**Bài tập chương 2**

**Bài 1. Cho một vector chứa các số nguyên. Hãy đưa ra số lượng phần tử khác nhau trong vector đó.**

#include <bits/stdc++.h>

using namespace std;

int distinctCount(vector<int> v){

unordered\_set<int> s(v.begin(), v.end());

return s.size();

}

main(){

vector<int> inputVector { 1, 3, 3, 2, 5, 2, 2, 5, 1 };

cout << distinctCount(inputVector);

}

**Bài 2. Anh Việt đang thống kê số liệu cho công ty, anh muốn giá trị nhỏ thứ hai ở trong một dãy số nguyên. Hãy giúp anh Việt tìm ra giá trị đó, nếu không có kết quả như yêu cầu thì trả về "NO".**

#include <bits/stdc++.h>

using namespace std;

string secRank(vector<int> v) {

set <int> s(v.begin(), v.end());

return (v.size() < 2 || s.size() == 1) ? "NO" : to\_string(\*(++s.begin()));

}

main(){

vector<int> inputVector { 1, 3, 3, 2, 2 };

cout << secRank(inputVector);

}

**Bài 3. Tính số lượng tối thiểu các ký tự cần thay đổi trong chuỗi str để chuỗi đó có k ký tự khác nhau hoặc xuất ra điều đó là không thể. Chuỗi str chỉ bao gồm những chữ cái latinh viết thường và cũng chỉ có thể thay thành những chữ cái latinh viết thường.**

string diversity(string str, int k) {

if (str.size() < k) return "impossible";

unordered\_set <char> s(str.begin(), str.end());

return to\_string(k - s.size());

}

main() {

vector <pair<string, int>> input { { "yandex", 6 }, { "google", 7 }, { "codelearn", 9 } };

for (auto x : input) {

cout << "diversity(" << x.first << ", " << x.second << ") = " << diversity(x.first, x.second) << endl;

}

}

**Bài 4. An có một danh sách các sản phẩm trong siêu thị nhưng thật không may trong danh sách** **lại có những sản phẩm xuất hiện nhiều lần. Bạn hãy giúp An làm lại danh sách sao cho một sản phẩm chỉ xuất hiện một lần trong danh sách và các sản phẩm xuất hiện theo thứ tăng dần trong từ điển.**

vector<string> getDistinctProducts(vector<string> &products) {

set<string> s(products.begin(), products.end());

vector <string> v(s.begin(), s.end());

return v;

}

main() {

vector <string> products { "watermelon", "grapes", "grapes", "apple", "grapes" };

products = getDistinctProducts(products);

for (auto x : products) cout << x << endl;

}

**Bài 5. Cho một số n, hãy đưa số dãy số siêu nguyên tố nhỏ hơn hoặc bằng n đã được sắp xếp tăng dần.**

vector <int> basePrimeNumbers { 2, 3, 5, 7 };

vector <int> nextPrimeDigit { 1, 3, 7, 9 };

bool isPrime(int n) {

if (n <= 3) return n > 1;

if (n % 2 == 0 || n % 3 == 0) return 0;

for (int i = 5; i \* i <= n; ++i) {

if (n % i == 0 || n % (i + 2) == 0) return 0;

}

return 1;

}

vector <int> superPrimeNumber(int n) {

queue <int> q;

vector <int> v;

for (auto x : basePrimeNumbers) q.push(x);

while (q.size()) {

for (auto x : nextPrimeDigit) {

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

if (k > n) break;

if (isPrime(k)) q.push(k);

}

v.push\_back(q.front());

q.pop();

}

return v;

}

main() {

int n = 30;

vector <int> v = superPrimeNumber(n);

for (auto x : v) cout << x << " ";

}

**Bài 6. Cho một mảng các số nguyên arr và một số nguyên dương k. Hãy viết hàm trả về các phần tử âm đầu tiên trong cửa số k. Nếu không tồn tại số âm nào output 0.**

vector <int> firstNegative(vector <int> arr, int k) {

queue <int> q;

vector <int> ans;

for(int i = 0; i < k – 1; ++i) if (arr[i] < 0) q.push(i);

for(int i = k - 1; i < arr.size(); ++i) {

if (arr[i] < 0) q.push(i);

while (q.size() && q.front() < i - k + 1) q.pop();

ans.push\_back(q.empty() ? 0 : arr[q.front()]);

}

return ans;

}

main() {

vector <int> arr { -8, 2, 3, -6, 10 };

int k = 2;

vector <int> ans = firstNegative(arr, k);

for (auto x : ans) cout << x << " ";

}

**Bài 7. Cho một ma trận trong đó mỗi ô trong ma trận có thể có các giá trị 0, 1 hoặc 2 có nghĩa sau:**

**• 0: Là một ô trống.**

**• 1: Là một quả cam tươi.**

**• 2: Là một quả cam bị hỏng.**

**Chúng ta phải xác định thời gian tối thiểu mà tất cả các quả cam bị hỏng. Một quả cam hỏng ở chỉ số [i, j] có thể làm hỏng quả cam tươi khác ở các chỉ số [i-1, j], [i + 1, j], [i, j-1], [i, j + 1] (lên , xuống, trái và phải). Nếu không thể bị hỏng tất cả quả cam thì trả về -1.**

vector <int> dx { 1, -1, 0, 0 };

vector <int> dy { 0, 0, -1, 1 };

bool isInRange(int i, int j, int n, int m) {

return i >= 0 && j >= 0 && i < n && j < m;

}

int rotOrange(vector<vector<int>> &input) {

queue <pair<int, int>> q;

int n = input.size();

int m = input[0].size();

int freshOrangeCounts = 0;

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

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

if (input[i][j] == 2) q.push({i, j});

if (input[i][j] == 1) ++freshOrangeCounts;

}

}

if (!freshOrangeCounts) return 0;

int ans = -1;

while (q.size()) {

int count = q.size();

++ans;

while (count--) {

pair<int, int> p = q.front(); q.pop();

for (int i = 0; i < 4; ++i) {

int X = p.first + dx[i];

int Y = p.second + dy[i];

if (isInRange(X, Y, n, m) && input[X][Y] == 1) {

--freshOrangeCounts;

input[X][Y] = 2;

q.push({X, Y});

}

}

}

}

if (freshOrangeCounts) return -1;

return ans;

}

main() {

vector <vector<int>> matrix { { 2, 1, 0, 2, 1 },

{ 1, 0, 1, 2, 1 },

{ 1, 0, 0, 2, 1 } };

vector <vector<int>> matrix2 { { 2, 1, 0, 2, 1 },

{ 0, 0, 1, 2, 1 },

{ 1, 0, 0, 2, 1 } };

vector <vector<int>> matrix3 { { 0, 0, 0 },

{ 0, 0, 0 },

{ 0, 0, 0 } };

cout << rotOrange(matrix3);

}

**Bài 8. Do những ngày hè quá nóng bức và nhàm chán nên Tide đã nghĩ ra một trò chơi khá thú vị với queue. Ban đầu trong queue có 5 số 1, 2, 3, 4, 5 với mỗi lượt chơi Tide sẽ xóa phần tử ở đầu queue và cho 2 phần tử đó xuống cuối của queue và cứ tiếp tục cho đến khi Tide cảm thấy mệt và không chơi được nữa.**

int solveNonQueue(int n) {

int x = 1, val = n;

while(val - (x \* 5) > 0) {

val -= x \* 5;

x \*= 2;

}

return (val + x - 1) / x;

}

int solveWithQueue(int n) {

queue <pair<int, int>> q;

for (auto x : { 1, 2, 3, 4, 5 }) q.push({x, 1});

while(n > q.front().second) {

n -= q.front().second;

q.push({q.front().first, q.front().second \* 2});

q.pop();

}

return q.front().first;

}

main() {

int n = 15999998;

cout << solveNonQueue(n) << endl;

cout << solveWithQueue(n);

}

**Bài 9. Giả sử có một đường tròn. Có n trạm xăng trên vòng tròn đó. Bạn được cho hai tập dữ liệu.**

**1. Lượng xăng mà mọi trạm xăng có.**

**2. Khoảng cách từ trạm xăng đó đến trạm xăng kế tiếp.**

**Tính điểm đầu tiên mà từ đó một xe tải có thể kết thúc hành trình trên vòng tròn (Xe tải sẽ dừng lại tại mỗi trạm xăng và nó có dung tích vô hạn). Độ phức tạp thời gian kỳ vọng là O(n). Giả sử với một lít xăng, xe tải có thể đi được một đơn vị quãng đường.**

int getStart(vector<pair<int, int>> input) {

int start = 0, fuel = 0, lost = 0;

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

fuel += input[i].first - input[i].second;

if (fuel < 0) {

start = i + 1;

lost += fuel;

fuel = 0;

}

}

return fuel + lost >= 0 ? start : -1;

}

main() {

vector<pair<int, int>> input { { 4, 6 }, { 6, 5 }, { 7, 3 }, { 4, 5 } };

cout << getStart(input);

}

**Bài 10. Một hiệu trưởng muốn đi thăm các ký túc xá (có tất cả n ký túc). Ông ta rất bận nên chỉ thăm được k (k ≤ n) nơi gần nhất. Các ký túc xá được tọa lạc trên một mặt phẳng hai chiều. Hãy giúp ông ấy, biết rằng bạn có tọa độ của n ký túc và vị trí hiện tại của ông ấy có tọa độ là O(0,0).**

class Compare {

public:

bool operator() (const pair<int, int> &a, const pair<int, int> &b) {

return a.first \* a.first + a.second \* a.second > b.first \* b.first + b.second \* b.second;

}

};

vector<pair<int, int>> closetPoints(vector<pair<int, int>> input, int k) {

priority\_queue <pair<int, int>, vector<pair<int, int>>, Compare> pq;

vector<pair<int, int>> ans;

for (auto x : input) pq.push(x);

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

ans.push\_back(pq.top());

pq.pop();

}

return ans;

}

main() {

vector<pair<int, int>> input = { { 1, 0 }, { 2, 1 }, { 3, 6 }, { -5, 2 }, { 1, -4 } };

int k = 3;

vector<pair<int, int>> ans = closetPoints(input, k);

for (auto x : ans) cout << "(" << x.first << ", " << x.second << ") ";

}

**Bài 11. Monk hiện có một mảng các số nguyên A. Với mỗi chỉ số i, anh ta muốn tìm tích của các số lớn nhất, số lớn thứ hai và số lớn thứ ba trong khoảng [1, i]. Chú ý: các số có thể giống nhau nhưng chúng phải khác nhau về chỉ số.**

vector<int> biggestProducts(vector<int> input) {

priority\_queue<int> pq;

vector<int> bg(3);

vector<int> ans { input[0], input[0] \* input[1] };

for(int i = 0; i < 2; ++i) pq.push(input[i]);

for(int i = 2; i < input.size(); ++i) {

pq.push(input[i]);

for(int j = 0 ; j < 3; ++j) {

bg[j] = pq.top();

pq.pop();

}

ans.push\_back(bg[0] \* bg[1] \* bg[2]);

for(int j = 0; j < 3; ++j) pq.push(bg[j]);

}

return ans;

}

main() {

vector<int> input { 0, 1, 3, 5, 6, 2, 7, 2, 3, 8 };

for (auto x : biggestProducts(input)) cout << x << " ";

}

**Bài 12. Cho một mảng số nguyên và một số nguyên k. Tìm số lớn nhất trong từng mảng con liên tiếp kích thước k.**

vector <int> biggestVal(vector<int> input, int k) {

deque<int> dq;

vector<int> ans;

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

while(dq.size() && input[i] > input[dq.back()]) dq.pop\_back();

dq.push\_back(i);

}

ans.push\_back(input[dq.front()]);

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

if (dq.front() <= i - k) dq.pop\_front();

while(dq.size() && input[i] > input[dq.back()]) dq.pop\_back();

dq.push\_back(i);

ans.push\_back(input[dq.front()]);

}

return ans;

}

main() {

vector<int> arr { 1, 2, 3, 1, 4, 5, 2, 3, 6 };

int k = 3;

vector<int> ans = biggestVal(arr, k);

for (auto i : ans) cout << i << " ";

}

**Bài 13. Cho một mảng số nguyên và một số nguyên k. Tính tổng các số nhỏ nhất và lớn nhất trong từng mảng con liên tiếp kích thước k.**

int biggestSum(vector<int> input, int k) {

deque<int> dq1, dq2;

int ans = 0;

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

while(dq1.size() && input[i] > input[dq1.back()]) dq1.pop\_back();

while(dq2.size() && input[i] < input[dq2.back()]) dq2.pop\_back();

dq1.push\_back(i);

dq2.push\_back(i);

}

ans += input[dq1.front()] + input[dq2.front()];

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

if (dq1.front() <= i - k) dq1.pop\_front();

if (dq2.front() <= i - k) dq2.pop\_front();

while(dq1.size() && input[i] > input[dq1.back()]) dq1.pop\_back();

while(dq2.size() && input[i] < input[dq2.back()]) dq2.pop\_back();

dq1.push\_back(i);

dq2.push\_back(i);

ans += input[dq1.front()] + input[dq2.front()];

}

return ans;

}

main() {

vector<int> arr { 2, 5, -1, 7, -3, -1, -2 };

int k = 4;

cout << biggestSum(arr, k);

}

**Bài 14. Cho dãy số nguyên n phần tử a1, a2, ..., an, hãy thống kê tần suất xuất hiện của từng phần tử có trong dãy theo thứ tự từ bé đến lớn.**

void solve(vector<int> input) {

map <int, int> m;

for (auto x : input) ++m[x];

for (auto i : m) cout << i.first << ": " << i.second << endl;

}

main() {

vector <int> input { 5, 6, 5, 1, 2, 3, 3, 2, 1, 5, 7, 7, 7, 1, 5 };

solve(input);

}

**Bài 15. Cho dãy số nguyên n phần tử a1, a2, ..., an, hãy tìm những phần tử có tần suất xuất hiện nhiều nhất trong dãy. Ví dụ: a = {4 7 2 8 4 8 3 2}. Kết quả là các số {2 4 8} có tần suất lớn nhất là 2.**

void solve(vector<int> input) {

int maxFrequency = 0;

map<int, int> m;

for (auto x : input) maxFrequency = max(maxFrequency, ++m[x]);

for (auto x : m) if (x.second == maxFrequency) cout << x.first << " ";

cout << endl << "FREQUENCY: " << maxFrequency;

}

main() {

vector <int> a { 4, 7, 2, 8, 4, 8, 3, 2 };

solve(a);

}

**Bài 16. Cho một chuỗi s, hãy đưa ra một dãy lần lượt là các ký tự và số lần xuất hiện của nó, các ký tự sắp xếp theo thự tự từ điển.**

vector <string> countChar(std::string str) {

map<char, int> m;

vector<string> v;

for (auto x : str) ++m[x];

for (auto x : m) {

string temp = "";

temp += x.first;

temp += " " + to\_string(x.second);

cout << temp << endl;

v.push\_back(temp);

}

return v;

}

main() {

string s1 = "aacccd";

string s2 = "aabbbca";

vector <string> v1 = countChar(s1), v2 = countChar(s2);

for (auto x : v1) cout << x << " ";

cout << endl;

for (auto x : v2) cout << x << " ";

cout << endl;

}

**Bài 17. Bạn được cung cấp một chuỗi s làm đầu vào và bạn phải in biểu mẫu đã sửa đổi.**

**Chuỗi được sửa đổi theo cách sau:**

**• Những ký tự giống nhau chỉ lấy duy nhất một ký tự.**

**• Ký tự nào có tần số xuất hiện trong chuỗi s nhiều hơn được sắp xếp trước.**

**• Những ký tự có cùng tần số xuất hiện thì sắp xếp theo thứ từ từ điển từ nhỏ đến lớn.**

bool customCompare(pair<char, int> &a, pair<char, int> &b) {

if (a.second == b.second) return a.first < b.first;

return a.second > b.second;

}

string modifyString(string s) {

unordered\_map<char, int> m;

for (auto x : s) ++m[x];

string ans = "";

vector <pair<char, int>> v(m.begin(), m.end());

sort(v.begin(), v.end(), customCompare);

for (auto x : v) ans += x.first;

return ans;

}

main() {

string s1 = "codelearn"; cout << modifyString(s1) << endl;

string s2 = "helloworld"; cout << modifyString(s2);

}

**Bài 18. Cho hai dãy số nguyên arr1 và arr2, hãy tính tổng những số xuất hiện trong cả hai dãy, lưu ý là mỗi số chỉ được tính một lần.**

int sumOfCommon(vector<int> arr1, vector<int> arr2) {

unordered\_map<int, bool> m;

int sum = 0;

for (auto x : arr1) m[x] = 1;

for (auto x : arr2) {

if (m[x] == 1) {

sum += x;

m[x] = 0;

}

}

return sum;

}

main() {

vector <int> arr1 { 6, 7, 5, 4, 6, 8 };

vector <int> arr2 { 2, 5, 7, 5, 3 };

cout << sumOfCommon(arr1, arr2) << endl;

}

**Bài 19. Cho một dãy số nguyên arr và một số nguyên dương sum. Hãy kiểm tra xem dãy số có tồn tại hai số có tổng bằng sum hay không.**

bool checkSum(vector<int> arr, int sum) {

unordered\_map <int, bool> m;

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

if (m.find(sum - arr[i]) != m.end()) return 1;

m[arr[i]] = 1;

}

return 0;

}

main() {

vector<int> arr1 { 2, 4, -1, 9, 8 }, arr2 { 2, 5, 3, 8, 9}, arr3 { 4, 7, 3, 5 };

int sum1 = 6, sum2 = 3, sum3 = 6;

cout << checkSum(arr1, sum1) << endl;

cout << checkSum(arr2, sum2) << endl;

cout << checkSum(arr3, sum3) << endl;

}

**Bài 20. Cho một dãy gồm các số nguyên, hãy đưa ra số thỏa mãn là số lớn nhất trong những số có tần số bé nhất.**

int largestElement(vector<int> v) {

map<int, int> m;

for (auto x : v) ++m[x];

int frequency = LLONG\_MAX, res;

for (auto x : m) {

if (x.second <= frequency) {

frequency = x.second;

res = x.first;

}

}

return res;

}

main() {

vector <int> arr1 { 2, 2, 4, 4, 7, 7, 7 }, arr2 { 1, 3, 4, 5, 5 };

cout << largestElement(arr1) << endl;

cout << largestElement(arr2);

}

**Bài 21. Cho một danh bạ điện thoại và danh sách các tên. Ứng với mỗi tên trong danh sách hãy đếm xem tên này là bắt đầu của bao nhiêu tên trong danh bạ điện thoại, hay nói cách khác hãy đếm xem tên này là prefix của bao nhiêu tên trong danh bạ.**

vector<int> countPrefix(vector<string> contacts, vector<string> names) {

unordered\_map <string, int> m;

vector <int> ans;

for (auto contact : contacts) {

string s = "";

for (auto i : contact) {

s += i;

++m[s];

}

}

for (auto name : names) ans.push\_back(m[name]);

return ans;

}

main() {

vector <string> contacts { "Codelearn", "Codewar" }, names { "Code", "Codel", "io" };

vector <int> v = countPrefix(contacts, names);

for (auto i : v) cout << i << " ";

}

**Bài 22. Cho danh sách các sản phẩm của 2 kho hàng A và B. Do chiến lược kinh doanh bạn được giao nhiệm nhập các sản phẩm từ kho B vào kho A sao cho những sản phẩm nào đã có trong kho A thì không nhập.**

vector<bool> mergeProducts(vector<string> A, vector<string> B) {

unordered\_map <string, int> m;

vector <bool> ans;

for (auto x : A) ++m[x];

for (auto x : B) ans.push\_back(m.find(x) == m.end());

return ans;

}

main() {

vector <string> A { "Banana", "Banana", "Apple" }, B { "Orange", "Apple", "Banana", "Watermelon" };

vector <bool> v = mergeProducts(A, B);

for (auto i : v) cout << i << " ";

}

**Bài tập chương 3**

**Bài 1. Cho mảng số nguyên. Hãy sắp xếp chúng theo nguyên tắc: a đứng trước b nếu tổng các chữ số của a nhỏ hơn b. Nếu hai số có tổng các chữ số bằng nhau, số nào nhỏ hơn sẽ đứng trước.**

**Bài 2. 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}.**

#include "iostream"

#include "vector"

using namespace std;

const int RANGE = 10;

void countingSort(vector<int> v) {

vector<int> count = vector<int>(10, 0);

vector<int> res = vector<int>(v.size());

int mot = 0; // má»‘t

int maxCount = 0;

for (int i: v) {

count[i]++;

}

for (int i: v) {

if (count[i] > maxCount) {

mot = i;

maxCount = count[i];

}

}

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

count[i] += count[i - 1];

}

for (int i: v) {

res[count[i] - 1] = i;

count[i]--;

}

for (int i: res) { cout << i << ' '; }

cout << "\nmot: " << mot << "\n";

if (res.size() % 2 == 0) {

cout << "trung vi: " << (res[res.size() / 2] \* 1.0 + res[res.size() / 2 - 1] \* 1.0) / 2;

}

else {

cout << "trung vi: " << res[res.size() / 2];

}

}

int main() {

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

countingSort(v);

}

**Bài 3. Bạn được cho một mảng A kích thước N và một số nguyên K. Mảng này gồm  
N số nguyên có giá trị từ 1 đến 107. Mỗi phần tử a[i] của mảng này có** trọng số đặc  
biệt **được tính: a[i] % K. Bạn cần sắp xếp mảng này theo thứ tự giảm dần của các** trọng  
số đặc biệt**, tức là phần tử có trọng số đặc biệt lớn nhất thì đứng ở đầu dãy, tiếp đến là  
phần tử có trọng số đặc biệt lớn thứ hai, … Trong trường hợp hai phần tử có cùng trọng  
số đặc biệt thì phần tử có giá trị nhỏ hơn sẽ đứng trước.**

#include "iostream"

#include "vector"

using namespace std;

class Number {

public:

int n;

int trongso;

Number(int n, int trongso) : n(n), trongso(trongso) {}

Number() {}

};

bool cmp(Number a, Number b) {

if (a.trongso == b.trongso) { return a.n < b.n; }

else { return a.trongso > b.trongso; }

}

int main() {

vector<int> v = {13, 120, 7, 4};

int k = 3;

vector<Number> num = vector<Number>(v.size());

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

num[i] = Number(v[i], v[i] % k);

}

sort(num.begin(), num.end(), cmp);

for (Number i: num) {

cout << i.n << ' ' << i.trongso << '\n';

}

}

**Bài 5. Chèn dãy giá trị {4371, 1323, 6173, 4199, 4344, 9679, 1989} vào bảng băm với hàm băm hash(x) = x % 10:**

1. **Bảng băm dây chuyền**

int hash(int x){

int index;index = x % 10;

return index;}

void insert(int x, vector<int> a[]){int index = hash(x); a[index].push\_back(x);}

int main(){

vector<int> a[10];insert(4371,a);

insert(1323,a);insert(6173,a);

insert(4199,a);insert(4344,a);

insert(9679,a);insert(1989,a);}

1. **Thăm dò bậc hai**

int hashTableSize = 10;

int size = 0;

class Node{

public:

int key;

bool ischeck = true;};

int hash(int x){

int index;index = x % hashTableSize;

return index;}

void insert(int x, Node a[]){

int index = hash(x);

if(size == hashTableSize){ cout << "Bang bam da day khong the them phan tu " <<x <<" vao bang bam" << endl;}

else{ int i = 1;

while(a[index].ischeck == false){index = (index + i\*i) % hashTableSize;i++;cout <<"a ";}

a[index].key = x;a[index].ischeck = false;

size++;}}

int main(){

Node a[hashTableSize];

insert(4371,a);insert(1323,a);

insert(6173,a);insert(4199,a);

insert(4344,a);insert(9679,a);

insert(1989,a);

for(int i = 0 ; i < 10 ; i++)

cout<<a[i].key<<"\t";}

**Bài 6. Cài đặt từ điển dựa trên bảng băm:**

**- Dây chuyền.**

#include<bits/stdc++.h>

using std::vector;

using std::list;

using std::string;

using std::cout;

using std::endl;

struct dict{

string key,value;

dict():key(""),value(""){};

dict(string Key, string Value):key(Key),value(Value){};

};

class HashChain\_std{

private:

int size,count;

vector<list<dict> > table;

int PreHashing(string key\_str);

int HashFunction(string key\_str);

public:

HashChain\_std(){};

HashChain\_std(int m):size(m),count(0){

table.resize(size); }

void Insert(dict data);

string Search(string key);

};

string HashChain\_std::Search(string key\_str){

int index = HashFunction(key\_str);

for (list<dict>::iterator itr = table[index].begin(); itr != table[index].end(); itr++){

if ((\*itr).key == key\_str) {

return (\*itr).value;}

}

return "Tu can kiem khong co trong tu dien bang bam";

}

void HashChain\_std::Insert(dict data){

int index = HashFunction(data.key);

table[index].push\_front(data);

}

int HashChain\_std::PreHashing(string key\_str){

int exp = 9,key\_int = 0,p = 1;

for (int i = (int)key\_str.size()-1; i >= 0; i--) {

key\_int += key\_str[i]\*p;

p \*= exp;}

return key\_int;

}

int HashChain\_std::HashFunction(string key\_str){

return (PreHashing(key\_str) % this->size);

}

int main() {

HashChain\_std hash(5);

hash.Insert(dict("dog","cho"));

hash.Insert(dict("hello","xin chao"));

cout <<hash.Search("hi") << ". " << endl;

}

* **Thăm dò tuyến tính**

#include<bits/stdc++.h>

#include<hashtable.h>

using namespace std;

int hashTableSize = 23;

typedef struct{

string tienganh; string tiengviet;

}tudien;

int hash(string s){

int index,sum = 0;

for(int i = 0 ; i < s.size() ; i++) {sum+=(int)s[i];}

index = sum % hashTableSize;

return index;

}

void insert(string s, string s1, tudien a[]){

int index = hash(s);

while(a[index].tienganh != ""){

index = (index + 1) % hashTableSize ;}

a[index].tienganh = s; a[index].tiengviet = s1;

}

string search(string s, tudien a[]){

int index = hash(s) , i = 0;

while(a[index].tienganh != s && a[index].tienganh != ""){

index = (index + 1) % hashTableSize ;i++;}

if(a[index].tienganh == ""){return "Tu can kiem khong co trong bang bam";}

return a[index].tiengviet;

}

int main(){

tudien a[23];

for(int i = 0 ; i < 23 ; i++) {a[i].tienganh = "";}

insert("hello","xin chao",a);

insert("good","tot",a);

insert("sad","buon",a);

cout<<search("hi",a);

}

* **Thăm dò bậc hai**

#include<bits/stdc++.h>

#include<hashtable.h>

using namespace std;

int hashTableSize = 23;

typedef struct{

string tienganh; string tiengviet;

}tudien;

int hash(string s){

int index,sum = 0;

for(int i = 0 ; i < s.size() ; i++) {sum+=(int)s[i];}

index = sum % hashTableSize;

return index;}

void insert(string s, string s1, tudien a[]){

int index = hash(s),i=1;

while(a[index].tienganh != ""){

index = (index + i\*i) % hashTableSize ;}

a[index].tienganh = s; a[index].tiengviet = s1;

}

string search(string s, tudien a[]){

int index = hash(s) , i = 0;

while(a[index].tienganh != s && a[index].tienganh != ""){

index = (index + 1) % hashTableSize ;i++;}

if(a[index].tienganh == ""){return "Tu can kiem khong co trong bang bam";}

return a[index].tiengviet;

}

int main(){

tudien a[23];

for(int i = 0 ; i < 23 ; i++) {a[i].tienganh = "";}

insert("hello","xin chao",a);

insert("good","tot",a);

insert("sad","buon",a);

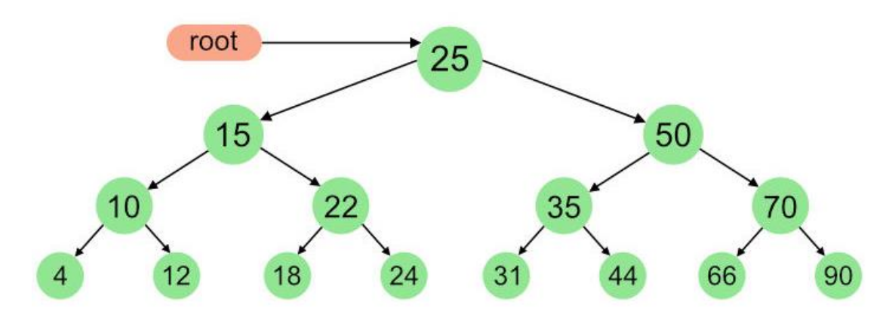
cout<<search("hi",a);}

**Bài tập chương 5**

**Bài 3. Cho một mảng các xâu ký tự. Ví dụ: s[] = {"flower", "flow", "flight"}.**

**a. Vẽ hình minh họa quá trình chèn các ký tự của các xâu trong mảng s[] vào cây**

**tiền tố (trie).**

****

**b. Viết chương trình chèn các xâu ký tự vào cây tiền tố và tìm tiền tố chung dài**

**nhất của các xâu ký tự đó. Ví dụ: tiền tố chung dài nhất của mảng s[] là “fl”.**

struct trie {

vector<trie \*> child;

bool hasWord;

trie() {

child.resize(26, 0);

hasWord = 0;

}

};

void add(trie \*t, string s) {

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

if (!t->child[s[i] - 'a']) t->child[s[i] - 'a'] = new trie();

t = t->child[s[i] - 'a'];

}

t->hasWord = 1;

}

bool haveOnlyChild(trie \* root) {

int cnt = 0;

for (int i = 0; i < 26; ++i) if (root -> child[i]) ++cnt;

return cnt == 1;

}

string findLCP(trie \* root) {

string LCP = "";

while (root && haveOnlyChild(root)) {

for (int i = 0; i < 26; ++i) {

if (root -> child[i]) {

root = root -> child[i];

LCP += (char) (i + 'a');

break;

}

}

}

return LCP;

}

int main() {

trie \*t = new trie();

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

for (string temp : words) add(t, temp);

cout << findLCP(t);

return 0;

}

**Bài 4. Cho một mảng các ký tự ch[] và tần suất của mỗi ký tự là freq[] như sau:**

**Đầu vào: ch[] = {‘a’, ‘b’, ‘c’, ‘d’, ‘e’, ‘f’}, freq[] = {5, 9, 12, 13, 16, 45}.**

**Đầu ra:**

**f 0**

**c 100**

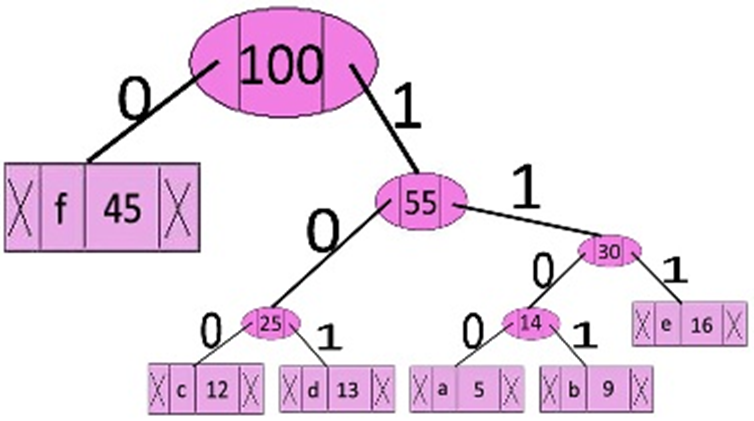
**d 101**

**a 1100**

**b 1101**

**e 111**

1. **Vẽ cây Huffman với dữ liệu đầu ra như trên.**



**b. Viết chương trình xây dựng cây Huffman và gán từ mã (code word) cho mọi ký**

**tự trong ch[] bằng cách sử dụng Hàng đợi ưu tiên (có thể khai thác thư viện của**

**ngôn ngữ lập trình).**

class HuffmanTreeNode {

public:

char data;

int freq;

HuffmanTreeNode \*left, \*right;

HuffmanTreeNode(char data, int freq, HuffmanTreeNode \*left = nullptr, HuffmanTreeNode \*right = nullptr) {

this -> data = data;

this -> left = left;

this -> right = right;

this -> freq = freq;

}

};

class Compare {

public:

bool operator()(HuffmanTreeNode \*a, HuffmanTreeNode \*b) {

return a -> freq > b -> freq;

}

};

HuffmanTreeNode \*makeTree(priority\_queue<HuffmanTreeNode \*, vector<HuffmanTreeNode \*>, Compare> pq) {

while (pq.size() != 1) {

HuffmanTreeNode \*left = pq.top(); pq.pop();

HuffmanTreeNode \*right = pq.top(); pq.pop();

pq.push(new HuffmanTreeNode('$', left -> freq + right -> freq, left, right));

}

return pq.top();

}

void printTree(HuffmanTreeNode \*root, string s) {

if (root -> left) printTree(root -> left, s + "0");

if (root -> right) printTree(root -> right, s + "1");

if (root -> left || root -> right) return;

cout << root -> data << " " << s << endl;

}

void HuffmanCodes(vector <char> ch, vector <int> freq) {

priority\_queue<HuffmanTreeNode \*, vector<HuffmanTreeNode \*>, Compare> pq;

for (int i = 0; i < freq.size(); ++i) pq.push(new HuffmanTreeNode(ch[i], freq[i]));

HuffmanTreeNode \*root = makeTree(pq);

vector<int> v(freq.size());

printTree(root, "");

}

int main() {

vector<char> ch = vector<char> {'a', 'b', 'c', 'd', 'e', 'f'};

vector<int> freq = vector<int> {5, 9, 12, 13, 16, 45};

HuffmanCodes(ch, freq);

return 0;

}