK

first1

OK

second1

OK

third1

**🧠 Summary**

| **Step** | **Idea** |
| --- | --- |
| 1️⃣ | Use unordered\_map<string, int> to store how many times each name appeared |
| 2️⃣ | If not found → print “OK”, store count 0 |
| 3️⃣ | If found → increment count, print name + count, store that new one |
| 4️⃣ | Average O(1) per operation — super efficient |

**D. Line -** [**https://codeforces.com/problemset/problem/1722/D**](https://codeforces.com/problemset/problem/1722/D)

#include <bits/stdc++.h>

using namespace std;

#define ll long long

int main() {

    int t;cin >> t;

    while (t--) {

        int n;string s;cin >> n >> s;

        vector<ll> gains;

        ll base = 0;

        for (int i = 0; i < n; ++i) {

            if (s[i] == 'L') {

                base += i;

                gains.push\_back((n - i - 1) - i);

            } else {

                base += (n - i - 1);

                gains.push\_back(i - (n - i - 1));

            }

        }

        sort(gains.rbegin(), gains.rend());

        vector<ll> result;

        ll current = base;

        for (int k = 0; k < n; ++k) {

            if (gains[k] > 0)

                current += gains[k];

            result.push\_back(current);

        }

        for (ll val : result)

            cout << val << " ";

        cout << endl ;

    }

    return 0;

}

**Approach**

**🧠 Problem Summary**

Each person in the line looks either **left (L)** or **right (R)**. Their **count** is the number of people they see in that direction:

* If a person at index i looks left → they see i people
* If they look right → they see n - i - 1 people

The **value of the line** is the sum of all these counts.

You can **flip the direction** of at most k people. For each k = 1 to n, compute the **maximum possible value** of the line.

**💡 Key Insight**

Instead of recalculating the entire line for each k, we:

1. Compute the **initial value** of the line.
2. For each person, calculate the **gain** if we flip their direction:
   * If L at position i: gain = (n - i - 1) - i
   * If R at position i: gain = i - (n - i - 1)
3. Sort the gains in **descending order**.
4. For each k, add the top k gains (only if positive) to the initial value.

**✅ Final C++ Code**

#include <bits/stdc++.h>

using namespace std;

#define ll long long

int main() {

int t;

cin >> t;

while (t--) {

int n;

string s;

cin >> n >> s;

vector<ll> gains;

ll base = 0;

for (int i = 0; i < n; ++i) {

if (s[i] == 'L') {

base += i;

gains.push\_back((n - i - 1) - i);

} else {

base += (n - i - 1);

gains.push\_back(i - (n - i - 1));

}

}

sort(gains.rbegin(), gains.rend());

vector<ll> result;

ll current = base;

for (int k = 0; k < n; ++k) {

if (gains[k] > 0) current += gains[k];

result.push\_back(current);

}

for (ll val : result) cout << val << " ";

cout << '\n';

}

return 0;

}

**🧪 Example Walkthrough**

**Input:**

1

5

LRRLL

**Initial value:**

* L at 0 → sees 0
* R at 1 → sees 3
* R at 2 → sees 2
* L at 3 → sees 3
* L at 4 → sees 4  
  → Total = 0 + 3 + 2 + 3 + 4 = **12**

**Gains:**

* L at 0 → gain = 4 - 0 = **4**
* R at 1 → gain = 1 - 3 = **-2**
* R at 2 → gain = 2 - 2 = **0**
* L at 3 → gain = 1 - 3 = **-2**
* L at 4 → gain = 0 - 4 = **-4**

Sorted gains: [4, 0, -2, -2, -4]

**Output:**

16 16 16 16 16

**⏱️ Time Complexity**

* Each test case:
  + O(n) for base value and gain computation
  + O(n log n) for sorting
  + O(n) for prefix sum

✅ Efficient for up to 200,000 people across all test cases

**B. Chat Order -** [**https://codeforces.com/problemset/problem/637/B**](https://codeforces.com/problemset/problem/637/B)

#include <bits/stdc++.h>

using namespace std;

int main() {

    int n; cin >> n;

    vector<string> v(n);

    unordered\_set<string> seen;

    vector<string> result;

    for (int i = 0; i < n; ++i) {

        cin >> v[i];

    }

    for (int i = n - 1; i >= 0; i--) {

        if (seen.find(v[i]) == seen.end()) {

            seen.insert(v[i]);

            result.push\_back(v[i]);

        }

    }

    for (auto name : result) {

        cout << name << '\n';

    }

    return 0;

}

**A. Tom Riddle's Diary -** [**https://codeforces.com/problemset/problem/855/A**](https://codeforces.com/problemset/problem/855/A)

#include<bits/stdc++.h>

using namespace std;

int main(){

    vector<string>v;

    int n;cin>>n;

    while (n--)

    {

        string s;cin>>s;

        auto it = find(v.begin(), v.end(), s);

        if(it != v.end())

            cout<<"YES"<<endl;

        else{

            cout<<"NO"<<endl;

            v.push\_back(s);

        }

    }

    return 0;

}

**A. Strong Password -** [**https://codeforces.com/problemset/problem/1997/A**](https://codeforces.com/problemset/problem/1997/A)

#include <bits/stdc++.h>

using namespace std;

int main()

{

    int t;

    cin >> t;

    while (t--)

    {

        string s, s2 = "", newS;

        cin >> s;

        string s1 = "";

        s1 += s[0];

        bool found = false;

        int idx;

        char c;

        if (s.size() == 1)

        {

            if (s[0] == 'a')

                cout << "b" << s << endl;

            else

                cout << "a" << s << endl;

        }

        else

        {

            for (int i = 1; i < s.size(); i++)

            {

                if (s[i] == s[i - 1])

                {

                    idx = i;

                    if (s[i] == 'a')

                        c = 'b';

                    else

                        c = 'a';

                    found = true;

                    break;

                }

                s1 += s[i];

            }

            if (found)

            {

                for (int i = idx; i < s.size(); i++)

                {

                    s2 += s[i];

                }

                newS = s1 + c + s2;

            }

            else

            {

                if (s.back() == 'a')

                    newS = s + "b";

                else

                    newS = s + "a";

            }

            cout << newS << endl;

        }

    }

    return 0;

}

**B. Symmetric Encoding -** [**https://codeforces.com/problemset/problem/1974/B**](https://codeforces.com/problemset/problem/1974/B)

#include <bits/stdc++.h>

using namespace std;

int main() {

    ios::sync\_with\_stdio(false);

    cin.tie(nullptr);

    int t; cin >> t;

    while (t--) {

        int n;string s;cin >> n >> s;

        set<char> st(s.begin(), s.end());

        string r(st.begin(), st.end());

        map<char, char> mp;

        int m = r.size();

        for (int i = 0; i < m; i++) {

            mp[r[i]] = r[m - i - 1];

        }

        for (char ch : s)

            cout << mp[ch];

        cout << "\n";

    }

    return 0;

}

**B. Lady Bug -** [**https://codeforces.com/problemset/problem/2092/B**](https://codeforces.com/problemset/problem/2092/B)

#include <bits/stdc++.h>

using namespace std;

int main()

{

    ios\_base::sync\_with\_stdio(0);

    cin.tie(0), cout.tie(0);

    int t ; cin >> t;

    while (t--){

        int n;cin >> n;

        string s, ss;cin >> s >> ss;

        string s1, s2;

        for (int i = 0; i < n; i++)

        {

            if (i & 1)

            {

                s1.push\_back(ss[i]);

                s2.push\_back(s[i]);

            }

            else

            {

                s1.push\_back(s[i]);

                s2.push\_back(ss[i]);

            }

        }

        int zeros1 = 0, zeros2 = 0;

        for (int i = 0; i < s1.size(); i++)

        {

            if (s1[i] == '0')

            {

                zeros1++;

            }

        }

        for (int i = 0; i < s2.size(); i++)

        {

            if (s2[i] == '0')

            {

                zeros2++;

            }

        }

        if (zeros1 >= ceil(s1.size() / 2.0) && zeros2 >= (s2.size() / 2))

        {

            cout << "YES" << endl;

        }

        else

        {

            cout << "NO" << endl;

        }

    }

    return 0;

}

**Solution :**

**🧩 Problem Understanding — “Lady Bug”**

You are given **two binary strings** a and b, each of length n.

Lady Bug wants to make the entire first string a consist of **only zeros** (000...0).

**🔁 Allowed Operations**

She can perform **any number of operations**.  
In one operation, she can choose any index i (2 ≤ i ≤ n) and:

1. Swap a[i] with b[i−1], **OR**
2. Swap b[i] with a[i−1].

So she can swap diagonally between the two rows, but only between **adjacent columns**.

**🧠 Example to Visualize**

Let’s visualize a and b as two rows of a grid:

| **Index** | **1** | **2** | **3** | **4** | **5** |
| --- | --- | --- | --- | --- | --- |
| **a** | 0 | 1 | 0 | 0 | 0 |
| **b** | 0 | 1 | 0 | 1 | 1 |

Here, allowed swaps are diagonals:

* (a[2], b[1])
* (b[2], a[1])
* (a[3], b[2])
* (b[3], a[2])  
  and so on.

So effectively, you can move bits along **zigzag diagonal paths**:

a1 ↔ b2 ↔ a3 ↔ b4 ↔ ...

b1 ↔ a2 ↔ b3 ↔ a4 ↔ ...

This leads to a powerful insight 👇

**💡 Key Insight — “Two Zigzag Paths”**

All the bits can move freely **only along these two zigzag lines**:

1. Zigzag 1: a1, b2, a3, b4, ...
2. Zigzag 2: b1, a2, b3, a4, ...

⚙️ You can never swap bits **between** these two zigzags.  
So, effectively, these are **two independent groups of cells**.

**🎯 Goal Restated**

We want the entire a string to become all zeros.

That means:

* Every cell of a should be able to get a ‘0’ from its corresponding zigzag path.

So for that to be possible:

* **Each zigzag** must contain **at least as many zeros** as the number of a-cells it covers.

**🔍 Step-by-Step Approach**

1. Build two zigzag strings:
   * s1 = a[0], b[1], a[2], b[3], ...
   * s2 = b[0], a[1], b[2], a[3], ...
2. Count the number of zeros in each:
   * zeros1 = count of ‘0’ in s1
   * zeros2 = count of ‘0’ in s2
3. Compute the number of positions each zigzag needs to cover in a:
   * s1 covers ⌈n/2⌉ positions of a
   * s2 covers ⌊n/2⌋ positions of a
4. If each zigzag has at least that many zeros → **YES**, else **NO**.

**✅ C++ Solution (Final Correct and Accepted)**

#include <bits/stdc++.h>

using namespace std;

int main() {

ios\_base::sync\_with\_stdio(false);

cin.tie(nullptr);

int t;

cin >> t;

while (t--) {

int n;

cin >> n;

string a, b;

cin >> a >> b;

string s1, s2;

for (int i = 0; i < n; i++) {

if (i % 2 == 0) {

s1.push\_back(a[i]);

s2.push\_back(b[i]);

} else {

s1.push\_back(b[i]);

s2.push\_back(a[i]);

}

}

int zeros1 = count(s1.begin(), s1.end(), '0');

int zeros2 = count(s2.begin(), s2.end(), '0');

int need1 = (n + 1) / 2; // ceil(n/2)

int need2 = n / 2; // floor(n/2)

if (zeros1 >= need1 && zeros2 >= need2)

cout << "YES\n";

else

cout << "NO\n";

}

return 0;

}

**🧮 Example Walkthrough**

**Input**

6

010001

010111

a = 010001

b = 010111

Now form the zigzags:

* s1 = a1 b2 a3 b4 a5 b6 = 0 1 0 1 0 1 → zeros1 = 3
* s2 = b1 a2 b3 a4 b5 a6 = 0 1 0 0 1 0 → zeros2 = 3
* need1 = 3 (ceil(6/2))
* need2 = 3 (floor(6/2))

Both zigzags have enough zeros ✅ → **YES**

**⏱️ Complexity Analysis**

| **Aspect** | **Complexity** |
| --- | --- |
| Building zigzags | O(n) |
| Counting zeros | O(n) |
| Total per test | **O(n)** |
| Total across all tests (∑n ≤ 2×10⁵) | **O(2×10⁵)** → very fast |
| Space complexity | **O(n)** |

**🧠 Summary of Thought Process**

| **Step** | **Reasoning** |
| --- | --- |
| 1️⃣ | Recognize swaps only occur diagonally |
| 2️⃣ | Identify two independent zigzag paths |
| 3️⃣ | Realize bits can move only within their zigzag |
| 4️⃣ | Count zeros in each zigzag |
| 5️⃣ | Check if each zigzag has enough zeros to fill all a positions it influences |

**B. Remove Prefix -** [**Problem - 1714B - Codeforces**](https://codeforces.com/problemset/problem/1714/B)

#include <bits/stdc++.h>

using namespace std;

int main() {

    ios::sync\_with\_stdio(false);

    cin.tie(nullptr);

    int t;cin >> t;

    while (t--) {

       int n;cin>>n;

       vector<int>v;

       for (int i = 0; i < n; i++)

       {

            int x;cin>>x;

            v.push\_back(x);

       }

       set<int>seen;

       int idx = n-1;

       while(idx >= 0){

            if(seen.count(v[idx])) break;

            seen.insert(v[idx]);

            idx--;

       }

       cout<<idx + 1 << endl ;

    }

    return 0;

}

**B. Two-gram -** [**https://codeforces.com/problemset/problem/977/B**](https://codeforces.com/problemset/problem/977/B)

#include<bits/stdc++.h>

using namespace std;

int main(){

    ios\_base::sync\_with\_stdio(false);

    cin.tie(nullptr);

    int n;cin>>n;

    string s;cin>>s;

    set<string> st;

    map<string,int> mp;

    multimap<int,string> mp1;

    for(int i=0;i<n-1;i++){

        string t="";

        t+=s[i];

        t+=s[i+1];

        st.insert(t);

        mp[t]++;

    }

    for(auto x:mp){

        mp1.insert({x.second,x.first});

    }

    auto it=mp1.end();

    it--;

    cout<<it->second<<endl;

    return 0;

}

**E1. Permutation Minimization by Deque -** [**https://codeforces.com/problemset/problem/1579/E1**](https://codeforces.com/problemset/problem/1579/E1)

#include <bits/stdc++.h>

using namespace std;

int main() {

    ios::sync\_with\_stdio(false);

    cin.tie(nullptr);

    int t;cin >> t;

    while (t--) {

        int n ; cin >> n;

        vector<int> p(n);

        for (int i = 0; i < n; i++)

            cin >> p[i];

        deque<int> dq;

        dq.push\_back(p[0]);

        for (int i = 1; i < n; i++) {

            if (p[i] < dq.front())

                dq.push\_front(p[i]);

            else

                dq.push\_back(p[i]);

        }

        for (int x : dq)

            cout << x << " ";

        cout << "\n";

    }

    return 0;

}

**Palindrome Reorder -** [**https://cses.fi/problemset/task/1755**](https://cses.fi/problemset/task/1755)

**🔍 Problem Understanding**

We are given a string s (A–Z).  
We need to rearrange the letters so the string becomes a **palindrome** (reads same forwards and backwards).

A **palindrome condition** for character frequencies:

* If the length n is **even** → all characters must appear an **even** number of times.
* If n is **odd** → **only one character** can appear an odd number of times.

If that condition fails → "NO SOLUTION".

**🧠 Approach**

1. Count the frequency of each character.
2. Check how many characters have an odd frequency:
   * If more than 1 → **No solution.**
3. Construct the palindrome:
   * Build the **first half** by taking freq[c] / 2 of each character.
   * If there’s a character with odd count, place it in the **middle**.
   * The **second half** is the reverse of the first half.

**🧩 Simplified Clean Version**

Cleaner and clearer version (recommended):

#include <bits/stdc++.h>

using namespace std;

int main() {

string s; cin >> s;

map<char, int> freq;

for (char c : s) freq[c]++;

int oddCount = 0;

char oddChar;

for (auto [ch, cnt] : freq) {

if (cnt % 2 != 0) {

oddCount++;

oddChar = ch;

}

}

if (oddCount > 1) {

cout << "NO SOLUTION\n";

return 0;

}

string first = "", second = "";

for (auto [ch, cnt] : freq)

first += string(cnt / 2, ch);

second = first;

reverse(second.begin(), second.end());

if (oddCount)

cout << first << string(freq[oddChar] % 2 ? 1 : 0, oddChar) << second;

else

cout << first << second;

}

**⚙️ Example Walkthrough**

**Input:**

AAAACACBA

**Frequencies:**

A → 5

B → 1

C → 2

* Odd characters → A, B → 2 → ❌ (Wait)  
  Actually, total = 8 letters → even → NO SOLUTION?  
  No, check again: total = 9 letters → odd, so **only one odd allowed.**  
  A = 5 (odd), B = 1 (odd) → 2 odds → ❌  
  So we must fix counts:  
  A=5, B=1, C=2 → oddCount = 2 → "NO SOLUTION" ✅

But if B had 0 (like AAAACACAA):

A → 6

C → 2

=> Even counts only → palindrome = AACACAA ✅

**⏱️ Time & Space Complexity**

| **Aspect** | **Complexity** |
| --- | --- |
| Counting frequencies | **O(n)** |
| Building palindrome | **O(n)** |
| Total | **O(n)** |
| Space | **O(26)** → constant |