Cấu trúc dữ liệu và giải thuật

Nhóm 15

BÀI TẬP VỀ NHÀ BUỔI 1

Họ và tên	**Mã sinh viên**
Vũ Công Tuấn Dương	B22DCKH024

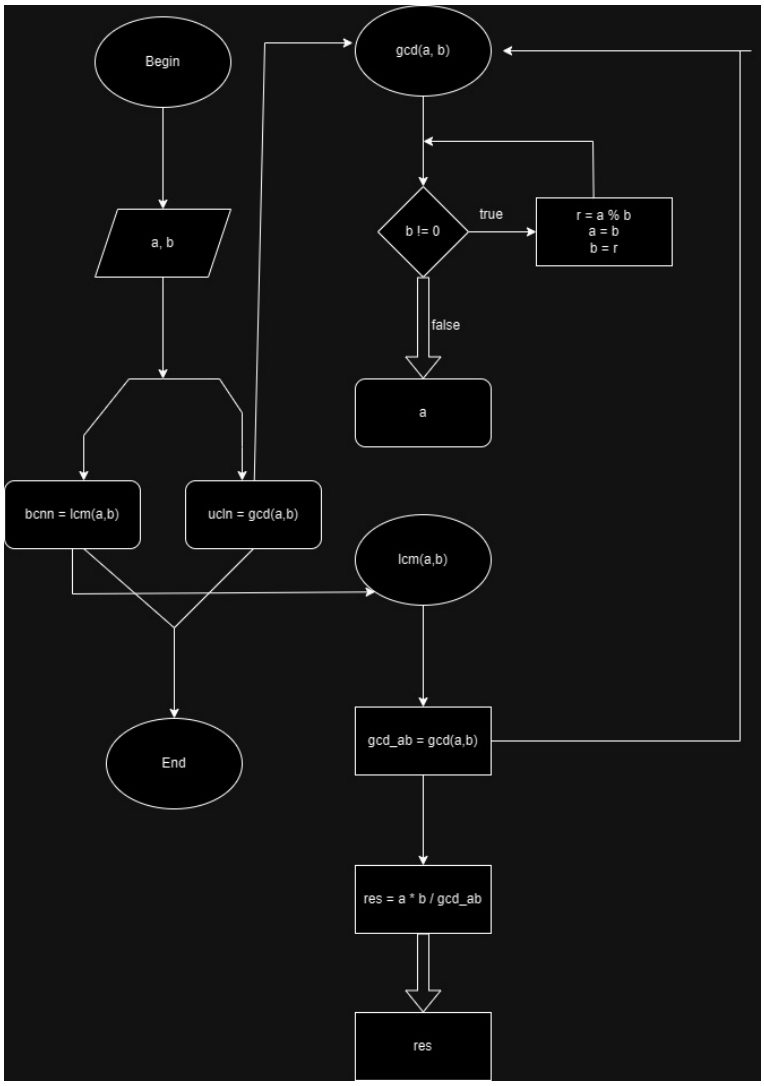
Giảng viên:

TS Đỗ Thị Liên

https://www.overleaf.com/project/65ab8b190051c8e50463843e **Ngày nộp : 21/01/2024**

1 ƯỚC SỐ CHUNG LỚN NHẤT VÀ BỘI SỐ CHUNG NHỎ NHẤT

1. Xác định input, output:
   - Input: Hai số nguyên dương a, b
   - Output: UCLN và BCNN của a và b
2. Phân tích bài toán: Để tìm ước số chung lớn nhất của hai số nguyên dương, ta áp dụng thuật toán Euclid và để tìm bội số chung nhỏ nhất của hai số, ta tính tích hai số rồi chia cho ước số chung lớn nhất của hai số:
3. Vẽ lưu đồ: Lưu đồ được thể hiện ở hình 1.



Hình 1: Lưu đồ thuật toán bài 1

4. Chạy từng bước
5. Viết code: Code được viết như sau:

```
#include <iostream>
using namespace std;
int gcd(int a, int b) {
    while (b != 0) {
        int r = a % b;
        a = b;
        b = r;
    }
    return a;
}
long long lcm(int a, int b) {
    int gcd_ab = gcd(a, b);
    long long res = a * b / gcd_ab;
    return res;
}
int main () {
    int t;
    cin >> t;
    while (t--) {
        int a, b;
        cin >> a >> b;
        int ucln = gcd(a, b);//chỉ thị tích cực
        long long bcnn = lcm(a, b);//chỉ thị tích cực
        cout << ucln << " " << bcnn << endl;
    }
    return 0;
}
```

6. Đánh giá độ phức tạp thuật toán: O(t * log(min(a,b)))

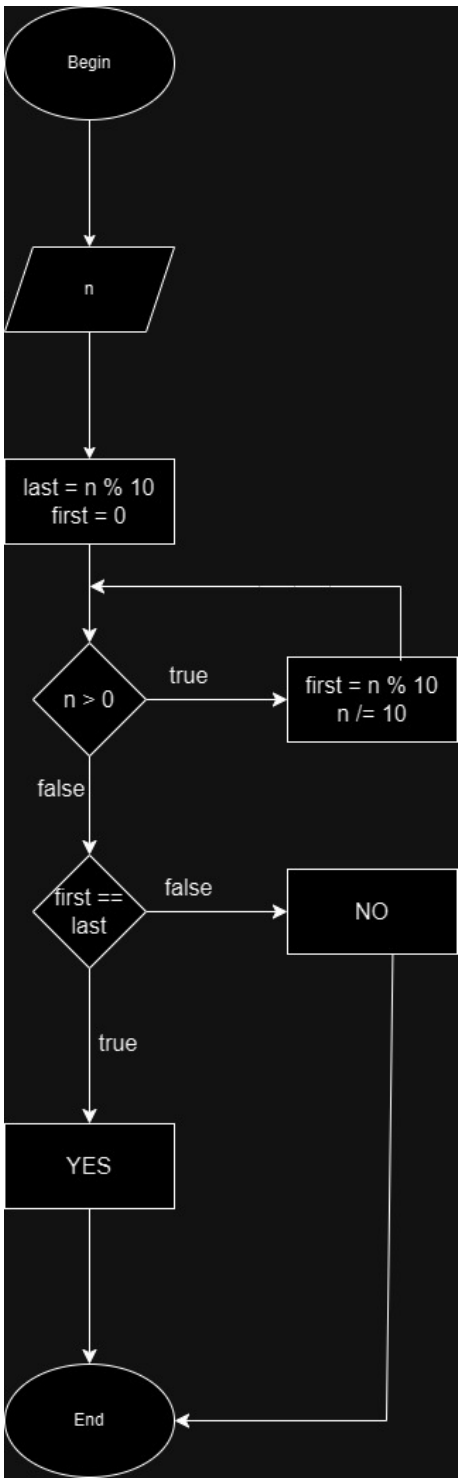
2 BẮT ĐẦU VÀ KẾT THÚC

1. Xác định input, output:
   - Input: 1 số nguyên dương n
   - Output: YES (nếu chữ số đầu tiên và chữ số cuối cùng của n giống nhau) hoặc NO(nếu chữ số đầu tiên và chữ số cuối cùng của n khác nhau)
2. Phân tích bài toán:
   - Ta dễ thấy chữ số cuối cùng của n là số dư của n khi chia cho 10
   - Để tính chữ số đầu tiên của n, ta sẽ lặp từ chữ số cuối cùng cho đến chữ số đầu tiên của n bằng cách chia n cho 10 và cập nhật giá trị của n
3. Vẽ lưu đồ: Lưu đồ được thể hiện ở hình 2
4. Chạy từng bước

5. Viết code: Code được viết như sau:

```
#include <bits/stdc++.h>
using namespace std;
void testCase();
int main () {
    int t;
    cin >> t;
    while (t--) {
        testCase();
        cout << endl;
    }
    return 0;
}
void testCase() {
    int n;
    cin >> n;
    int last = n % 10;
    int first = 0;
    while (n > 0) {
        first = n % 10;//chỉ thị tích cực
        n /= 10;
    }
    if (first == last) {
        cout << "YES";
    } else {
        cout << "NO";
    }
}
```

6. Đánh giá độ phức tạp thuật toán: O (t * log(n))



Hình 2: Lưu đồ thuật toán bài 2

3 MẢNG ĐỐI XỨNG

1. Xác định input, output:

   - Input: Dãy số nguyên có n phần tử
   - Output: YES nếu các phần tử của dãy đối xứng hoặc NO nếu các phần tử của dãy không đối xứng

2. Phân tích bài toán:

    - Cách 1: Do ta đang xét tính đối xứng của mảng nên ứng với mỗi phần tử tại một chỉ số bất kỳ, ta sẽ có một phần tử tại chỉ số đối xứng với chỉ số đó tương ứng. Nên ta sẽ chỉ duyệt từ đầu mảng đến giữa mảng và kiểm tra
    - Cách 2: Dùng hai con trỏ, thực hiện duyệt tuần tự từ đầu mảng đến giữa mảng và từ cuối mảng về giữa mảng cùng một lúc. Mỗi lần duyệt, so sán giá trị của các phần tử tại 2 vị trí đó. Nếu khác nhau thì mảng đó không đối xứng. Khi 2 con trỏ đó đứng tại cùng vị trí thì đã duyệt xong, thuật toán kết thúc

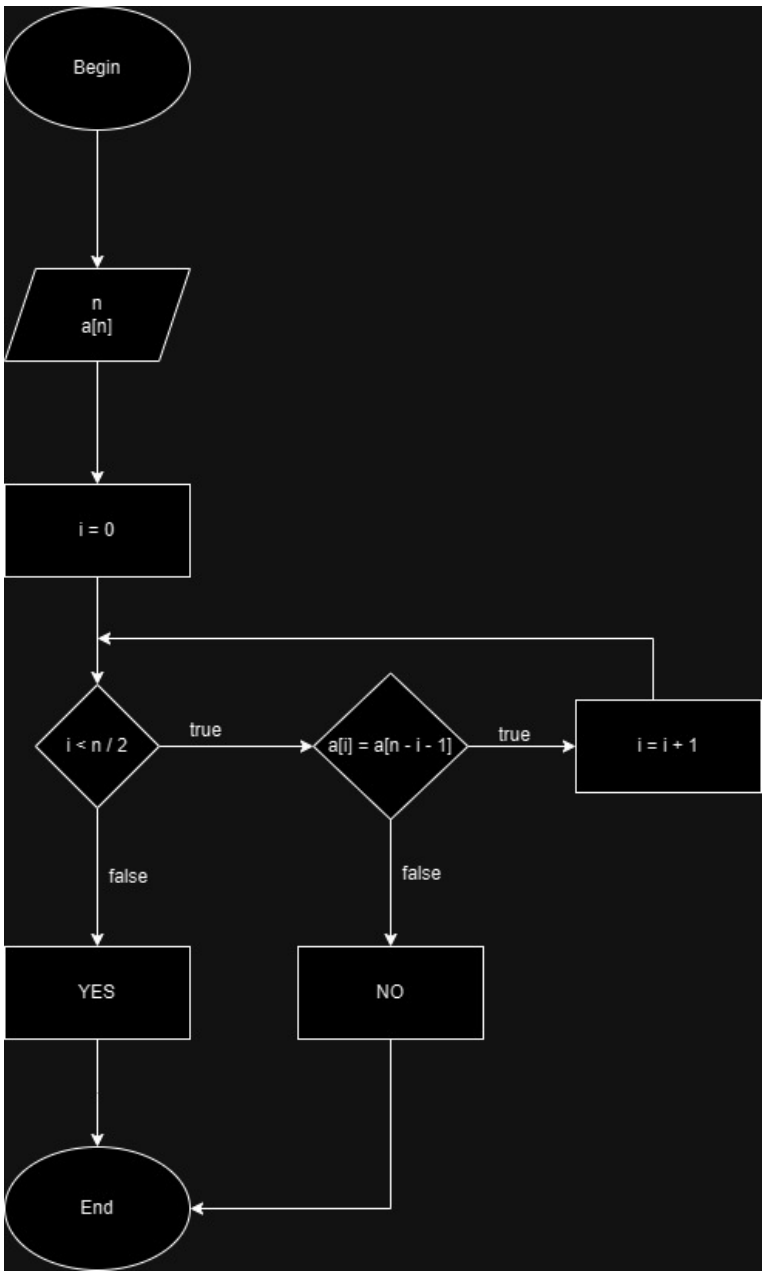
3. Vẽ lưu đồ: Lưu đồ được thể hiện ở hình 3

4. Chạy từng bước

5. Viết code: Code được viết như sau:

```
#include <bits/stdc++.h>
using namespace std;
void testCase();
int main () {
    int t;
    cin >> t;
    while (t--) {
        testCase();
        cout << endl;
    }
    return 0;
}
void testCase() {
    int n;
    cin >> n;
    int a[n];
    for (int i = 0; i < n; i++) {
        cin >> a[i];
    }
    for (int i = 0; i < n / 2; i++) {
        if (a[i] != a[n - i - 1]) {
            cout << "NO";//chỉ thị tích cực
            return;
        }
    }
    cout << "YES";
}
```

6. Đánh giá độ phức tạp thuật toán: O(t * n)



Hình 3: Lưu đồ thuật toán bài 3

4 PHÂN TÍCH THỪA SỐ NGUYÊN TỐ

1. Xác định input, output:
   - Input: Số nguyên dương n
   - Output: Các thừa số nguyên tố(đi kèm số mũ) của n
2. Phân tích bài toán: Để phân tích thành các thừa số nguyên tố, ta chia lần lượt n cho 2,3,5,..(các số nguyên tố) , với mỗi số nguyên tố, ta chia đến khi nào không chia hết cho số đó thì chuyển sang số nguyên tố kế tiếp. Do đó ta thấy số chia chỉ có thể từ 2 đến $\sqrt{n}$ .Với mỗi lần chia ta sẽ kiểm soát số mũ của thừa số nguyên tố bằng một biến đếm
3. Vẽ lưu đồ: Lưu đồ được thể hiện ở hình 4

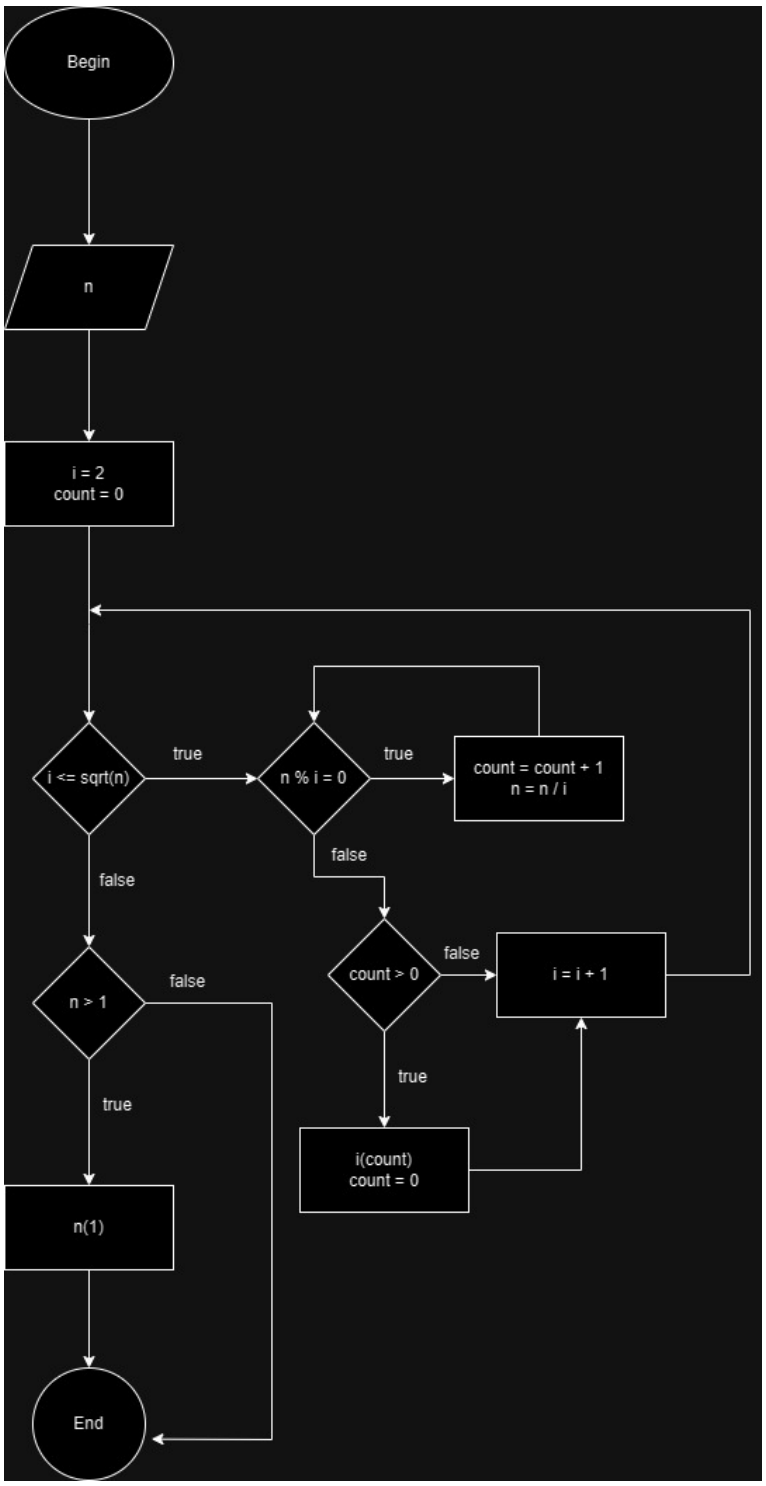
4. Chạy từng bước

5. Viết code: Code được viết như sau:

```
#include <bits/stdc++.h>
using namespace std;
void testCase();
int main () {
    int t;
    cin >> t;
    for  (int i = 1; i <= t; i++) {//p3
        cout << "Test " << i << ": ";
        testCase();
        cout << endl;
    }
    return 0;
}
void testCase() {
    int n;
    cin >> n;
    for (int i = 2; i <= sqrt(n); i++) {//p2
        int count = 0;
        while (n % i == 0) { //p1
            count++;//chỉ thị tích cực
            n /= i;
        }
        if (count) {
            cout << i << "(" << count << ") ";
        }
    }
    if (n > 1) {
        cout << n << "(1)";
    }
}
```

6. Đánh giá độ phức tạp thuật toán: O ($t * \sqrt{n} * log(n)$)

   - Xét p1: O(log(n))
   - Xét p2: O($\sqrt{n}$)
   - Xét p3: O(t)



Hình 4: Lưu đồ thuật toán bài 4

5 PHÉP CỘNG

1. Xác định input, output:
   - Input: Một phép toán có dạng a + b = c (a,b,c là số nguyên có 1 chữ số)
   - Output: YES nếu phép toán đúng hoặc NO nếu phép toán sai

2. Phân tích bài toán: Ta thấy a, b, c là các số có 1 chữ số nên sẽ ở các vị trí cố định trong xâu đầu vào. Vì thế ta chỉ việc tách các số từ xâu cho trước rồi so sánh, kiểm tra điều kiện

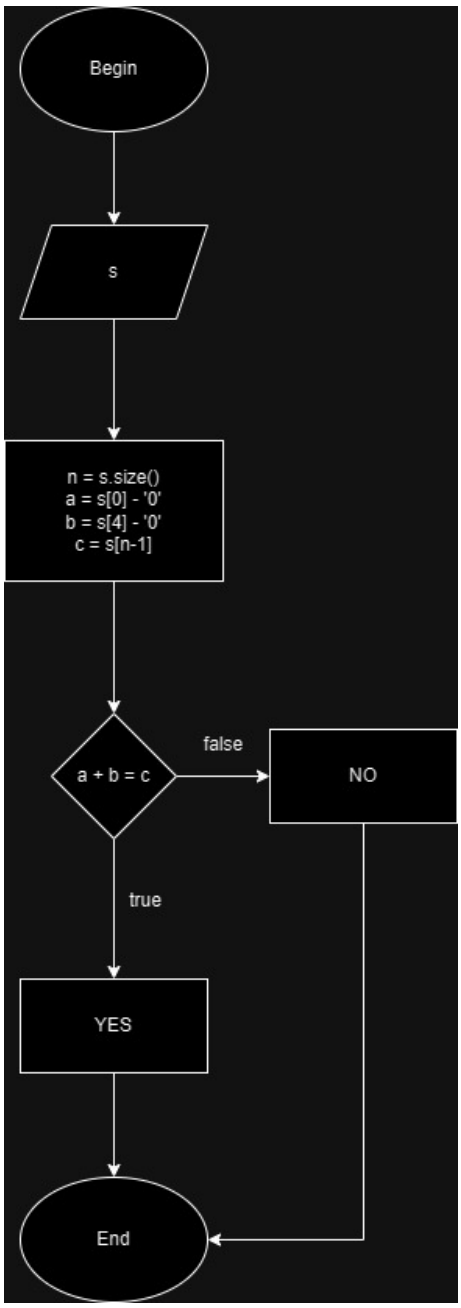
3. Vẽ lưu đồ: Lưu đồ được thể hiện ở hình 5

4. Chạy từng bước

5. Viết code: Code được viết như sau:

```
#include <bits/stdc++.h>
using namespace std;
void testCase();
int main () {
    testCase();
    return 0;
}
void testCase() {
    string s;
    getline(cin, s);
    int c = s[s.size() - 1] - '0';
    int a = s[0] - '0';
    int b = s[4] - '0';
    if (a + b == c) {
        cout << "YES";
    } else {
        cout << "NO";
    }
}
```

6. Đánh giá độ phức tạp thuật toán: O(t)



Hình 5: Lưu đồ thuật toán bài 5

6 SỐ TĂNG GIẢM

1. Xác định input, output:

   - Input: Số nguyên dương n có tối đa 500 chữ số
   - Output:YES nếu đó đúng là số tăng giảm, chữ NO nếu ngược lại.

2. Phân tích bài toán: Do n có tối đa 500 chữ số nên ta sẽ sử dụng kiểu dữ liệu string để lưu trữ số. Một số được gọi là số tăng giảm nếu số đó có các chữ số thỏa mãn hoặc không giảm, hoặc không tăng từ trái qua phải. Do đó với mỗi điều kiện không giảm, không tăng thì ta sẽ đi từ trái qua phải và kiểm tra chữ số với chữ số đứng sau nó. Nếu không thoả mãn sẽ ngay

lập tức dừng và trả kết quả là không thoả mãn. Nếu chạy đến chữ số cuối cùng mà vẫn đúng thì số đó thoả mãn điều kiện

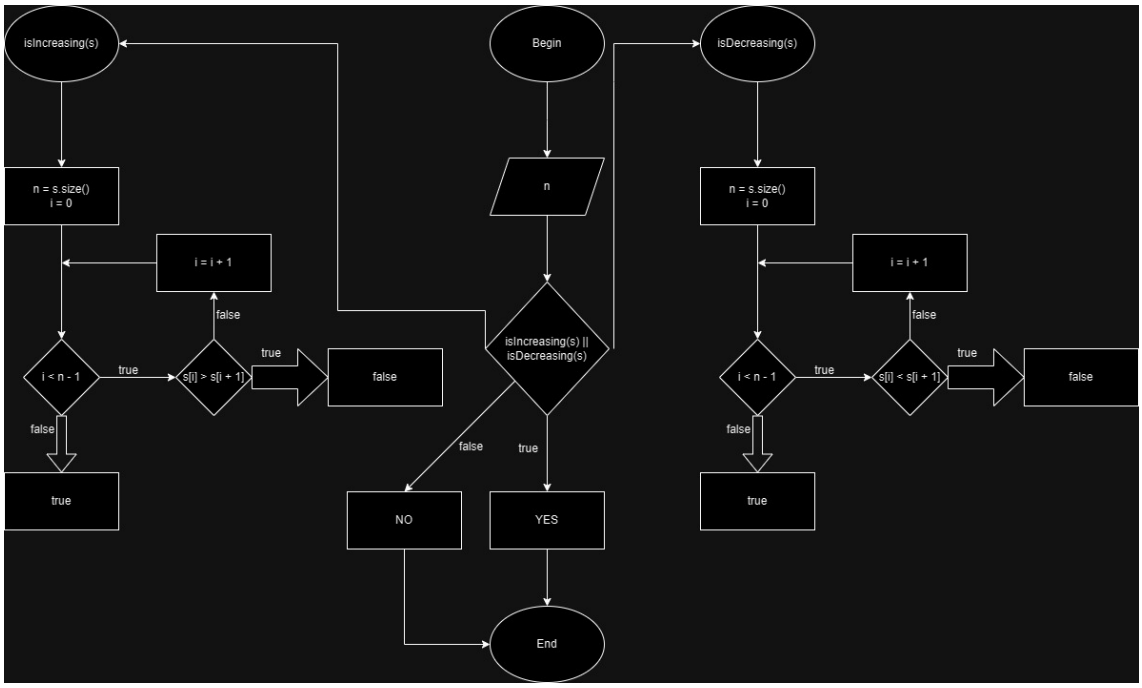
3. Vẽ lưu đồ: Lưu đồ được thể hiện ở hình 6
4. Chạy từng bước
5. Viết code: Code được viết như sau:

```
#include <bits/stdc++.h>
using namespace std;
void testCase();
bool isIncreasing(string s);
bool isDecreasing(string s);
int main () {
    int t;
    cin >> t;
    while (t--) {
        testCase();
        cout << endl;
    }
    return 0;
}
bool isIncreasing(string s) {
    for (int i = 0; i < s.length() - 1; i++) {
        if (s[i] > s[i + 1]) {
            return false;
        }
    }
    return true;

}
bool isDecreasing(string s) {
    for (int i = 0; i < s.length() - 1; i++) {
        if (s[i] < s[i + 1]) {
            return false;
        }
    }
    return true;

}
void testCase() {
    string n;
    cin >> n;
    if (isIncreasing(n) || isDecreasing(n)) {
        cout << "YES";//chỉ thị tích cực
    } else {
        cout << "NO";
    }
}
```

6. Đánh giá độ phức tạp thuật toán: O(t * n) với n là độ dài xâu đầu vào



Hình 6: Lưu đồ thuật toán bài 6

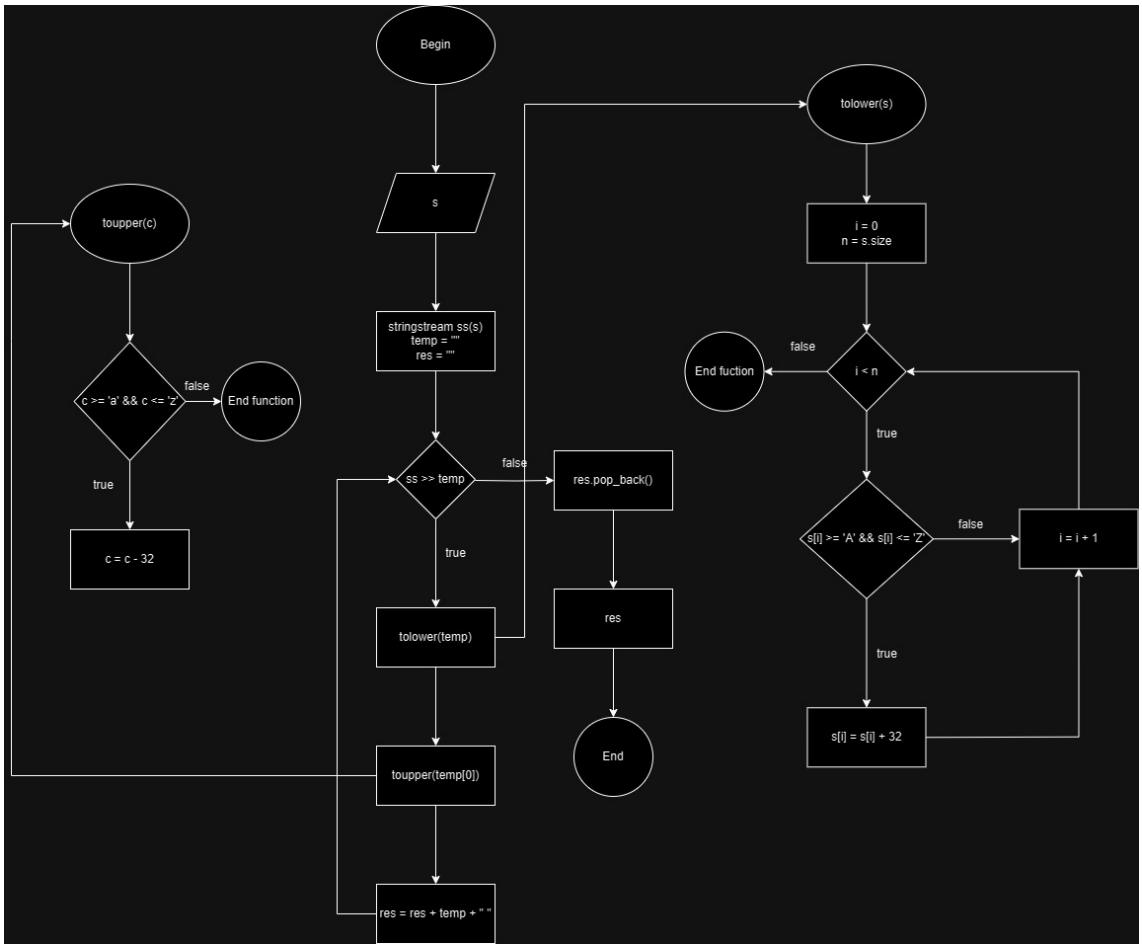
7 CHUẨN HÓA 1

1. Xác định input, output:
   - Input: Xâu họ tên s chưa được chuẩn hoá
   - Output: Xâu họ tên đã được chuẩn hoá
2. Phân tích bài toán: Để có thể loại bỏ kí tự space thừa và chuẩn hoá tên, ta sẽ phải tách từ và với mỗi từ ta sẽ in hoa chữ cái đầu tiên và in thường các chữ cái còn lại. Để chuyển từ in hoa thành in thường và ngược lại, ta cần chú ý đến các chỉ số trong bảng mã ASCII, ta thấy kí tự 'A' trong bảng mã là số 65 và kí tự 'a' là 97, hai kí tự cách nhau 32 số. Do đó, để chuyển từ chữ cái in hoa thành in thường, ta có thể cộng 32 vào giá trị của biến lưu, và ngược lại
3. Vẽ lưu đồ: Lưu đồ được thể hiện ở hình 7
4. Chạy từng bước
5. Viết code: Code được viết như sau:

```
#include <bits/stdc++.h>
using namespace std;
void testCase();
void tolower(string &s);
void toupper(char &c);
int main () {
    int t;
    cin >> t;
    cin.ignore();
    while (t--) {
        testCase();
        cout << endl;
    }
    return 0;
}
void toupper(char &c) {
    if (c >= 'a' && c <= 'z') {
        c -= 32;
    }
}
void tolower(string &s) {
    for (int i = 0; i < s.size(); i++) {
        if (s[i] >= 'A' && s[i] <= 'Z') {
            s[i] += 32;
        }
    }

}
void testCase() {
    string s;
    getline(cin, s);
    stringstream ss(s);
    string temp;
    string res = "";
    while (ss >> temp) {
        tolower(temp);//chỉ thị tích cực
        toupper(temp[0]);
        res += temp + " ";
    }
    res.pop_back();
    cout << res;
}
```

6. Đánh giá độ phức tạp thuật toán: O(t * n) với n là độ dài xâu đầu vào



Hình 7: Lưu đồ thuật toán bài 7

8 CHUẨN HÓA 2

1. Xác định input, output:
   - Input: Xâu họ tên s chưa được chuẩn hoá
   - Output: Xâu họ tên đã được chuẩn hoá
2. Phân tích bài toán: Tương tự bài 7, chỉ khác ở việc là cần in hoa hết các chữ cái của từ đầu tiên và viết theo thứ tự khác
3. Vẽ lưu đồ: Lưu đồ được thể hiện ở hình 8
4. Chạy từng bước
5. Viết code: Code được viết như sau:

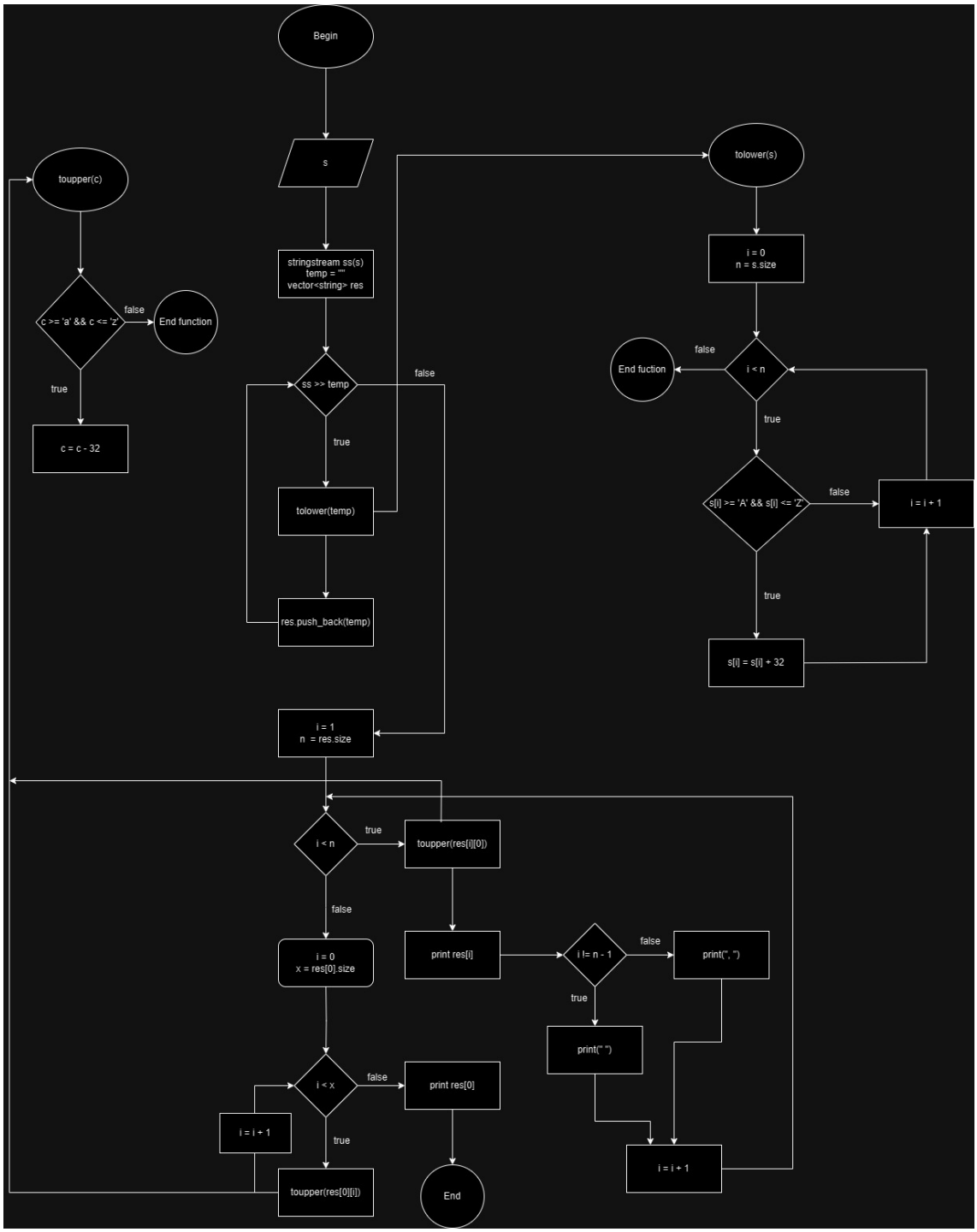
```
#include <bits/stdc++.h>
using namespace std;
void testCase();
void tolower(string &s);
```

```
void toupper(char &c);
int main () {
    int t;
    cin >> t;
    cin.ignore();
    while (t--) {
        testCase();
        cout << endl;
    }
    return 0;
}
void toupper(char &c) {
    if (c >= 'a' && c <= 'z') {
        c -= 32;
    }
}
void tolower(string &s) {
    for (int i = 0; i < s.size(); i++) {
        if (s[i] >= 'A' && s[i] <= 'Z') {
            s[i] += 32;
        }
    }

}
void testCase() {
    string s;
    getline(cin, s);
    stringstream ss(s);
    string temp;
    vector<string> res;
    while (ss >> temp) {
        tolower(temp);//chỉ thị tích cực
        res.push_back(temp);
    }

    for (int i = 1; i < res.size(); i++) {
        toupper(res[i][0]);
        cout << res[i];
        if (i != res.size() - 1) {
            cout << " ";
        } else {
            cout << ", ";
        }
    }
    for (int i = 0; i < res[0].size(); i++) {
        toupper(res[0][i]);
    }
    cout << res[0];

}
```



Hình 8: Lưu đồ thuật toán bài 8

6. Đánh giá độ phức tạp thuật toán: O(t * n) với n là độ dài xâu đầu vào

9 VÒNG TRÒN

1. Xác định input, output:

   - Input: một xâu có đúng 52 kí tự in hoa. Mỗi kí tự xuất hiện đúng 2 lần.
   - Output: số giao điểm khi nối các kí tự giống nhau

2. Phân tích bài toán: Ta giả sử có 4 điểm A1, A2, B1, B2. Xét cặp cạnh A1A2, B1B2. Ta thấy khi đi theo vòng tròn nếu A1 xuất hiện trước B1 và A2 thì A2 phải xuất hiện trước B2 mới có cặp cạnh cắt nhau. Để kiểm soát thứ tự xuất hiện, ta cần có 2 mảng để lưu trữ chỉ số

3. Vẽ lưu đồ: Lưu đồ được thể hiện ở hình 9

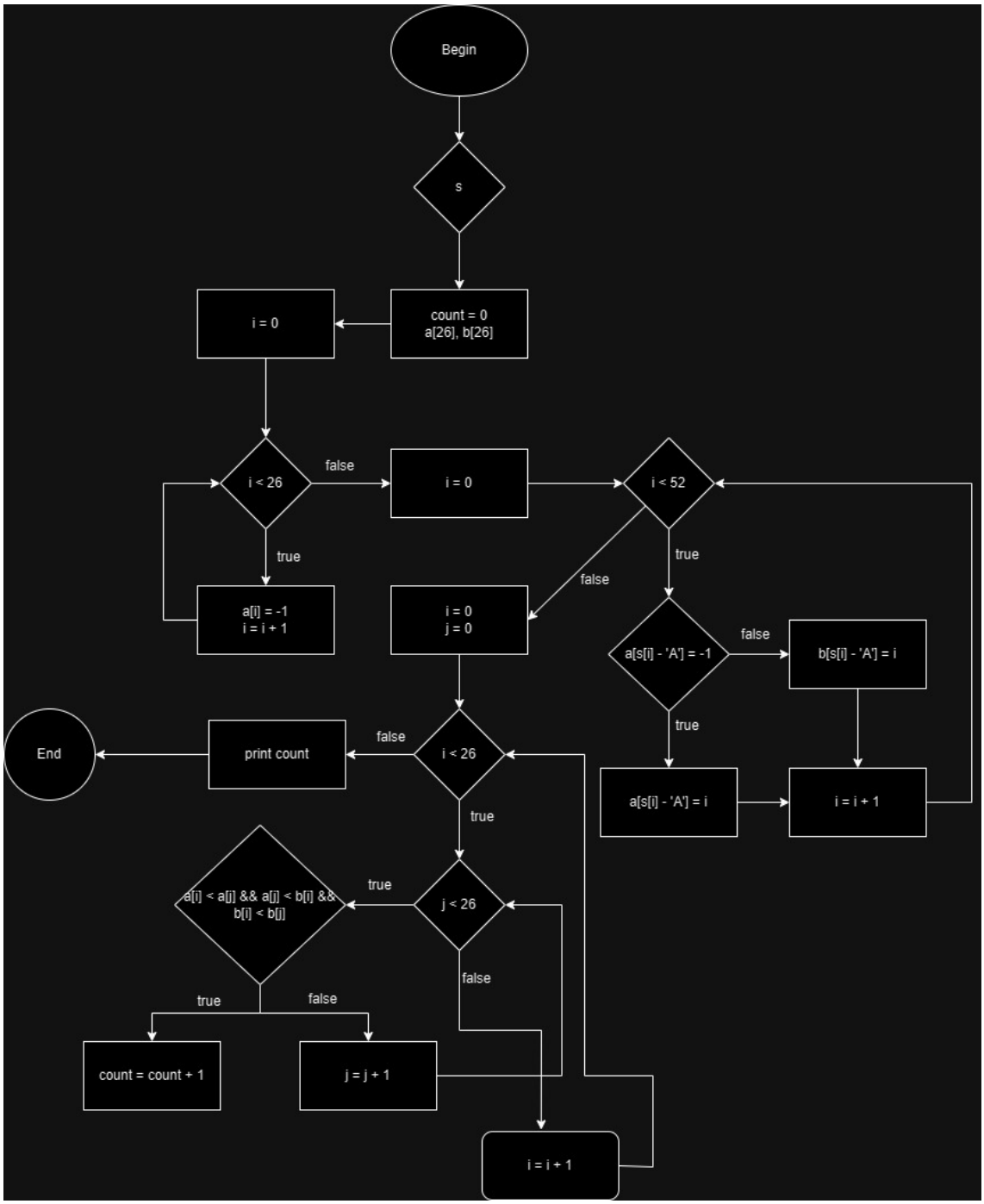
4. Chạy từng bước

5. Viết code: Code được viết như sau:

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    string s;
    cin >> s;
    int count  = 0;
    int a[26], b[26];
    for (int i = 0; i < 26; i++) {
        a[i] = -1;
    }
    for (int i = 0; i < 52; i++) {
        if (a[s[i] - 'A'] == -1) { // first occurence
            a[s[i] - 'A'] = i;
        } else {
            b[s[i] - 'A'] = i; //second occurence
        }
    }

    for(int i = 0; i < 26; i++){
        for(int j = 0; j < 26; j++){
            if(a[i] < a[j] && a[j] < b[i] && b[i] < b[j]){
                count++;//chỉ thị tích cực
            }
        }
    }
    cout << count;
}
```

6. Đánh giá độ phức tạp thuật toán: $O(t * 52 + 26^2) = O(t)$



Hình 9: Lưu đồ thuật toán bài 9

10 CHIA HẾT CHO 2

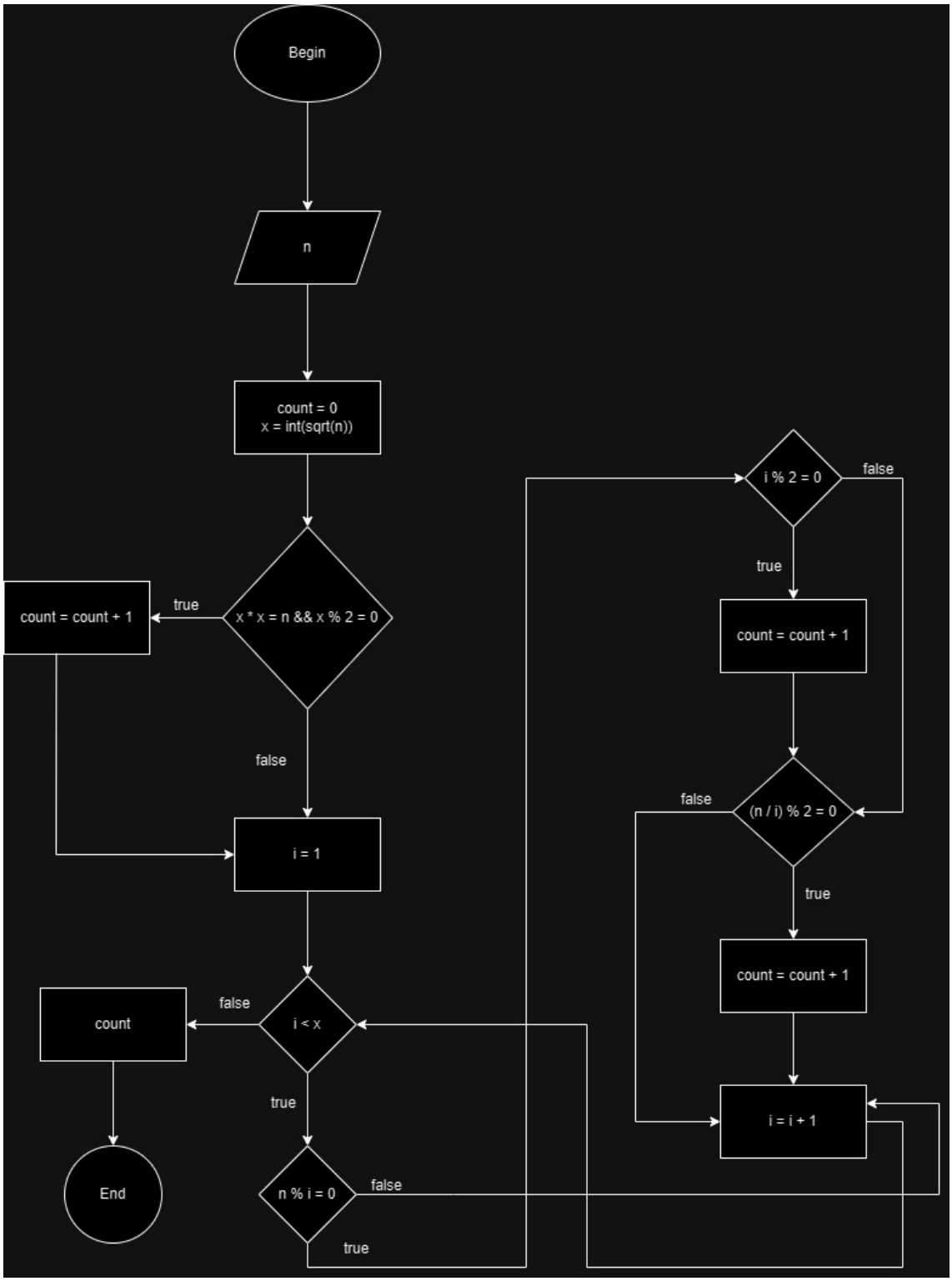
1. Xác định input, output:
   - Input: Số nguyên dương n có tối đa 9 chữ số
   - Output: Số ước số của n chia hết cho 2
2. Phân tích bài toán: Ta thấy ứng với mỗi ước x của n sẽ luôn có ước n/x tương ứng, ngoại trừ trường hợp với n là số chính phương thì x và n/x là một. Do đó, để duyệt các ước, ta sẽ duyệt từ 1 đến $\sqrt{n}$ và kiểm tra, ta

cũng cần có biến đếm để kiểm soát kết quả

3. Vẽ lưu đồ: Lưu đồ được thể hiện ở hình 10
4. Chạy từng bước
5. Viết code: Code được viết như sau:

```
#include <bits/stdc++.h>
using namespace std;
void testCase();
int main () {
    int t;
    cin >> t;
    while (t--) {
        testCase();
        cout << endl;
    }
    return 0;
}
void testCase() {
    int n;
    cin >> n;
    int count = 0;
    int x = sqrt(n);
    if (x * x == n && x % 2 == 0) {
        count++;
    }
    for (int i = 1; i < sqrt(n); i++) {
        if (n % i == 0) {
            if (i % 2 == 0) {
                count++;
            }//chỉ thị tích cực
            if ((n / i) % 2 == 0) {
                count++;
            }
        }
    }
    cout << count;
}
```

6. Đánh giá độ phức tạp thuật toán: O(t * $\sqrt{n}$)



Hình 10: Lưu đồ thuật toán bài 10