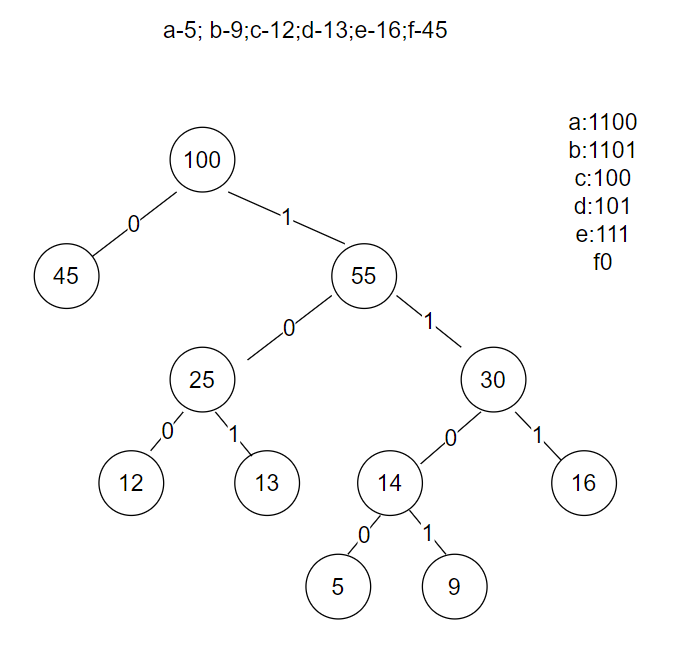
Câu 1. Cho một mảng các ký tự ch[] và tần suất của mỗi ký tự là freq[] như sau:  
Đầu vào: ch[] = {‘a’, ‘b’, ‘c’, ‘d’, ‘e’, ‘f’}, freq[] = {5, 9, 12, 13, 16, 45}.  
Đầu ra:  
f 0  
c 100  
d 101  
a 1100  
b 1101  
e 111  
a. Vẽ cây Huffman với dữ liệu đầu ra như trên.  
b. Viết chương trình xây dựng 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).  
c. Vẽ minh họa các bước xây dựng cây Huffman với dữ liệu đầu vào như trên



#include <bits/stdc++.h>

using namespace std;

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;

}

Câu 2. Viết chương trình chèn dãy giá trị {4371, 1323, 6173, 4199, 4344, 9679, 1989}  
vào bảng băm thăm dò tuyến tính với hàm băm hash(x) = x % 10 và vẽ hình minh họa  
quá trình chèn từng giá trị trên vào bảng băm thăm dò tuyến tính.(python)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 4371 | 1323 | 6173 | 4199 | 4344 | 9679 | 1989 |
|  |  |  |  |  |  |  |  |  |
| 0 |  |  |  |  |  |  | 9679 | 9679 |
| 1 |  | 4371 | 4371 | 4371 | 4371 | 4371 | 4371 | 4371 |
| 2 |  |  |  |  |  |  |  | 1989 |
| 3 |  |  | 1323 | 1323 | 1323 | 1323 | 1323 | 1323 |
| 4 |  |  |  | 6173 | 6173 | 6173 | 6173 | 6173 |
| 5 |  |  |  |  |  | 4344 | 4344 | 4344 |
| 6 |  |  |  |  |  |  |  |  |
| 7 |  |  |  |  |  |  |  |  |
| 8 |  |  |  |  |  |  |  |  |
| 9 |  |  |  |  | 4199 | 4199 | 4199 | 4199 |

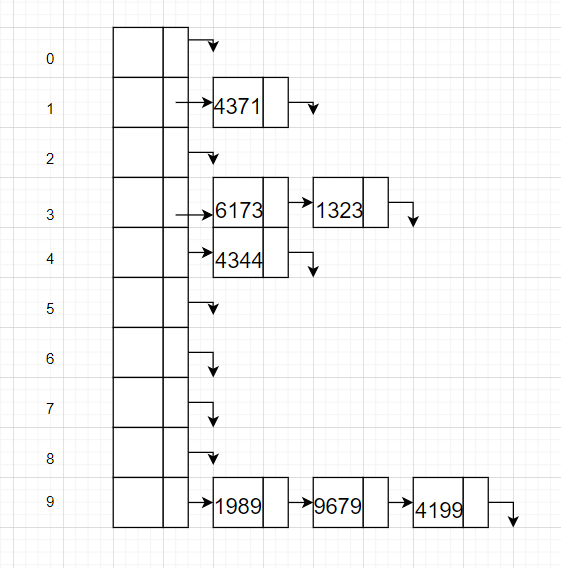
hash\_table\_size = 10 # kích cỡ bảng băm  
hash\_table = [-1] \* hash\_table\_size # bảng băm  
  
  
# hàm băm  
def hash\_function(x):  
 return x % 10  
  
  
# thêm số mới  
def insert(x):  
 hash\_value = hash\_function(x)  
 # thêm vào bảng băm bằng phương pháp thăm dò tuyến tính  
 for i in range(len(hash\_table)):  
 if hash\_table[(hash\_value + i) % len(hash\_table)] == -1:  
 hash\_table[(hash\_value + i) % len(hash\_table)] = x  
 break  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 lst = [4371, 1323, 6173, 4199, 4344, 9679, 1989]  
 for i in lst:  
 insert(i)  
  
 # in ra toàn bộ số  
 for i in range(0, len(hash\_table)):  
 if hash\_table[i] == -1:  
 print(str(i) + ". ")  
 else:  
 print(str(i) + ". " + str(hash\_table[i]))

Câu 3. Viết chương trình chèn dãy giá trị {4371, 1323, 6173, 4199, 4344, 9679, 1989}  
vào bảng băm thăm dò bậc hai với hàm băm hash(x) = x % 10 và vẽ hình minh họa quá  
trình chèn từng giá trị trên vào bảng băm thăm dò bậc hai.(python)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 4371 | 1323 | 6173 | 4199 | 4344 | 9679 | 1989 |
|  |  |  |  |  |  |  |  |  |
| 0 |  |  |  |  |  |  | 9679 | 9679 |
| 1 |  | 4371 | 4371 | 4371 | 4371 | 4371 | 4371 | 4371 |
| 2 |  |  |  |  |  |  |  |  |
| 3 |  |  | 1323 | 1323 | 1323 | 1323 | 1323 | 1323 |
| 4 |  |  |  | 6173 | 6173 | 6173 | 6173 | 6173 |
| 5 |  |  |  |  |  | 4344 | 4344 | 4344 |
| 6 |  |  |  |  |  |  |  |  |
| 7 |  |  |  |  |  |  |  |  |
| 8 |  |  |  |  |  |  |  | 1989 |
| 9 |  |  |  |  | 4199 | 4199 | 4199 | 4199 |

hash\_table\_size = 10 # kích cỡ bảng băm  
hash\_table = [-1] \* hash\_table\_size # bảng băm  
  
  
# hàm băm  
def hash\_function(x):  
 return x % 10  
  
  
# thêm số mới  
def insert(x):  
 hash\_value = hash\_function(x)  
 # thêm vào bảng băm bằng phương pháp thăm dò bậc 2  
 for i in range(len(hash\_table)):  
 if hash\_table[(hash\_value + i \* i) % len(hash\_table)] == -1:  
 hash\_table[(hash\_value + i \* i) % len(hash\_table)] = x  
 break  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 lst = [4371, 1323, 6173, 4199, 4344, 9679, 1989]  
 for i in lst:  
 insert(i)  
  
 # in ra toàn bộ số  
 for i in range(0, len(hash\_table)):  
 if hash\_table[i] == -1:  
 print(str(i) + ". ")  
 else:  
 print(str(i) + ". " + str(hash\_table[i]))

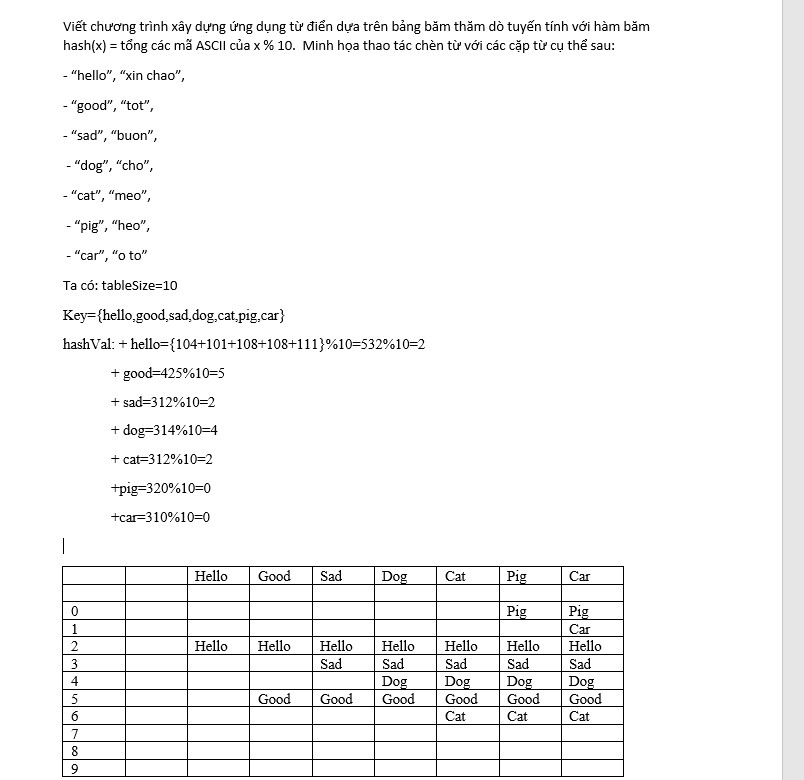
Câu 4. Viết chương trình chèn dãy giá trị {4371, 1323, 6173, 4199, 4344, 9679, 1989}  
vào bảng băm dây truyền với hàm băm hash(x) = x % 10 và vẽ hình minh họa quá trình  
chèn từng giá trị trên vào bảng băm dây truyền.(python)



hash\_table\_size = 10 # kích cỡ bảng băm  
hash\_table = [] # bảng băm  
  
for i in range(hash\_table\_size):  
 hash\_table.append([])  
  
  
# hàm băm  
def hash\_function(x):  
 return x % 10  
  
  
# thêm số mới  
def insert(x):  
 hash\_value = hash\_function(x)  
 # thêm vào bảng băm bằng phương pháp dây chuyền  
 hash\_table[hash\_value].append(x)  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 lst = [4371, 1323, 6173, 4199, 4344, 9679, 1989]  
 for i in lst:  
 insert(i)  
  
 # in ra toàn bộ số  
 for i in range(0, len(hash\_table)):  
 print(str(i) + ". ", end="")  
  
 for j in hash\_table[i]:  
 print(str(j) + " ", end="")  
 print()

**Câu 5**. Viết chương trình xây dựng ứng dụng từ điển dựa trên bảng băm thăm dò tuyến  
tính với hàm băm hash(*x*) = tổng các mã ASCII của *x* % 10.(python)  
a. Từ điển bao gồm các chức năng: chèn từ vào từ điển và lấy nghĩa tiếng Việt với  
đầu vào là một từ tiếng Anh.  
b. Minh họa thao tác chèn từ với các cặp từ cụ thể sau:  
- “hello”, “xin chao”,  
- “good”, “tot”,  
- “sad”, “buon”,  
- “dog”, “cho”,  
- “cat”, “meo”,  
- “pig”, “heo”,  
- “car”, “o to”.  
Biết rằng mã ASCII của các ký tự từ ‘a’ đến ‘z’ được cho ở bảng dưới đây:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Ký tự | a | b | c | d | e | f | g | h | i | j | k | l | m |
| Mã ASCII | 97 | 98 | 99 | 100 | 101 | 102 | 103 | 104 | 105 | 106 | 107 | 108 | 109 |
| Ký tự | n | o | p | q | r | s | t | u | v | w | x | y | z |
| Mã ASCII | 110 | 111 | 112 | 113 | 114 | 115 | 116 | 117 | 118 | 119 | 120 | 121 | 122 |



hash\_table\_size = 10 # kích cỡ bảng băm  
en\_hash\_table = [""] \* hash\_table\_size # bảng băm nghĩa tiếng anh  
vi\_hash\_table = [""] \* hash\_table\_size # bảng băm nghĩa tiếng việt  
  
  
# hàm băm  
def hash\_function(s):  
 res = 0  
 for i in s:  
 res += ord(i)  
 return res % 10  
  
  
# thêm từ mới  
def insert(en, vi):  
 hash\_value = hash\_function(en)  
 # thêm vào bảng băm bằng phương pháp thăm dò tuyến tính  
 for i in range(len(en\_hash\_table)):  
 if en\_hash\_table[(hash\_value + i) % len(en\_hash\_table)] == "":  
 en\_hash\_table[(hash\_value + i) % len(en\_hash\_table)] = en  
 vi\_hash\_table[(hash\_value + i) % len(vi\_hash\_table)] = vi  
 break  
  
  
# lấy từ đã có  
def get(en):  
 hash\_value = hash\_function(en)  
 if en\_hash\_table[hash\_value % len(en\_hash\_table)] != "":  
 for i in range(len(en\_hash\_table)):  
 if en\_hash\_table[(hash\_value + i) % len(en\_hash\_table)] == en:  
 return vi\_hash\_table[(hash\_value + i) % len(vi\_hash\_table)]  
 # không tìm thấy từ  
 return "word not found!!!"  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 insert("hello", "xin chao")  
 insert("good", "tot")  
 insert("sad", "buon")  
 insert("dog", "cho")  
 insert("cat", "meo")  
 insert("pig", "heo")  
 insert("car", "o to")  
 print(get("fish"))  
  
 # in ra toàn bộ từ điển  
 for i in range(0, len(en\_hash\_table)):  
 if en\_hash\_table[i] == "":  
 print(str(i) + ". ")  
 else:  
 print(str(i) + ". " + en\_hash\_table[i] + " = " + vi\_hash\_table[i])

Câu 6. Anh Nam hiện có một mảng các số nguyên A. Với mỗi chỉ số i, anh ta muốn  
tìm tích của ba số lớn nhất trong khoảng [1, i], nếu không có kết quả như yêu cầu thì trả  
CâuChú ý: các số trong mảng A có thể giống nhau nhưng chúng phải khác nhau về chỉ số.  
Ví dụ: A = {1, 2, 3, 3, 4, 5}, kết quả là: -1, -1, 6, 6, 24, 60.

#include "iostream"

#include "vector"

#include "set"

using namespace std;

int main() {

int n;

cin >> n;

vector<int> v = vector<int>(n);

set<int> min = set<int>();

for (int &i: v) {

cin >> i;

}

for (int i: v) {

min.insert(i);

if (min.size() < 3) {

cout << "-1 ";

continue;

}

else if (min.size() > 3) {

min.erase(min.begin());

}

cout << \*min.begin() \* (\*min.begin() + 1) \* (\*min.begin() + 2) << ' ';

}

}

Câu 7. Cài đặt cấu trúc dữ liệu hàng đợi ưu tiên sử dụng cấu trúc đống (heap).  
a. Hàng đợi ưu tiên bao gồm các chức năng: chèn phần tử insert(), lấy giá trị của  
phần tử ở đầu hàng đợi top(), loại bỏ phần tử ra khỏi hàng đợi ưu tiên pop().  
b. Minh họa bằng hình vẽ thao tác chèn lần lượt vào hàng đợi các giá trị 8, 3, 9, 1,  
7, 5, 6, 4 và rút tất cả các phần tử ra khỏi hàng đợi.

#include "iostream"

#include "vector"

using namespace std;

void heapify(vector<int> &pQueue, int i) {

int n = pQueue.size();

int max;

int left = 2 \* i + 1;

int right = 2 \* i + 2;

if (left < n && pQueue[left] > pQueue[i]) {

max = left;

}

else {

max = i;

}

if (right < n && pQueue[right] > pQueue[max]) {

max = right;

}

if (max != i) {

swap(pQueue[i], pQueue[max]);

heapify(pQueue, max);

}

}

void push(vector<int> &pQueue, int value) {

int i = pQueue.size();

pQueue.push\_back(value);

while (i > 0 && pQueue[i] > pQueue[(i - 1) / 2]) {

swap(pQueue[(i - 1) / 2], pQueue[i]);

i = (i - 1) / 2;

}

}

void pop(vector<int> &pQueue) {

if (pQueue.empty()) {

cout << "Empty priority queue";

}

else {

swap(pQueue.back(), pQueue[0]);

pQueue.pop\_back();

heapify(pQueue, 0);

}

}

int top(vector<int> pQueue) {

return pQueue[0];

}

int main() {

vector<int> pQueue = vector<int>(0);

push(pQueue, 8);

push(pQueue, 3);

push(pQueue, 9);

push(pQueue, 1);

push(pQueue, 7);

push(pQueue, 5);

push(pQueue, 6);

push(pQueue, 4);

while (!pQueue.empty()) {

cout << top(pQueue) << ' ';

pop(pQueue);

}

}

Câu 8. Viết chương trình 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 theo  
thứ tự tăng dần, trung vị là số chính giữa mảng nếu mảng có số lẻ phần tử, là trung bình  
của hai số chính giữa mảng nếu mảng có số chẵn phần tử.

#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];

}

}

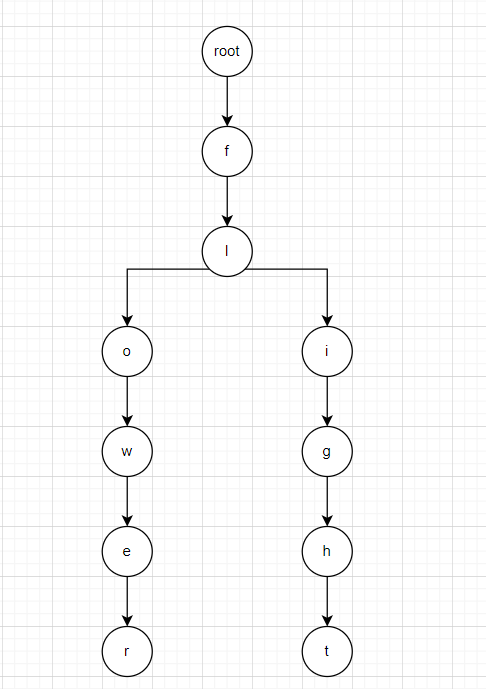
int main() {

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

countingSort(v);

}

Câu 9. 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”.



#include <iostream>

#include <vector>

using namespace std;

struct tree {

int count;

vector<tree \*> child;

bool hasWord;

tree() {

count = 0;

child.resize(26, nullptr);

hasWord = false;

}

};

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

string maxPrefix = "";

void add(tree \*&t, string s, int count) {

if (t == nullptr) {

t = new tree();

}

t->count++;

if (count < s.size() - 1) {

add(t->child[s[count + 1] - 'a'], s, count + 1);

}

else {

t->hasWord = true;

}

}

void findAll(tree \*t, string s) {

if (t->hasWord) {

cout << s << ' ';

}

for (int i = 0; i < t->child.size(); i++) {

if (t->child[i] != nullptr) {

findAll(t->child[i], s + (char) (i + 'a'));

}

}

if (t->count == keys.size() && s.length() > maxPrefix.length()) {

maxPrefix = s;

}

}

int main() {

tree \*t = new tree();

for (string i: keys) {

add(t->child[i[0] - 'a'], i, 0);

}

findAll(t, "");

cout << "\n" << maxPrefix;

return 0;

}

Câu 10. Cài đặt cấu trúc dữ liệu cây nhị phân tìm kiếm sử dụng ngôn ngữ lập trình C++  
hoặc Python, trong đó mỗi nút chứa thông tin về một sinh viên bao gồm số báo danh,  
họ tên, giới tính, quê quán, lớp, điểm trung bình. Cấu trúc dữ liệu này cho phép chèn  
sinh viên mới, tìm sinh viên và xóa sinh viên theo số báo danh.

#include <iostream>

using namespace std;

class SinhVien {

public:

string Sbd;

string HoTen;

string GioiTinh;

string QueQuan;

string Lop;

double Dtb;

SinhVien() {}

SinhVien(string sbd, string hoTen, string gioiTinh, string queQuan, string lop, double dtb) {

Sbd = sbd;

HoTen = hoTen;

GioiTinh = gioiTinh;

QueQuan = queQuan;

Lop = lop;

Dtb = dtb;

}

};

class Tree {

public:

SinhVien elem;

Tree \*left;

Tree \*right;

Tree(SinhVien e) {

elem = e;

left = nullptr;

right = nullptr;

}

};

void insert(Tree \*&T, SinhVien x) {

if (T == nullptr) {

T = new Tree(x);

}

else {

if (T->elem.Sbd > x.Sbd) {

insert(T->left, x);

}

else if (T->elem.Sbd < x.Sbd) {

insert(T->right, x);

}

else {

return;

}

}

}

SinhVien max(Tree \*T) {

if (T->right) {

return max(T->right);

}

else {

return T->elem;

}

}

SinhVien min(Tree \*T) {

if (T->left) {

return min(T->left);

}

else {

return T->elem;

}

}

void remove(Tree \*&T, string x) {

if (T->elem.Sbd == x) {

if (T->left == nullptr) {

T = T->right;

}

else if (T->right == nullptr) {

T = T->left;

}

else {

SinhVien sv = max(T->left);

T->elem = sv;

remove(T->left, sv.Sbd);

}

}

else if (T->elem.Sbd > x) {

remove(T->left, x);

}

else {

remove(T->right, x);

}

}

void search(Tree \*T, string x) {

SinhVien res;

while (T != nullptr) {

res = T->elem;

if (x < T->elem.Sbd) {

T = T->left;

}

else if (x > T->elem.Sbd) {

T = T->right;

}

else {

cout << res.Sbd << ' ' << res.HoTen << "\n";

return;

}

}

cout << "-1\n";

}

int main() {

Tree \*T = nullptr;

insert(T, SinhVien("001", "aaa", "nam", "ha noi", "cntt", 1));

insert(T, SinhVien("004", "ddd", "nam", "ha noi", "cntt", 4));

insert(T, SinhVien("003", "ccc", "nam", "ha noi", "cntt", 3));

insert(T, SinhVien("002", "bbb", "nam", "ha noi", "cntt", 2));

remove(T, "003");

search(T, "003");

search(T, "002");

return 0;

}

Câu 11. Giải phương trình 3x + 2y + z + q = 34 bằng thuật toán tối ưu bầy đàn PSO.(python)

# Câu 11. Giải phương trình 3x + 2y + z + q = 34 bằng thuật toán tối ưu bầy đàn PSO.  
  
import random  
  
  
class PSO:  
 def \_\_init\_\_(self, a, b, c, d, s):  
 self.\_a = a  
 self.\_b = b  
 self.\_c = c  
 self.\_d = d  
 self.\_s = s  
  
 self.vel = [[], [], [], []]  
 self.newSwarm = [[], [], [], []]  
 self.localSwarm = [[], [], [], []]  
  
 self.newFitness = [] # fit moi  
 self.localFitness = [] # fit cu  
  
 self.gb = [] # toan cuc  
 self.pb = [[], [], [], []] # cuc bo  
  
 def khoitao(self):  
 for item in range(0, 4):  
 for i in range(0, 4):  
 self.vel[item].insert(i, random.uniform(1, 21))  
 self.localSwarm[item].insert(i, random.randrange(1, 21))  
 self.pb = list(self.localSwarm)  
  
 def khoitaofit(self):  
 for item in range(0, 4):  
 t = self.\_a \* self.localSwarm[item][0] + self.\_b \* self.localSwarm[item][1] + self.\_c \* \  
 self.localSwarm[item][2] + self.\_d \* self.localSwarm[item][3]  
 self.localFitness.insert(item, abs(t - self.\_s))  
  
 def tim\_fit(self):  
 self.newFitness = []  
 for item in range(0, 4):  
 t = self.\_a \* self.newSwarm[item][0] + self.\_b \* self.newSwarm[item][1] + self.\_c \* self.newSwarm[item][  
 2] + self.\_d \* self.newSwarm[item][3]  
 self.newFitness.insert(item, abs(t - self.\_s))  
  
 def update\_better\_fit\_swarm(self):  
 for item in range(0, 4):  
 if self.localFitness[item] > self.newFitness[item]:  
 self.localFitness[item] = self.newFitness[item]  
 self.localSwarm[item] = list(self.newSwarm[item])  
 self.pb[item] = self.newSwarm[item]  
  
 def tim\_gb(self):  
 for item in range(0, 4):  
 if self.localFitness[item] == min(self.localFitness):  
 self.gb = self.localSwarm[item]  
  
 def update\_particle(self):  
 self.newSwarm = [[], [], [], []]  
  
 for item in range(0, 4):  
 for i in range(0, 4):  
 w = 0.7  
 c1 = c2 = 1.5  
 r1 = random.uniform(-1, 1)  
 r2 = random.uniform(-1, 1)  
 x = self.localSwarm[item][i]  
 pb = self.pb[item][i]  
 gb = self.gb[i]  
 self.vel[item][i] = w \* self.vel[item][i] + c1 \* r1 \* (pb - x) + c2 \* r2 \* (gb - x)  
 self.newSwarm[item].insert(i, round(x + self.vel[item][i], 0))  
  
 while (0 <= self.newSwarm[item][i] <= 20) == False: # bien nam ngoai pvi [0,20]  
 r1 = random.uniform(-1, 1)  
 r2 = random.uniform(-1, 1)  
 self.vel[item][i] = w \* self.vel[item][i] + c1 \* r1 \* (pb - x) + c2 \* r2 \* (gb - x)  
 self.newSwarm[item][i] = x + self.vel[item][i]  
  
 def main(self):  
 self.khoitao()  
 self.khoitaofit()  
 self.tim\_gb()  
  
 while 0 not in self.localFitness:  
 self.update\_particle()  
 self.tim\_fit()  
 self.update\_better\_fit\_swarm()  
 self.tim\_gb()  
 print(self.gb)  
  
  
a = PSO(3, 2, 1, 1, 34)  
a.main()

Câu 12. Giải phương trình 3x + 2y + z + q = 34 bằng thuật toán di truyền (genetic  
algorithm).

#include "iostream"

#include "cstdlib"

#include "ctime"

#include "vector"

#include "set"

using namespace std;

vector<vector<int>> chromosomeList = vector<vector<int>>(0);

set<vector<int>> result;

const int EQUATION\_RESULT = 34;

const int ESTIMATE\_NUMBER\_OF\_RESULT = 10; //thay Ä‘á»•i biáº¿n nÃ y Ä‘á»ƒ chÆ°Æ¡ng trÃ¬nh cháº¡y ra khoáº£ng bao nhiÃªu káº¿t quáº£

//hÃ m táº¡o 1 nhiá»…m sáº¯c thá»ƒ má»›i

vector<int> generateNewChromosome() {

vector<int> v = vector<int>(0);

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

v.push\_back(rand() % 10);

}

return v;

}

void checkResult(vector<int> v) {

if (3 \* v[0] + 2 \* v[1] + v[2] + v[3] == EQUATION\_RESULT) {

result.insert(v);

}

}

//hÃ m táº¡o nhiá»…m sáº¯c thá»ƒ ban Ä‘áº§u

void generateInitChromosome() {

chromosomeList.clear();

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

chromosomeList.push\_back(generateNewChromosome());

checkResult(chromosomeList[i]);

}

}

int getSum(vector<int> v) {

int res = 0;

for (int i: v) {

res += i;

}

return res;

}

//tÃ¬m khoáº£ng cÃ¡ch giá»¯a káº¿t quáº£ nhiá»…m sáº¯c thá»ƒ vÃ  káº¿t quáº£ bÃ i toÃ¡n

int getScore(int x) {

return 100 - abs(x - EQUATION\_RESULT);

}

void deleteLowestScoreChromosome(vector<int> scoreList) {

int minPos = 0;

for (int i = 1; i < scoreList.size(); i++) {

if (scoreList[i] < scoreList[minPos]) {

minPos = i;

}

}

chromosomeList.erase(chromosomeList.begin() + minPos);

}

void shuffleChromosomeList() {

int pos1 = rand() % 4;

int pos2;

do {

pos2 = rand() % 4;

} while (pos1 == pos2);

swap(chromosomeList[pos1], chromosomeList[pos2]);

}

//táº¡o thÃªm 2 con tá»« 2 nhiá»…m sáº¯c thá»ƒ Ä‘Ã£ cÃ³

void insertChildren(vector<int> father, vector<int> mother) {

int pos = rand() % 4;

for (int i = pos; i < father.size(); i++) {

swap(father[i], mother[i]);

}

chromosomeList.push\_back(father);

chromosomeList.push\_back(mother);

}

//táº¡o Ä‘á»™t biáº¿n

void makeMutate(vector<int> v) {

int pos = rand() % 4;

int mutate;

do {

mutate = rand() % 10;

} while (v[pos] == mutate);

v[pos] = mutate;

chromosomeList.push\_back(v);

}

void printVector(vector<int> v) {

for (int i: v) {

cout << i << ' ';

}

cout << '\n';

}

void printResult() {

cout << "result:\n";

for (auto i: result) {

printVector(i);

}

}

void solve() {

// táº¡o 4 nhiá»…m sáº¯c thá»ƒ Ä‘áº§u tiÃªn

generateInitChromosome();

int count = 0;

while (result.size() < ESTIMATE\_NUMBER\_OF\_RESULT) {

count++;

shuffleChromosomeList();

for (int i = 0; i < 4; i += 2) {

insertChildren(chromosomeList[i], chromosomeList[i + 1]);

}

for (int i = 0; i < 8; i++) {

makeMutate(chromosomeList[i]);

}

vector<int> chromosomeSumList = vector<int>(0);

for (vector<int> i: chromosomeList) {

chromosomeSumList.push\_back(getSum(i));

}

vector<int> scoreList = vector<int>(0);

for (int i: chromosomeSumList) {

scoreList.push\_back(getScore(i));

}

for (int i = 0; i < 12; i++) {

deleteLowestScoreChromosome(scoreList);

}

int resultCount = result.size();

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

checkResult(chromosomeList[i]);

}

if (result.size() != resultCount) {

count = 0;

generateInitChromosome();

}

// táº¡o 1 bá»™ nhiá»…m sáº¯c thá»ƒ má»›i náº¿u bá»™ Ä‘ang dÃ¹ng tÃ¬m lÃ¢u ra káº¿t quáº£

else if (count > 500) {

count = 0;

generateInitChromosome();

}

}

printResult();

}

int main() {

srand(time(nullptr));

solve();

return 0;

}

**Câu** 13. 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.  
Ví dụ, cho 4 trạm xăng với khối lượng xăng và khoảng cách tới trạm kế tiếp là các cặp  
giá trị {4, 6}, {6, 5}, {7, 3} và {4, 5}. Điểm đầu tiên mà từ đó xe tải có thể thực hiện  
chuyến đi vòng tròn là trạm xăng thứ hai. Đầu ra phải là “start = 1” (chỉ số của trạm  
xăng thứ hai).

#include "iostream"

#include "vector"

using namespace std;

bool kiemtra(vector<int> chenhlech, int vitri) {

int xang = 0;

int size = chenhlech.size();

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

xang += chenhlech[(vitri + i) % size];

if (xang < 0) {

return false;

}

}

return true;

}

int main() {

int n;

cin >> n;

vector<int> chenhlech = vector<int>(n);

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

int khoiluong;

int khoangcach;

cin >> khoiluong >> khoangcach;

chenhlech[i] = khoiluong - khoangcach;

}

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

bool kt = kiemtra(chenhlech, i);

if (kt) {

cout << i;

return 0;

}

}

cout << -1;

}

**Câu** 14. Cho một mảng số nguyên và một số nguyên k. Tìm số lớn nhất trong từng mảng  
con liên tiếp kích thước k.  
Ví dụ, với mảng: arr[] = {1, 2, 3, 1, 4, 5, 2, 3, 6}, k = 3  
Kết quả là: 3 3 4 5 5 5 6, vì  
Số lớn nhất của {1, 2, 3} is 3  
Số lớn nhất của {2, 3, 1} is 3  
Số lớn nhất của {3, 1, 4} is 4

#include "iostream"

#include "deque"

#include "vector"

using namespace std;

int main() {

deque<int> d;

int n, k;

cin >> n >> k;

vector<int> v(n + 1);

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

cin >> v[i];

for (int j = d.size() - 1; j >= 0; j--) {

if (v[d.back()] < v[i]) {

d.pop\_back();

}

else {

break;

}

}

d.push\_back(i);

}

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

cin >> v[i];

for (int j = 0; j < d.size(); j++) {

if (d.front() <= i - k) {

d.pop\_front();

}

else {

break;

}

}

for (int j = d.size() - 1; j >= 0; j--) {

if (v[d.back()] < v[i]) {

d.pop\_back();

}

else {

break;

}

}

d.push\_back(i);

cout << v[d.front()] << ' ';

}

return 0;

}

**Câu** 15. 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 đặcbiệt được tính: a[i] % K. Bạn cần sắp xếp mảng này theo thứ tự giảm dần của các trọngsố 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';

}

}