**Đề số BB(ko có mã đề)-14**

Câu 1: Nêu ý tưởng về thuật toán sắp xếp (counting sort) và minh họa A={1,4,1,2,7,5,2)

Counting Sort là một thuật toán sắp xếp không dựa trên so sánh. Thay vì so sánh các phần tử, thuật toán sử dụng một mảng đếm để xác định số lần xuất hiện của từng giá trị, từ đó sắp xếp các phần tử.

Để đơn giản, giả sử dữ liệu trong khoảng 0 đến 9 ◦

Dữ liệu đầu vào: 1, 4, 1, 2, 7, 5, 2

Bước 1: Đếm số lần xuất hiện của từng phần tử trong mảng cần sắp

Index: 0 1 2 3 4 5 6 7 8 9

Count: 0 2 2 0 1 1 0 1 0 0

Bước 2: Sửa mảng count sao cho chỉ số của mỗi phần tử lưu tổng số lần đếm trước đó → vị trí của chúng trong mảng đầu ra

Index: 0 1 2 3 4 5 6 7 8 9

Count: 0 2 4 4 5 6 6 7 7 7

Bước 3: Xuất từng phần tử trong chuỗi đầu vào tại ví trí là tổng số lần đếm trừ đi 1 và trừ số lần đếm đi 1. Xử lý dữ liệu đầu vào: 1, 4, 1, 2, 7, 5, 2. Vị trí của 1 là (2-1) = 1 thì đặt số 1 vào chỉ số 1 trong mảng đầu ra. Gặp số 1 tiếp theo thì số lần đếm lúc này là 1 và giảm đi thành 0 và đặt số 1 vào chỉ số 0

Câu 2: Trạm xăng

#include <iostream>

#include <queue>

using namespace std;

// Hàm tìm điểm xuất phát

int canCompleteCircuit(queue<pair<int, int>>& stations) {

int total\_fuel = 0; // Tổng chênh lệch xăng và quãng đường

int current\_fuel = 0; // Nhiên liệu tích lũy hiện tại

int start\_station = 0; // Điểm xuất phát

int n = stations.size(); // Số lượng trạm xăng

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

pair<int, int> station = stations.front(); // Lấy trạm hiện tại

stations.pop(); // Xóa trạm khỏi hàng đợi

stations.push(station); // Đưa lại trạm vào cuối hàng đợi

int fuel = station.first;

int distance = station.second;

total\_fuel += fuel - distance;

current\_fuel += fuel - distance;

// Nếu nhiên liệu tích lũy âm, chọn trạm tiếp theo làm điểm xuất phát

if (current\_fuel < 0) {

start\_station = i + 1;

current\_fuel = 0; // Reset nhiên liệu tích lũy

}

}

// Nếu tổng chênh lệch xăng >= 0, hành trình có thể hoàn thành

return (total\_fuel >= 0) ? start\_station : -1;

}

int main() {

// Dữ liệu đầu vào (xăng tại trạm, khoảng cách tới trạm tiếp theo)

queue<pair<int, int>> stations;

stations.push({4, 6}); // Trạm 1

stations.push({6, 5}); // Trạm 2

stations.push({7, 3}); // Trạm 3

stations.push({4, 5}); // Trạm 4

// Gọi hàm giải bài toán

int result = canCompleteCircuit(stations);

// Xuất kết quả

if (result != -1) {

cout << "Xe có thể bắt đầu tại trạm xăng số " << result << " để hoàn thành hành trình." << endl;

} else {

cout << "Không có điểm xuất phát nào để hoàn thành hành trình." << endl;

}

return 0;

}

1. viết chương trình
2. Minh họa {4,6},{6,5},{7,3} và {4,5}

### **Minh họa từng bước**

#### **Khởi tạo**

* Hàng đợi (queue): {(4, 6), (6, 5), (7, 3), (4, 5)}
* total\_fuel = 0, current\_fuel = 0, start\_station = 0.

#### **Bước 1: Duyệt qua trạm 1**

* Trạm: (4, 6) → Chênh lệch: 4 - 6 = -2.
* total\_fuel = -2, current\_fuel = -2.

Nhiên liệu tích lũy âm (current\_fuel < 0), không thể xuất phát từ trạm 1. Đặt start\_station = 1 (trạm kế tiếp), reset current\_fuel = 0.

#### **Bước 2: Duyệt qua trạm 2**

* Trạm: (6, 5) → Chênh lệch: 6 - 5 = +1.
* total\_fuel = -1, current\_fuel = +1.

Nhiên liệu tích lũy dương, tiếp tục duyệt trạm tiếp theo.

#### **Bước 3: Duyệt qua trạm 3**

* Trạm: (7, 3) → Chênh lệch: 7 - 3 = +4.
* total\_fuel = +3, current\_fuel = +5.

Nhiên liệu tích lũy dương, tiếp tục duyệt trạm tiếp theo.

#### **Bước 4: Duyệt qua trạm 4**

* Trạm: (4, 5) → Chênh lệch: 4 - 5 = -1.
* total\_fuel = +2, current\_fuel = +4.

Nhiên liệu tích lũy vẫn dương, hoàn tất vòng duyệt.

#### **Kết luận**

* **Tổng nhiên liệu (**total\_fuel**) = +2 > 0**: Có thể hoàn thành hành trình.
* Điểm xuất phát phù hợp là **trạm 2 (start\_station = 1, tính từ 0-based)**.

### **Kết quả cuối cùng**

Xe có thể bắt đầu từ **trạm số 2** để hoàn thành hành trình vòng tròn.

Câu 3:Số siêu nguyên với n=30 thì siêu nguyên tố là {2,3,5,7,23,29}

Cho một số n, hãy đưa số dãy số siêu nguyên tố nhỏ hơn hoặc bằng n đã được sắp xếp tăng

dần.

Ví dụ:

• Với n = 30; thì superPrimeNumber(n) = [2, 3, 5, 7, 23, 29];

Vì các số 2, 3, 5, 7, 23 và 29 đều là số siêu nguyên tố và nhỏ hơn hoặc bằng 30.

Minh họa : Dùng phương pháp sinh, nếu x đã là số siêu nguyên tố thì ta sẽ lần lượt thêm các số tử 1 đến 9 vào cuối x (x\*10 + i), rồi kiểm tra xem nó có còn là số siêu nguyên tố hay không. Nếu là số nguyên tố thì lưu nó vào **queue.**

#include<iostream>

#include<queue>

#include<math.h>

using namespace std;

bool isPrime(int n){

if (n<2) return false;

for (int i=2; i<=sqrt(n); i++)

if (n%i==0) return false;

return true;

}

int main(){

queue<int> q;

int n;

cin >> n;

for (int i = 2; i <= n, i < 10; i++){

if (isPrime(i)){

q.push(i);

}

}

while (!q.empty()){

for (int i = 1; i <= 9; i++){

int k = q.front()\*10 + i;

if ( k <= n && isPrime(k)){

q.push(q.front()\*10 + i);

}

}

cout << q.front() << " ";

q.pop();

}

return 0;

}