Câu 1:Nêu ý tưởng thuật toán sắp xếp Radix Sort, minh họa A[]={10,15,18,63,52,100,25,50}

Ý tưởng

Radix Sort là thuật toán **sắp xếp không dựa trên so sánh**, dựa vào các chữ số của số nguyên (hoặc ký tự) để sắp xếp theo thứ tự. Thuật toán thực hiện sắp xếp **lần lượt theo từng chữ số**, bắt đầu từ chữ số ở hàng đơn vị, sau đó đến hàng chục, hàng trăm, v.v.

A={10,15,18,63,52,100,25,50}

B1.Sắp xếp các số theo hàng đơn vị

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 50 |  |  |  |  |  |  |  |  |  |
|  | 100 |  |  |  |  | 25 |  |  |  |  |
|  | 10 |  | 52 | 63 |  | 15 |  |  | 18 |  |
| Cơ số(Bt) | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

* A={10,100,50,52,63,15,25,18}

B2.Sắp xếp các số theo hàng chục

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 18 |  |  |  |  |  |  |  |  |
|  |  | 15 |  |  |  | 52 |  |  |  |  |
|  | 100 | 10 | 25 |  |  | 50 | 63 |  |  |  |
| Cơ số(Bt) | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

* A={100,10,15,18,25,50,52,63}

B3.Sắp xếp các số theo hàng trăm

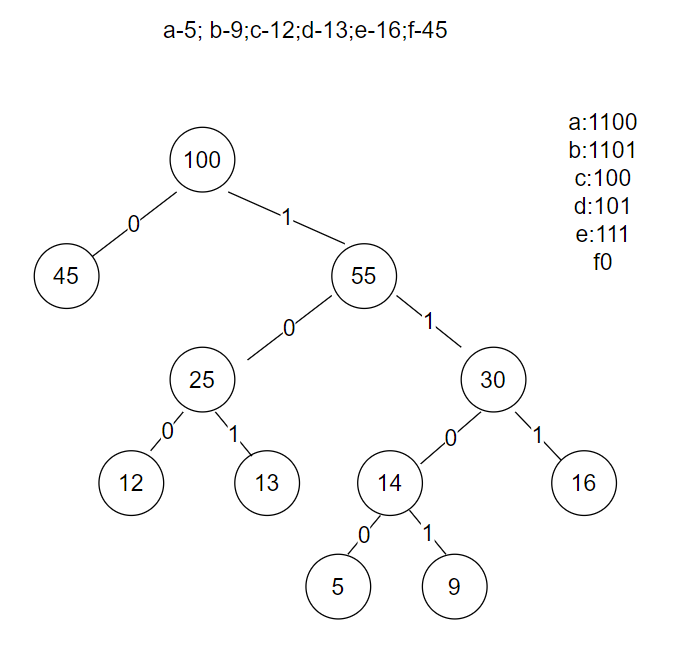
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 63 |  |  |  |  |  |  |  |  |  |
|  | 52 |  |  |  |  |  |  |  |  |  |
|  | 50 |  |  |  |  |  |  |  |  |  |
|  | 25 |  |  |  |  |  |  |  |  |  |
|  | 18 |  |  |  |  |  |  |  |  |  |
|  | 15 |  |  |  |  |  |  |  |  |  |
|  | 10 | 100 |  |  |  |  |  |  |  |  |
| Cơ số(Bt) | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

->A={10,15,18,25,50,52,63,100}

Câu 2:Đầu vào ch[]={‘a’, ‘b’, ‘c’ , ‘d’, ‘e’, ‘f’}, freq[]={5,9,12,13,16,45}

Đầu ra f-0,c-100, a-1100, b-1101,e-111

1. Vẽ cây Huffma b, viết chương trình c,,Minh họa



#include <iostream>

#include <queue>

#include <vector>

#include <map>

using namespace std;

struct HuffmanNode {

char ch;

int freq;

HuffmanNode \*left, \*right;

HuffmanNode(char c, int f) : ch(c), freq(f), left(NULL), right(NULL) {}

};

struct Compare {

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

return a->freq > b->freq;

}

};

void assignCodes(HuffmanNode\* root, string code, map<char, string>& huffmanCodes) {

if (!root) return;

if (root->ch != '\0') huffmanCodes[root->ch] = code;

assignCodes(root->left, code + "0", huffmanCodes);

assignCodes(root->right, code + "1", huffmanCodes);

}

int main() {

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

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

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

for (size\_t i = 0; i < ch.size(); i++) {

pq.push(new HuffmanNode(ch[i], freq[i]));

}

while (pq.size() > 1) {

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

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

HuffmanNode\* newNode = new HuffmanNode('\0', left->freq + right->freq);

newNode->left = left;

newNode->right = right;

pq.push(newNode);

}

map<char, string> huffmanCodes;

assignCodes(pq.top(), "", huffmanCodes);

for (auto pair : huffmanCodes) {

cout << pair.first << ": " << pair.second << endl;

}

return 0;

}

Câu 3: : Viết chương trình chèn dãy giá trị {4371,1323,6173,4199,4344,9679,1989} vào bảng băm thăm dò bậc 2 với hàm băm hash(x)=x%10 và vẽ hình minh họa chèn từng giá trị trên vào bảng băm thăm dò bậc 2

#include "bits/stdc++.h"

using namespace std;

const int TABLE\_SIZE = 10;

int hashFunction(int x){

return x%TABLE\_SIZE;

}

void insert(vector<int>& hashTable, int value){

int index = hashFunction(value);

int OGIndex = index;

int i = 1;

while(hashTable[index] != -1){

index = (OGIndex + i \* i) % TABLE\_SIZE;

i++;

if(index >= TABLE\_SIZE){

cout << "Bang bam day, ko the chen: " << value << endl;

return;

}

}

hashTable[index] = value;

cout << "Chen gia tri " << value << " vao vi tri " << index << endl;

}

void printHashTable(const vector<int>& hashTable) {

cout << "\nBang bam:\n";

for (int i = 0; i < TABLE\_SIZE; i++) {

if (hashTable[i] == -1) {

cout << "[" << i << "] (trong)\n";

} else {

cout << "[" << i << "] " << hashTable[i] << endl;

}

}

}

int main(){

vector<int> hashTable(TABLE\_SIZE, -1);

vector<int> values = {4371, 1323, 6173, 4199, 4344, 9679, 1989};

for (int value : values) {

insert(hashTable, value);

}

printHashTable(hashTable);

#### }

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 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 |