Bài 1. Xâu nhị phân kế tiếp

Bước 1. Xác định input, output

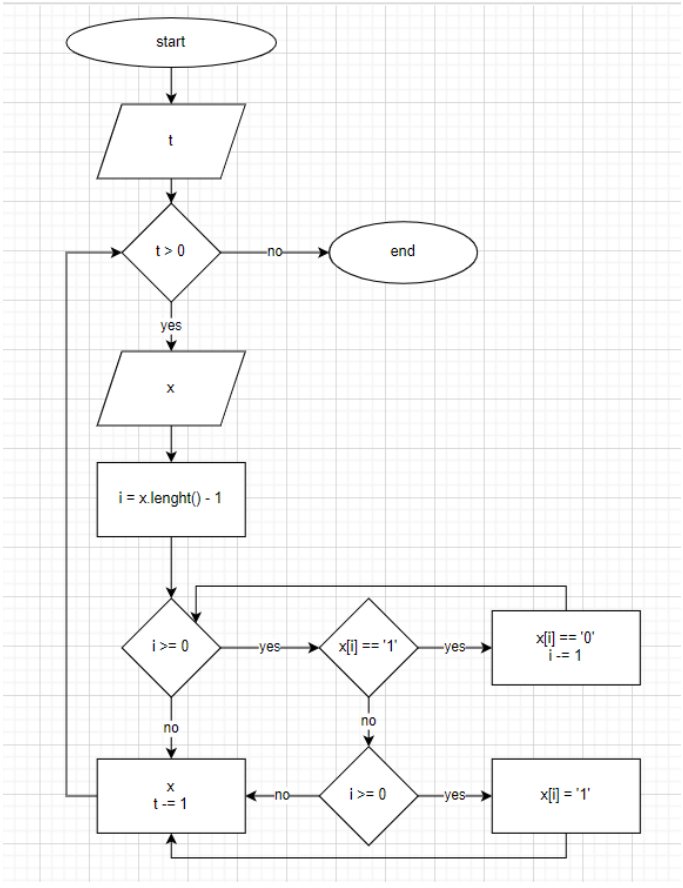
- Input: số bộ test – t, xâu nhị phân – x.
- Output: xâu nhị phân kế tiếp của x.

Bước 2. Phân tích bài toán

- Để tìm ra số nhị phân kế tiếp ta cần đem số nhị phân hiện tại cộng thêm 1 đơn vị. Phép cộng một số nhị phân với 1 sẽ tiến hành cộng lần lượt các bit từ phải sang trái của số nhị phân đó với bit 1.
- Tính chất cộng các bit nhị phân:
  - 1 + 1 = 0 nhớ 1.
  - 0 + 1 = 1.

⇨ Tìm vị trí bit 0 đầu tiên từ phải qua của số nhị phân x, chuyển nó thành bit 1 và tất cả các bit 1 phía sau vị trí đó chuyển thành bit 0.
Nếu tất cả các bit đều là 1 (không tìm thấy vị trí của bit 0) thì không có bước chuyển bit 0 thành 1.

Bước 3. Vẽ lưu đồ



Bước 4. Chạy thử trên lưu đồ

Intput t = 2

-> t = 2 thỏa mãn > 0.

-> nhập x = 010101.

-> i = 5.

-> i = 5 thỏa mãn >= 0.

-> x[5] thỏa mãn == ‘1’.

-> x[5] = ‘0’, i = 4.

-> i = 4 thỏa mãn >= 0.

-> x[4] không thỏa mãn == ‘1’.

-> i = 4 thỏa mãn >= 0.

-> x[4] = ‘1’

-> output x = ‘010110’, t = 1.

-> t = 1 thỏa mãn > 0.

-> nhập x = 11.

-> i = 1.

-> i = 1 thỏa mãn >= 0.

-> x[1] thỏa mãn == ‘1’.

-> x[1] = ‘0’, i = 0.

-> i = 0 thỏa mãn >= 0.

-> x[0] thỏa mãn == ‘1’.

-> x[0] = ‘0’, i = -1.

-> i = -1 không thỏa mãn >= 0.

-> ouput x = ‘00’, t = 0.

-> t = 0 không thỏa mãn > 0.

-> end.

Bước 5. Code

```
#include <iostream>

using namespace std;

int main() {
	int t;
	cin >> t;
	while(t--) {
		string s;
		cin >> s;
		int i = s.length() - 1;
		while(s[i] == '1' && i >= 0) {
			s[i] = '0';
			i -= 1;
		}
		s[i] = '1';
		cout << s << endl;
	}
	return 0;
}
```

Bài 2. Tập con kế tiếp

Bước 1. Xác định input, output

- Input:
  - Số bộ test: t.
  - Mỗi test gồm: n, k, x.
- Output: tập con kế tiếp của x.

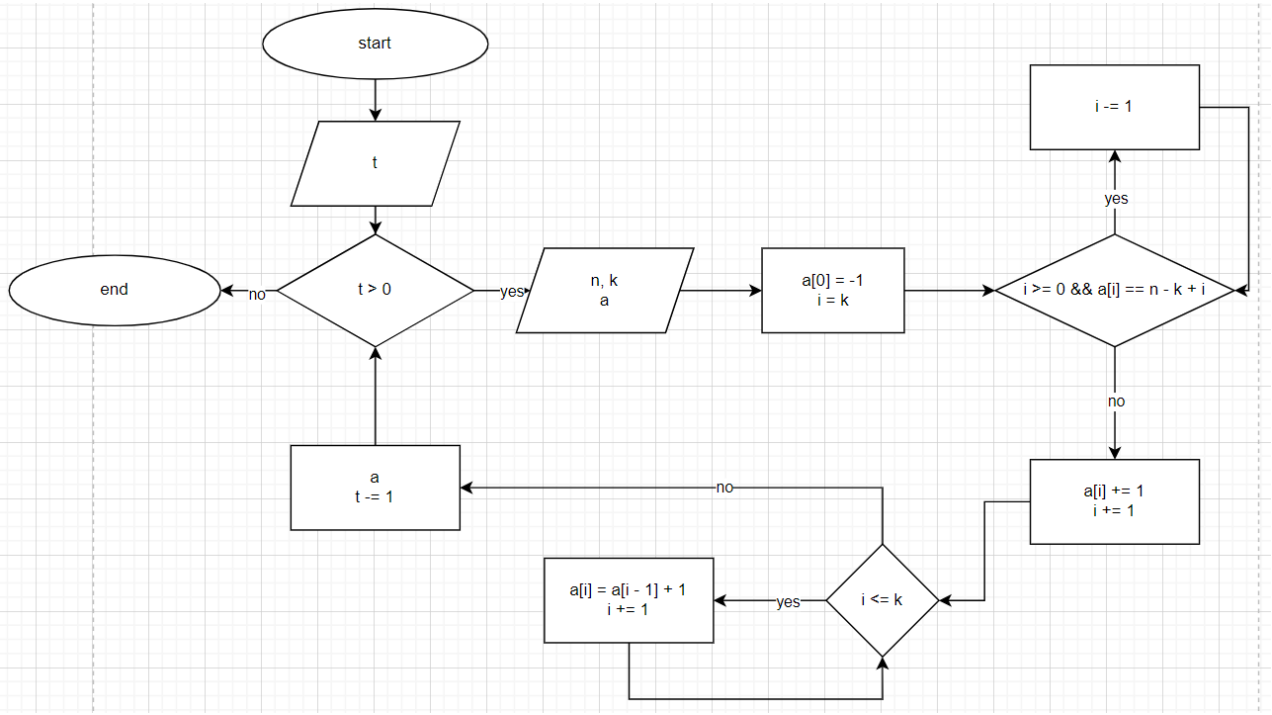
Bước 2. Phân tích

Ví dụ với n = 5, k = 3.

- X_i = {1, 4, 5}.
  ➔ X_{i+1} = {2, 3, 4}.
- X_i = {1, 3, 4}.
  ➔ X_{i+1} = {1, 3, 5}.
- Ta dễ dàng nhận thấy tại mỗi vị trí i: x[i – 1] < x[i] <= n – k + i với 1 <= i <= k.

⇨ Để tìm tập con kế tiếp của x, ta cần tìm vị trí i đầu tiên từ bên phải sang của tập x sao cho giá trị x[i] chưa đạt giá trị max = n – k + i. Sau đó tăng giá trị x[i] lên 1 đơn vị và từ vị trí i + 1 đến k thay x[i] = x[i – 1] + 1.

Bước 3. Lưu đồ



Bước 4. Chạy thử

Bước 5. Code

```
#include <iostream>
using namespace std;
int main() {
	int t; cin >> t;
	while(t--) {
		int a[1001];
		a[0] = -1;
		int n, k; cin >> n >> k;
		for(int i = 1; i <= k; i++)
			cin >> a[i];
		int i = k;
		while(a[i] == n - k + i && i > 0) {
			i -= 1;
		}
		a[i] += 1;
		i += 1;
		for(; i <= k; i++) {
			a[i] = a[i - 1] + 1;
		}
		for(i = 1; i <= k; i++)
			cout << a[i] << " ";
		cout << endl;
	}
	return 0;
}
```

Bài 3. Mảng đối xứng

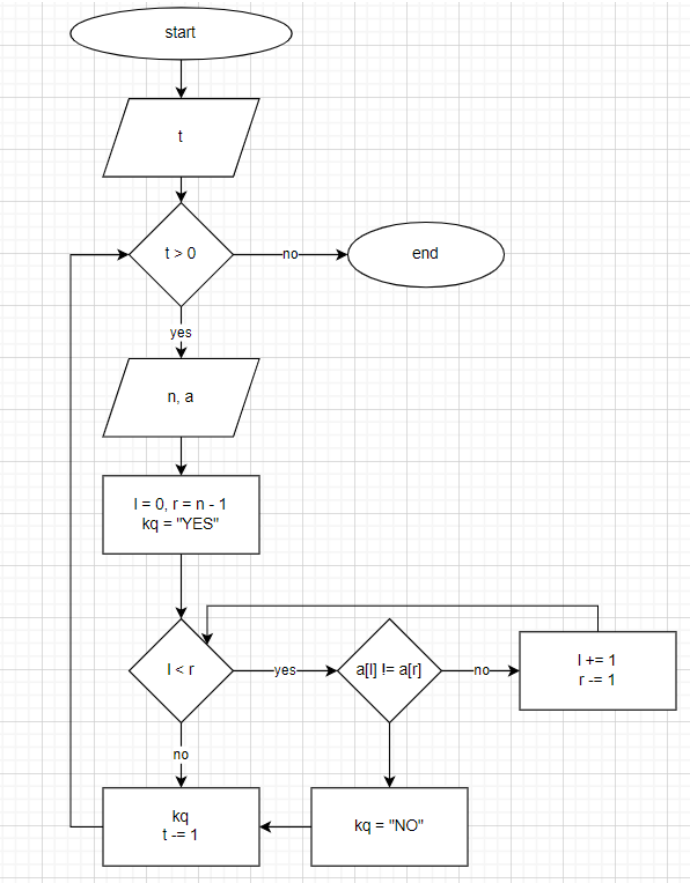
Bước 1. Xác định input, output

- Input:
  - Số bộ test: t.
  - Mỗi bộ test gồm: n, mảng a.
- Output:
  - YES – đối xứng.
  - NO – bất đối xứng.

Bước 2. Phân tích bài toán

- Kiểm tra các cặp số tại các vị trí đối xứng có bằng nhau hay không:
  - Vị trí 0 với n – 1.
  - Vị trí 1 với n – 2.
  - …
  - Vị trí i với n – i – 1.

Bước 3. Lưu đồ



Bước 4. Chạy thử

Bước 5. Code

```
#include <iostream>
using namespace std;
int main() {
	int t; cin >> t;
	while(t--) {
		int n;  cin >> n;
		int a[n];
		for(int i = 0; i < n; i++)
			cin >> a[i];
		int l = 0, r = n - 1;
		string kq = "YES";
		while(l < r) {
			if(a[l] != a[r]) {
				kq = "NO";
				break;
			}
			l += 1;
			r -= 1;
		}
		cout << kq << endl;
	}
	return 0;
}
```

Bài 4. Phân tích thừa số nguyên tố

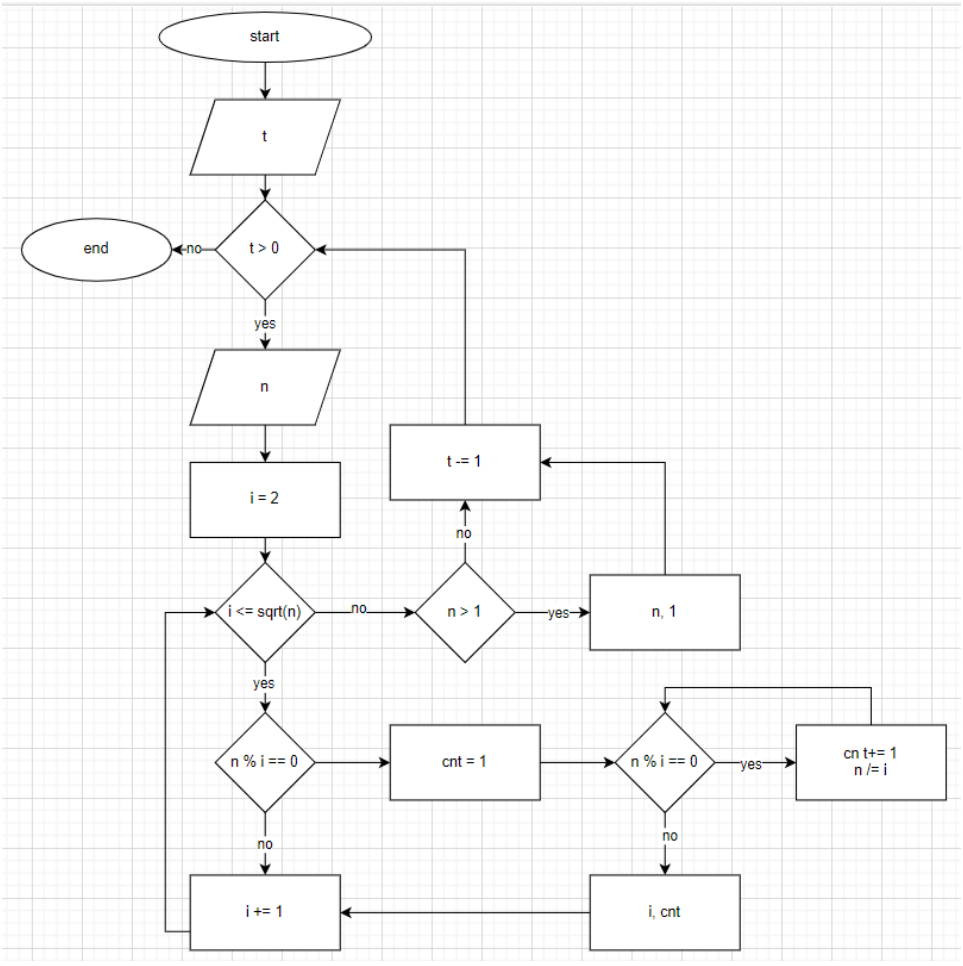
Bước 1. Xác định input, output

- Input:
  - Số bộ test: t.
  - Mỗi bột test: n.
- Output: Mỗi bộ test viết ra thứ tự bộ test, sau đó lần lượt là các số nguyên tố khác nhau có trong tích, với mỗi số viết thêm số lượng số đó.
  VD: Test 1: 2(2) 3(1) 5(1)

Bước 2. Phân tích

- Để phân tích thừa số nguyên tố của n, ta kiểm tra trong [2, sqrt(n)] các giá trị i mà n chia hết nếu n chia hết cho i ta sẽ dùng n chia cho i sau đó n mới sẽ có giá trị = thương của phép chia và thực hiện lặp lại cho tới khi n không chia hết cho i nữa mỗi lần chia sẽ tăng biến đếm lên 1.

Bước 3. Lưu đồ



Bước 4. Chạy thử

Bước 5. Code

```
#include <iostream>
#include <cmath>
using namespace std;
int main() {
	int t; cin >> t;
	for(int test = 1; test <= t; test++) {
		int n; cin >> n;
		cout << "Test " << test << ": ";
		for(int i = 2; i <= sqrt(n); i++) {
			if(n % i == 0) {
				int cnt = 0;
				while(n % i == 0) {
					n /= i;
					cnt += 1;
				}
				cout << i << "(" << cnt << ") ";
			}
		}
		if(n > 1) cout << n << "(1)";
		cout << endl;
	}
	return 0;
}
```

Bài 5. Phép cộng

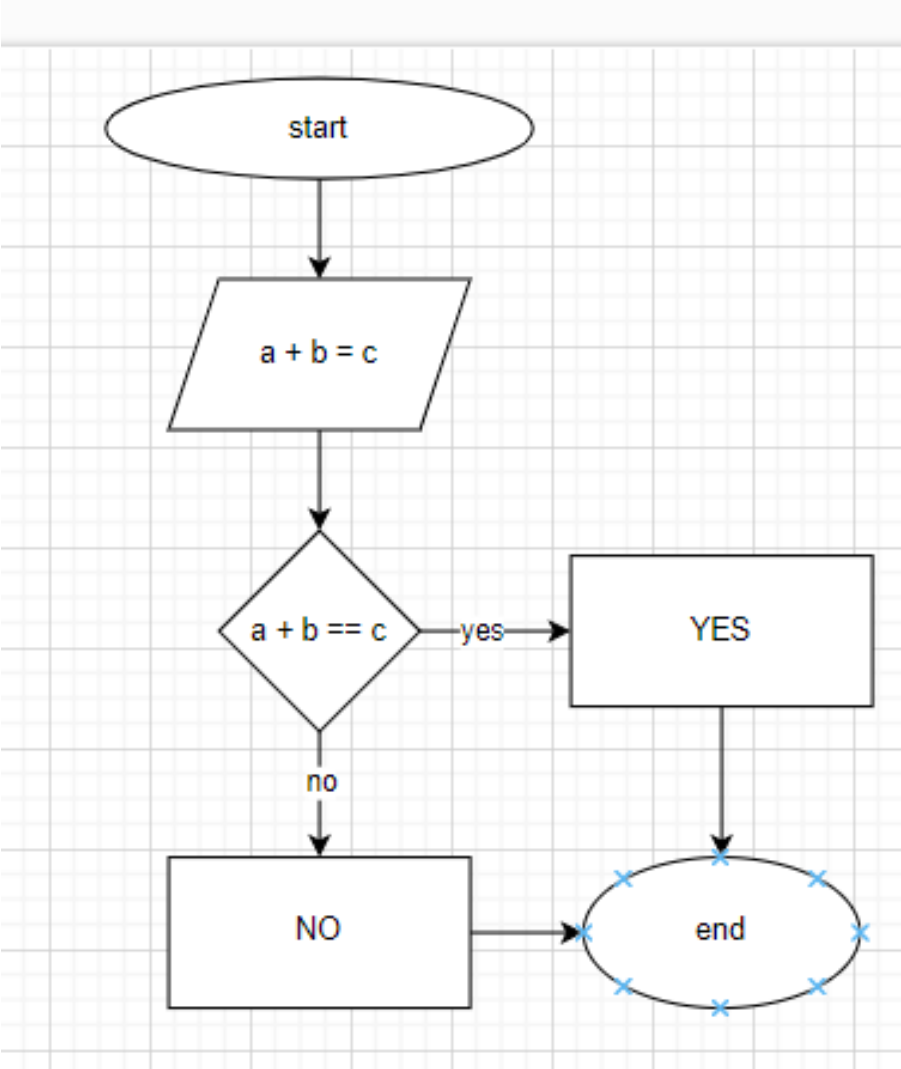
Bước 1. Xác định input, output

- Input: dạng a + b = c.
- Output:
  - YES nếu a + b = c.
  - NO nếu a + b != c.

Bước 2. Phân tích

- Nhập vào a sau đó nhập vào dấu +, nhập vào b, nhập vào dấu =, nhập vào c.
- Kiểm tra a + b có bằng c hay không và in ra kết quả.

Bước 3. Lưu đồ



Bước 4. Chạy thử

Bước 5. Code

```
#include <iostream>
using namespace std;
int main() {
	int a, b, c;
	char x;
	cin >> a >> x >> b >> x >> c;
	if(a + b == c) cout << "YES";
	else cout << "NO";
	return 0;
}
```

Bài 6. Số tăng giảm

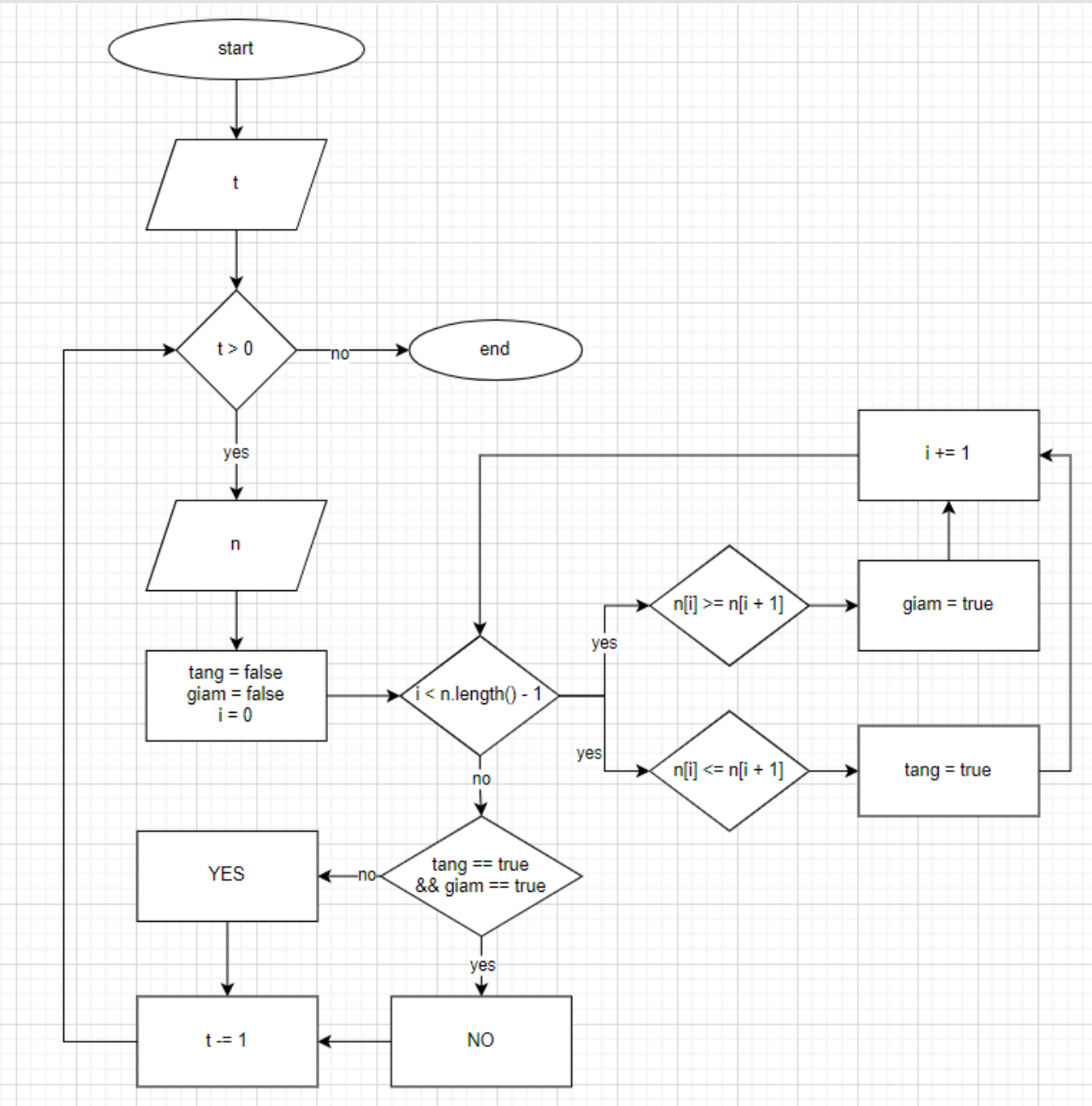
Bước 1. Xác định input, output

- Input:
  - Số bộ test: t.
  - Mỗi test gồm 1 string n.
- Ouput:
  - YES nếu n thỏa mãn.
  - NO nếu ngược lại.

Bước 2. Phân tích

- Kiểm tra từng cặp chữ số tại vị trí i và i + 1, nếu n[i] <= n[i + 1] thì đó lại xâu tăng, nếu n[i] >= n[i + 1] thì đó là xâu giảm.
- Nếu xâu đó vừa tăng vừa giảm thì không thỏa mãn.

Bước 3. Lưu đồ



Bước 4. Chạy thử

Bước 5. Code

```
#include <iostream>

using namespace std;

int main() {
	int t;
	cin >> t;
	while(t--) {
		string n;
		cin >> n;
		bool tang = false, giam = false;
		for(int i = 0; i < n.length() - 1; i++) {
			if(n[i] <= n[i + 1]) {
				tang = true;
			}
			if(n[i] >= n[i + 1]) {
				giam = true;
			}
		}
		if(tang && giam) cout << "NO" << endl;
		else cout << "YES" << endl;
	}
	return 0;
}
```