Câu 1:Viết chương trình tìm mốt và trung vị = counting sort. Minh họa A={1,4,1,2,7,1,2,5,3,6}

Tìm mốt và trung vị của một mảng số sử dụng phương pháp sắp xếp đếm phân phối(counting sort). Minh họa các bước tìm mốt với mảng số A = {1, 4, 1, 2, 7, 1, 2, 5, 3, 6}.  
Chú ý: mốt là số có tần suất lớn nhất trong mảng. Với một mảng các số được sắp theothứ tự tăng dần, trung vị là số chính giữa mảng nếu mảng có số lẻ phần tử, là trungbình của hai số chính giữa mảng nếu mảng có số chẵn phần tử.

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

void countingSortAndFindModeMedian(const vector<int>& A) {

// Bước 1: Tìm giá trị lớn nhất trong mảng

int maxVal = \*max\_element(A.begin(), A.end());

// Bước 2: Đếm tần suất của từng giá trị

vector<int> count(maxVal + 1, 0);

for (int num : A) {

count[num]++;

}

// Bước 3: Tìm mode (số có tần suất lớn nhất)

int mode = 0, maxFrequency = 0;

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

if (count[i] > maxFrequency) {

maxFrequency = count[i];

mode = i;

}

}

// Bước 4: Sắp xếp mảng bằng Counting Sort

vector<int> sortedArray;

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

for (int j = 0; j < count[i]; j++) {

sortedArray.push\_back(i);

}

}

// Bước 5: Tìm trung vị (median)

int n = sortedArray.size();

double median = 0.0;

if (n % 2 == 1) {

// Nếu số lượng phần tử lẻ

median = sortedArray[n / 2];

} else {

// Nếu số lượng phần tử chẵn

median = (sortedArray[n / 2 - 1] + sortedArray[n / 2]) / 2.0;

}

// Kết quả

cout << "Mảng đã sắp xếp: ";

for (int num : sortedArray) {

cout << num << " ";

}

cout << endl;

cout << "Mode (mốt): " << mode << endl;

cout << "Median (trung vị): " << median << endl;

}

int main() {

vector<int> A = {1, 4, 1, 2, 7, 1, 2, 5, 3, 6};

countingSortAndFindModeMedian(A);

return 0;

}

Minh họa

A = {1, 4, 1, 2, 7, 1, 2, 5, 3, 6}

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

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

Count: 0 3 2 1 1 1 1 1 0 0

Bước2: 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 3 5 6 7 8 9 10 10 10

Bước3: 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.

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

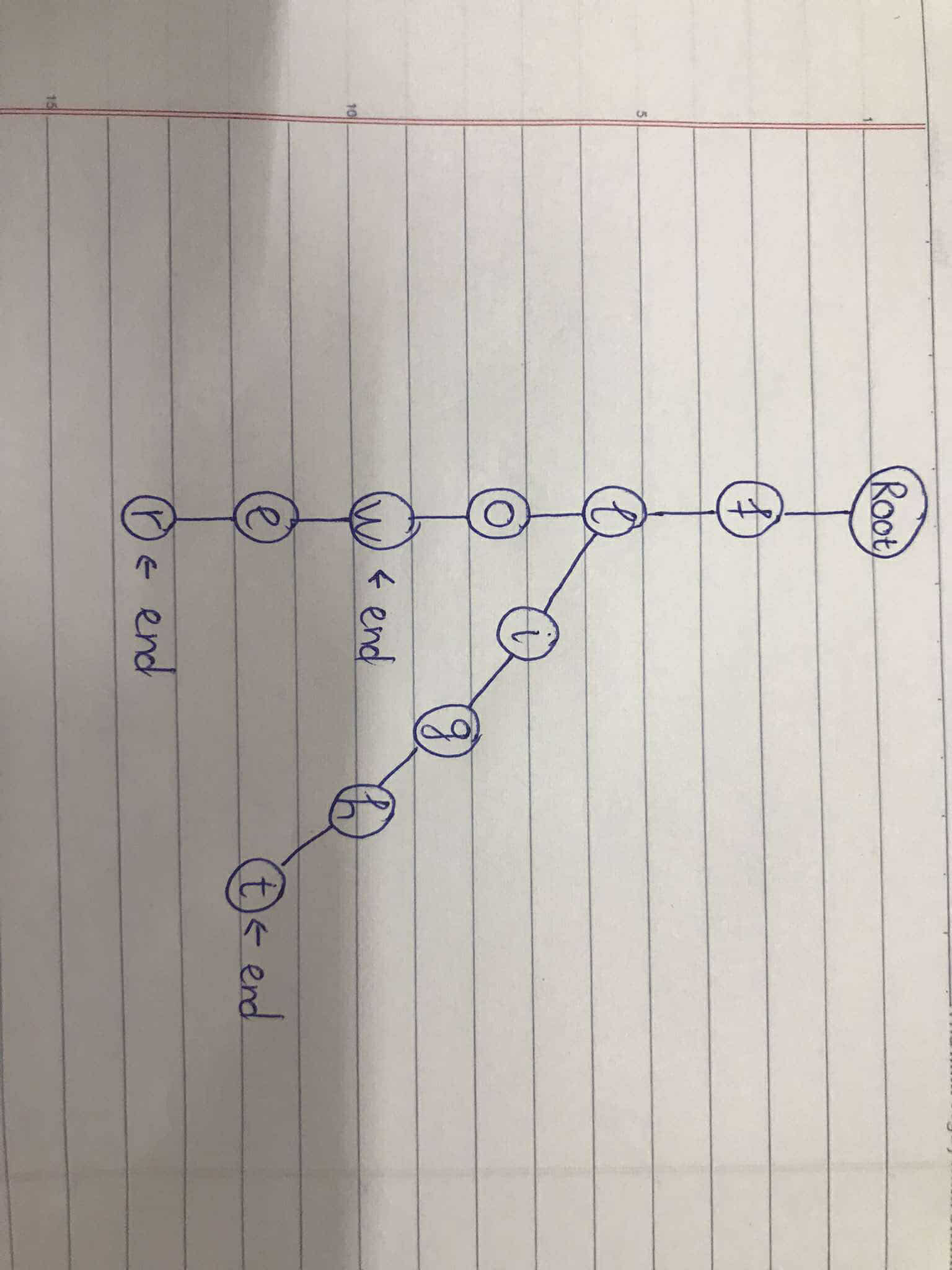
Count: 1 1 1 2 2 3 4 5 6 7

### ****Bước 2: Tìm Mode (Mốt)****

1. **Mode (Mốt)** là số có tần suất lớn nhất trong mảng count.
2. **Duyệt qua mảng** count **để tìm giá trị lớn nhất:**
   1. count[1] = 3 (lớn nhất).
   2. Mode = **1**.

Câu2:Xâu kí tự s=[]{“flower” , “flow”, ”flight”}

\* bài cây tiền tố flower, flow, flight



#include <iostream>

#include <unordered\_map>

#include <vector>

#include <string>

using namespace std;

// Định nghĩa một nút trong Trie

struct TrieNode {

unordered\_map<char, TrieNode\*> children; // Các con của nút

bool isEndOfWord; // Đánh dấu đây là điểm kết thúc của một từ

TrieNode() {

isEndOfWord = false;

}

};

class Trie {

private:

TrieNode\* root;

public:

Trie() {

root = new TrieNode(); // Khởi tạo gốc Trie

}

// Hàm chèn một từ vào Trie

void insert(const string& word) {

TrieNode\* current = root;

for (char ch : word) {

if (current->children.find(ch) == current->children.end()) {

current->children[ch] = new TrieNode();

}

current = current->children[ch];

}

current->isEndOfWord = true; // Đánh dấu kết thúc từ

}

// Hàm tìm tiền tố chung dài nhất

string findLongestCommonPrefix() {

string prefix = "";

TrieNode\* current = root;

// Duyệt xuống cây Trie đến khi không thể tiếp tục

while (current && current->children.size() == 1 && !current->isEndOfWord) {

auto it = current->children.begin(); // Lấy nút con duy nhất

prefix += it->first; // Thêm ký tự vào tiền tố

current = it->second; // Tiếp tục duyệt nút con

}

return prefix;

}

};

int main() {

// Test case

vector<string> words = {"flower", "flow", "flight"};

Trie trie;

// Chèn từng từ vào Trie

for (const string& word : words) {

trie.insert(word);

}

// Tìm tiền tố chung dài nhất

string lcp = trie.findLongestCommonPrefix();

// In kết quả

cout << "Tiền tố chung dài nhất: " << lcp << endl;

return 0;

}

Câu3: Đứa ra số lớn nhất trong số có tần số bé nhất.Minh họa 2,2,4,4,7,7,7.

Câu : Tìm số lớn nhất trong những số có tần suất nhỏ nhất

#include <iostream>

#include <unordered\_map>

#include <vector>

#include <algorithm>

using namespace std;

// Hàm tìm số lớn nhất trong những số có tần suất nhỏ nhất

int findMaxWithMinFrequency(const vector<int>& A) {

unordered\_map<int, int> frequency; // Lưu tần suất xuất hiện của các số

// Đếm tần suất

for (int num : A) {

frequency[num]++;

}

// Tìm tần suất nhỏ nhất

int minFrequency = INT\_MAX;

for (const auto& pair : frequency) {

minFrequency = min(minFrequency, pair.second);

}

// Tìm số lớn nhất trong những số có tần suất nhỏ nhất

int result = INT\_MIN;

for (const auto& pair : frequency) {

if (pair.second == minFrequency) {

result = max(result, pair.first);

}

}

return result;

}

int main() {

// Dãy số đầu vào

vector<int> A = {2, 2, 4, 4, 7, 7, 7};

// Tìm kết quả

int result = findMaxWithMinFrequency(A);

// In kết quả

cout << "Số lớn nhất trong những số có tần suất nhỏ nhất là: " << result << endl;

return 0;

}

Ví dụ minh họa

Dãy số:A = {2, 2, 4, 4, 7, 7, 7}

Khởi tạo bảng tần suất (unordered\_map) rỗng:frequency = {}

Duyệt qua từng phần tử trong mảng A và tăng giá trị tương ứng trong bảng frequency:

| **Phần tử hiện tại (A[i])** | **Cập nhật bảng** frequency | **Ghi chú** |
| --- | --- | --- |
| 2 | {2: 1} | Số 2 xuất hiện **1 lần** |
| 2 | {2: 2} | Số 2 xuất hiện **2 lần** |
| 4 | {2: 2, 4: 1} | Số 4 xuất hiện **1 lần** |
| 4 | {2: 2, 4: 2} | Số 4 xuất hiện **2 lần** |
| 7 | {2: 2, 4: 2, 7: 1} | Số 7 xuất hiện **1 lần** |
| 7 | {2: 2, 4: 2, 7: 2} | Số 7 xuất hiện **2 lần** |
| 7 | {2: 2, 4: 2, 7: 3} | Số 7 xuất hiện **3 lần** |

frequency = {2: 2, 4: 2, 7: 3}

### ****Tóm tắt từng bước trong bảng****

| **Bước** | **Kết quả** | **Ghi chú** |
| --- | --- | --- |
| 1. Đếm tần suất | {2: 2, 4: 2, 7: 3} | Tạo bảng tần suất từ dãy số |
| 2. Tìm tần suất nhỏ nhất | 2 | Nhỏ nhất là 2, thuộc về số 2 và 4 |
| 3. Tìm số lớn nhất | 4 | Trong các số có tần suất nhỏ nhất, 4 là lớn nhất |

Số lớn nhất trong những số có tần suất nhỏ nhất là: 4