ĐẠI HỌC BÁCH KHOA HÀ NỘI

**TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

----- 🙡 🕮 🙣 -----



**BÁO CÁO PROJECT I**

***Báo cáo tuần 4***

Giảng viên: **Ngô Lam Trung**

Sinh viên thực hiện: Nguyễn Quang Long

Mã số sinh viên: 20194320

**Hà Nội, năm 2021**

Nội dung đã hoàn thành:

**82. New Numeral System**

a. Đề bài:

Cho ánh xạ: A -> 0, B ->1, C->2,…., Z->25

Cho 1 chữ cái, tìm tất cả các cặp (không tính TH đổi vị trí các phần tử) sao cho tổng của các cặp đó bằng chữ cái đã cho và đưa ra màn hình tất cả các cặp đó.

b. Giải pháp:

Khi đầu vào là 1 chữ cái C thì sẽ có n= round((C-‘A’)/2) cặp

Duyệt từ i = 0 đến đến n, phần tử đầu tiên trong mỗi cặp là i+’A’ và phần tử cuối là C-i. Với mỗi 2 phần tử này ta được 1 cặp cần tìm.

c. Code:

vector<string> solution(char number) {

    string sum;

    vector<string> vec;

    for(int i =0;i<=(int)round((number-'A')/2);i++){

        sum="";

        sum+=(char)(i+'A');

        sum+= " + ";

        sum+=(char)(number-i);

        vec.push\_back(sum);

    }

    return vec;

}

**83. Cipher 26**

a. Đề bài

Cho ánh xạ a-0, b-1,c-2,…,z-25. Với mỗi chuỗi ký tự (gồm các chữ cái tiếng anh viết thường) sẽ được mã hóa như sau:

Ký tự thứ i sẽ được mã hóa thành số dư của tổng ánh xạ ký tự sang số của i ký tự đầu tiên trong chuỗi chia cho 26 rồi ánh xạ ngược lại thành ký tự

Ví dụ: cho chuỗi “thi”

Ký tự đầu tiên: ‘t’=19%26=19-> ‘t’

Ký tự thứ 2: ‘h’(=7) + ‘t’ (=19) = 26%26=0->’a’

Ký tự thứ 3: ‘i’(=8) + ‘h’(=7) + ‘t’ (=19) = 34 %26 = 8 - > ‘i’

* Chuỗi “thi” sẽ được mã hóa thành “tai”

Cho một chuỗi đã được mã hóa, tìm ngược lại chuỗi ban đầu.

b. Giải pháp.

Gọi i là ký tự ở vị trí i trong chuỗi ban đầu, mod[i] là ký tự được mã hóa từ i, sum[i] là tổng của i-1 ký tự đầu tiên của chuỗi ban đầu.

* (i+sum[i]) %26 = mod[i] ⬄ i%26+sum[i]%26=mod[i]

Nhiệm vụ của ta là tìm ký tự i

TH1: Nếu mod[i] > sum[i] %26 thì i = mod-sum

TH2: Nếu mod[i] <= sum[i]%26 thì i= mod+26-sum

c. Code

string solution(string message) {

    string ans;

    int sum=0;

    for(int i=0;i<message.length();i++){

        int mod = message[i]-'a';

        char v;

        if(sum%26>mod) v=mod+26-sum%26+'a';

        else v=mod-sum%26+'a';

        sum+=(int)(v-'a');

        ans+=v;

    }

    return ans;

}

**84. Stolen Lunch**

a. Đề bài

Cho ánh xạ 0-a, 1-b,…,9-j và ngược lại. Cho 1 xâu, tìm ánh xạ của xâu đó.

b. Giải pháp

Nếu ký tự đang xét của xâu là chữ số thì chuyển sang chữ cái tương ứng. Nếu ký thự đang xét nằm trong ‘a’->’j’ thì chuyển sang chữ số tương ứng.

c. Code

string solution(string note) {

    for(auto &c: note){

        if(c >= '0' && c <= '9')

            c = c-'0' + 'a';

        else if(c >= 'a' && c <= 'j')

            c = c-'a' + '0';

    }

    return note;}

**85. Higher Version**

a. Đề bài

Cho 2 xâu là số chỉ phiên bản bao gồm các số thập phân không âm ngăn cách với nhau bởi dấu “.” Cả 2 xâu đều chứa số lượng trường số bằng nhau và không có số 0 ở đầu mỗi trường số.

Ví dụ: 1.0.5 và 1.2.3.

100.20.3 thay vì 100.020.003

Trả lại true nếu phiên bản thứ 1 cao hơn phiên bản thứ 2 và ngược lại.

b. Giải pháp

Tách các xâu thành các mảng bởi ký tự “.” Rồi so sánh lần lượt các phần tử ở vị trí giống nhau của 2 mảng từ trái qua, nếu phần tử của mảng nào lớn hơn thì mảng đó lớn hơn

c. Code

function solution(ver1, ver2) {

    var arr1 = ver1.split('.');

    var arr2 = ver2.split('.');

    for(var i=0; i<arr1.length ;i++){

        if(arr1[i]\*1>arr2[i]\*1) return true;

        if(arr1[i]\*1<arr2[i]\*1) return false;

    }

    return false;

}

**86. Decipher**

a. Đề bài

Cho 1 xâu là mã hóa theo bảng mã ASCII của 1 xâu gồm các ký tự tiếng Anh viết thường. Hãy tìm xâu chưa được mã hóa

Ví dụ "10197115121" => "easy" vì 101 = ‘e’, 97=’a’, 115=’s’, 121=’y’

b. Giải pháp

Tách các xâu con rồi chuyển về ký tự ASCII tương ứng. Nếu xâu con bắt đầu bằng ‘9’ thì xâu đó có độ dài = 2, nếu bắt đầu = ‘1’ thì xâu đó có độ dài bằng 3.

c. Code

string solution(string cipher) {

    int first=0;

    string ans;

    while(first<cipher.length()){

        if(cipher[first]=='9') {

            ans+=(char)stoi(cipher.substr(first,2));

            first+=2;

    }

        if(cipher[first]=='1') {

            ans+=(char)stoi(cipher.substr(first,3));

            first+=3;

    }

}

return ans;

}

**87. Alphanumeric Less**

a. Đề bài

Cho 2 xâu gồm các chữ cái và các số. So sánh 2 xâu theo quy tắc sau:

* Đầu tiên chia chuỗi thành các phần tử gồm chữ cái và số (không phải chữ số), ví dụ "ab01c004" thành [a, b, 01, c, 004] sau đó sẽ so sánh lần lượt các cặp phần tử tương ứng của 2 xâu đến khi tìm được cặp phần tử đầu tiên khác nhau.
* Nếu 1 chữ cái được so sánh với 1 chữ cái khác thì tuân theo thứ tự bảng chữ cái (1)
* 1 số thì nhỏ hơn 1 chữ cái (2)
* 2 số được so sánh thì giá trị của chúng được so sánh, nếu có các số 0 ở đầu thì được bỏ qua.(3)
* Tại lần so sánh thứ i, nếu xâu 1 không còn phần tử mà xâu 2 vẫn còn thì xâu 1 nhỏ hơn xâu 2.
* Nếu tất cả các phần tử đều bằng nhau thì so sánh số lượng số 0 đứng trước mỗi số ở mỗi phần tử, xâu nào có nhiều số 0 hơn thì nhỏ hơn.
* Nếu không thuộc tất cả các trường hợp trên thì 2 xâu bằng nhau

Trả về true nếu xâu 1 nhỏ hơn xâu 2 và false nếu ngược lại.

b. Giải pháp

Đầu tiên chia xâu thành các xâu con gồm các chữ cái và chữ số riêng biệt, ví dụ “aab01aa” -> “aab”,”01”,”aa” rồi đưa vào mảng. Lúc này ta được 2 mảng vt1 và vt2

So sánh lần lượt các phần tử tương ứng trong 2 mảng để xem có thể so sánh ngay với quy tắc (1), (2) và (3) không. Nếu theo cách so sánh này các phần tử đều bằng nhau thì so sánh kích thước 2 mảng. Nếu kích thước 2 mảng bằng nhau thì xem trong 2 mảng, ở các phần tử tương ứng, phần tử nào có nhiều số 0 đứng trước hơn thì nhỏ hơn. Nếu vẫn không thể chỉ ra xâu nào lớn hơn thì lúc này 2 xâu thực sự bằng nhau và trả về false.

c. Code

#include <vector>

vector<string> toVT(string s1){

    int first=0;

    vector<string> vt1;

    for(int i=0; i<s1.length();i++){

        if((isdigit(s1[first]) && isdigit(s1[i]) && !isdigit(s1[i+1]))

        || ( isalpha(s1[first]) && isalpha(s1[i]) &&!isalpha(s1[i+1]))){

        string str1=s1.substr(first,i-first+1);

        vt1.push\_back(str1);

        first=i+1;

    }

}

return vt1;

}

int compare(string s1, string s2){

    if(isdigit(s1[0])&&isdigit(s2[0])){

        int ss1= stoi(s1);

        int ss2 = stoi(s2);

        if(ss1>ss2) return 1;

        else if(ss1<ss2) return -1;

        else return 0;

    }

    if(isalpha(s1[0])&&isalpha(s2[0])){

        if(s1>s2) return 1;

        else if(s1<s2) return -1;

        else return 0;

    }

    if(isdigit(s1[0])&& isalpha(s2[0])) return -1;

    else return 1;

}

bool solution(string s1, string s2) {

    vector<string> vt1=toVT(s1),vt2=toVT(s2);

    for(int i=0; i<min(vt1.size(),vt2.size());i++){

        if(compare(vt1[i],vt2[i])!=0) {

            if(compare(vt1[i],vt2[i])==1) return false;

            else return true;

        }

    }

    if(vt1.size()>vt2.size()) return false;

    if(vt1.size()<vt2.size()) return true;

    if(vt1.size()==vt2.size()){

        for(int i=0;i<max(s1.size(),s2.size());i++){

            if(s1[i]!=s2[i]) {

                if(s1[i]=='0') return true;

                else return false;

            }

        }

    }

      return false;

}

**88. Array Conversion**

a. Đề bài

Cho 1 mảng gồm 2^k số nguyên. Cho thuật toán sau:

-Tại các lần thứ hiện thứ 1,3,5,… sẽ thay thế các cặp phần tử liên tiếp trong mảng bằng tổng của chúng

- Tại các lần thực hiện thứ 2,4,6,… sẽ thay thế các cặp phần tử liên tiếp trong mảng bằng tích của chúng

- Dừng lại khi mảng chỉ còn 1 phần tử

Áp dụng thuật toán trên và trả về giá trị cuối cùng với mảng đã cho.

b. Giải pháp

Có 1 biến k để xác định thuật toán thực hiện lần thứ mấy?

Ở mỗi phần tử thứ 0,1,2,3,… sẽ được thay thế bằng tổng(hoặc tích) của các cặp (0-1), (2-3), (4-5), (6-7),… tổng quát, phần tử i/2 sẽ được thay thế bằng tổng hoặc tích của phần tử i và i+1. Sau đó giảm kích thước của mảng đi 1 nửa. Thực hiện đến khi mảng chỉ còn 1 phần tử.

c. Code

int solution(vector<int> a) {

    int k = 1;

while(a.size()>1) {

    for(int i = 0; i < a.size() - 1; i += 2){

        if(k%2!=0) a[i/2] = a[i]+a[i+1];

        else a[i/2] = a[i]\*a[i+1];

    }

   k++;

   a.resize(a.size() / 2);

}

   return a[0];

}

**89. Array Previous Less**

a. Đề bài

Cho 1 mảng các số nguyên, với mỗi phần tử nằm ở vị trí i hãy thay thế nó bằng phần tử gần nhất nằm bên trái có giá trị nhỏ hơn nó, nếu ko có thì thay thế bằng -1

b. Giải pháp

Sử dụng 2 vòng lặp, vòng lặp 1 để duyệt qua tất cả các phần tử trong mảng, vòng lặp 2 duyệt qua các phần tử đứng trước phần tử nằm ở vị trí i để tìm vị trí gần nhất có giá trị nhỏ hơn ở vị trí i.

c. Code

vector<int> solution(vector<int> items) {

    vector<int> ans={-1};

    for(int i=1;i<items.size();i++){

        int j=i-1;

        while(j>=0){

            if(items[j]<items[i]) {ans.push\_back(items[j]);

            break;

        }

        j--;

        if(j<0) ans.push\_back(-1);

    }

}

return ans;

}

**90. Pair of Shoes**

a. Đề bài

Cho 1 mảng gồm các cặp <type,size> thể hiện loại (0 với bên trái hay 1 với bên phải) và kích thước của 1 chiếc giày. Có thể ghép tất cả các chiếc giày đó thành các cặp cùng loại được không? (không dư chiếc giày nào)

b. Giải pháp

Duyệt tất cả các chiếc giày, với 2 chiếc tạo thành 1 đôi thì đổi type của chúng thành 3. Nếu ở 1 chiếc giày nào đó mà type khác 3 và không thể tìm thấy chiếc giày khác để tạo thành 1 cặp thì trả về false, trả về true nếu duyệt hết tất cả các chiếc giày.

c. Code

bool solution(vector<vector<int>> shoes) {

    if(shoes.size()%2!=0) return false;

    int pairs=0;

    for(int i=0;i<shoes.size();i++){

        bool found =false;

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

            if(shoes[i][1]==shoes[j][1] && shoes[i][0]!=shoes[j][0] && shoes[i][0]!=3 && shoes[j][0] !=3) {

                shoes[i][0] = 3;

                shoes[j][0] = 3;

                found = true;

                pairs++;

                break;

            }

        } if(!found&&shoes[i][0]!=3) return false;

    }

    return true;

}

**91. Combs**

a. Đề bài

Text

Description automatically generatedCho 2 xâu biểu diễn 2 chiếc lược, 2 chiếc lược này mất 1 số răng (“\*” thể hiện răng của lược và “.” thể hiện vị trí mất răng) , tính xem kích thước tối thiểu có thể có khi ghép 2 chiếc lược này với nhau (răng của chiếc lược này sẽ ghép vào vị trí mất răng của chiếc lược kia). 2 chiếc răng đầu và cuối mỗi chiếc lược luôn ko bị mất

b. Giải pháp

Chuyển dạng biểu diễn của 2 xâu thành bitmask với bit 1 tương ứng có răng và bit 0 tương ứng không có răng.

Ví dụ \*…\* => 10001

Đầu tiên ta sẽ dịch bit biểu diễn của 1 chiếc lược sang trái n bit (n là độ dài của chiếc lược thứ 2), đây là tình huống tệ nhất khi 2 chiếc lược không thể ghép được với nhau, lần lượt dịch chiếc lược thứ 2 sang trái 1 đơn vị (bằng cách dịch bit) nếu ghép được với chiếc lược thứ 1 (tương ứng phép AND = 0) thì tính toán độ dài của 2 chiếc lược ghép với nhau và cập nhật độ dài nhỏ nhất.

c. Code

int convert(string comb){

     int bit\_com=0;

    for(int i=0; i<comb.length()-1;i++){

        if(comb[i]=='\*') bit\_com|=1;

        bit\_com<<=1;

    } bit\_com|=1;

    return bit\_com;

}

int solution(string comb1, string comb2) {

    int longer\_comb=0, shorter\_comb=0;

    int longer\_length=max(comb1.length(),comb2.length());

    int shorter\_length=min(comb2.length(),comb1.length());

    if(comb1.length()>comb2.length()){

        longer\_comb=convert(comb1);

        shorter\_comb=convert(comb2);

    }

    else{

        longer\_comb=convert(comb2);

        shorter\_comb=convert(comb1);

    }

    longer\_comb<<=shorter\_length;

    int min = comb1.length()+comb2.length();

    int sum =min;

    for(int i=0;i<=sum;i++){

       if((longer\_comb&shorter\_comb)==0) {

           int len;

           if(i<=shorter\_length) len = sum-i;

           if(i>shorter\_length&&i<=longer\_length) len=longer\_length;

           if(i>longer\_length) len=i;

           if(len<min) min = len;

       }

       shorter\_comb<<=1;

    }

    return min;

}

**92. String Crossover**

a. Đề bài

Cho 1 mảng gồm các xâu inputString, và 1 xâu result. Có bao nhiêu cặp xâu từ inputString tạo thành xâu result theo quy tắc: mỗi ký tự ở vị trí i của xâu result có thể chọn ngẫu nhiên từ ký tự vị trí i của 1 trong 2 xâu chọn từ inputString. Cho biết các xâu trong inputString có độ dài bằng xâu result.

b. Giải pháp

Duyệt tất cả các cặp có thể có trong mảng inputString rồi duyệt lần lượt các vị trí trong xâu result xem có thể tạo ra từ cặp trên không.

c. Code

int solution(vector<string> inputArray, string result) {

    int count =0;

    bool isPair;

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

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

            isPair=true;

            for(int k=0;k<result.length();k++){

                if(result[k]!=inputArray[i][k]&&result[k]!=inputArray[j][k]) {

                    isPair=false;

                        break;}

            }

            if(isPair) count++;

        }

    }

    return count;

}

**93. Cyclic String**

a. Đề bài

Cho 1 chuỗi con của 1 chuỗi tuần hoàn, hỏi độ dài nhỏ nhất của chuỗi có thể tạo ra chuỗi tuần hoàn đó là bao nhiêu?

b. Giải pháp

Nếu độ dài của chuỗi có thể tạo ra chuỗi tuần hoàn là i thì ký tự tại vị trí j phải giống với ký tự ở vị trí j%i

VD bcabca => độ dài có thể tạo ra chuỗi tuần hoàn trong TH này chỉ có 3 (abc), khi đó ký tự ở vị trí [3] (b) = ký tự [3%3] =[0] (b) , tương tự [4] (c) = [4%3]=[1]

Như vậy, ta sẽ thử từ trường hợp i =1, rồi duyệt qua các ký tự trong xâu nếu thỏa mãn tính chất trên thì đó là trường hợp có độ dài nhỏ nhất, nếu không tiếp tục tăng i. imax=độ dài chuỗi đã cho.

c. Code

int solution(string s) {

    for (int i = 1;; i++) {

        bool check = true;

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

            if (s[j] != s[j % i]) {

                check = false;

                break;

            }

        }

        if(check) return i;

    }

**}**

**94. Beautiful Text**

a. Đề bài

Cho 1 xâu gồm các từ và các khoảng trắng, có thể thay các khoảng trắng bằng ký tự xuống dòng. Hỏi có thể chia đoạn xâu trên thành các xâu nhỏ theo quy tắc trên với độ dài các xâu bằng nhau và nằm trong đoạn [r,l] không?

b. Giải pháp

Duyệt lần lượt i từ r đến l, nếu độ dài của xâu +1 đã cho chia hết cho i+1 thì kiểm tra xem các vị trí cuối của mỗi xâu con (trừ xâu cuối) có phải là khoảng trắng không, nếu là khoảng trắng thì trả về true, nếu không có i thỏa mãn thì trả về false.

c. Code

bool solution(string inputString, int l, int r) {

    int len=inputString.length();

    for(int i=l;i<=r;i++){

        bool found=true;

        if((len+1)%(i+1)==0){

            for(int j=i; j< len ;j+=i+1){

                if(inputString[j]!=' ') {

                    found=false;

                    break;

                }

            }

            if(found)   return true;

        }

    }

    return false;

}

**95. Runner Meeting**

a. Đề bài

Cho 2 mảng, mảng thứ 1 thể hiện vị trí xuất phát của các người chạy bộ và mảng 2 thể hiện tốc độ của từng người tương ứng. Tìm số người nhiều nhất cùng gặp nhau tại 1 vị trí và thời điểm.

b. Giải pháp

Duyệt lần lượt từng người và tính toán vị trí và thời gian gặp của người này với những người còn lại (nếu có). Số người nhiều nhất cùng gặp nhau tại 1 thời điểm chính là vị trí và thời gian có số lần xuất hiện nhiều nhất.

c. Code

int solution(vector<int> startPosition, vector<int> speed) {

    map<pair<float, float>,int>:: iterator it;

    int max =-1;

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

        map<pair<float, float>,int> meets;

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

            if (speed[j] - speed[i] != 0) {

                float t = (startPosition[i] - startPosition[j]) / (float) (speed[j] - speed[i]);

                float x = startPosition[i] + t \* speed[i];

                if (t >= 0) meets[{x, t}]++;

            }

        }

        for(it = meets.begin();it!=meets.end();it++){

            if(max<it->second+1) max = it->second+1;

        }

    }

        return max;

}

**97. File naming**

a. Đề bài

Cho 1 mảng là các xâu thể hiện tên tệp. Các tên tệp không được giống nhau nên nếu tên nào trùng sẽ được thêm “(i)” với i là số lần trùng của nó.

b. Giải pháp

Duyệt lần lượt các xâu và thêm vào set, nếu xâu đã xuất hiện trong set thì thêm “(i)” vào sau xâu đó rồi thêm vào set, lặp lại đến khi không tìm thấy trong set nữa.

c. Code

std::vector<std::string> solution(std::vector<std::string> names) {

    vector<string> nm;

    set<string> s;

    for(string str: names) {

        int t{};

        string temp = str;

        while(s.find(temp)!=s.end())

            temp = str +"("+to\_string(++t)+")";

        s.insert(temp);

        nm.push\_back(temp);

    }

    return nm;

}

**98. Extract Matrix Column**

a. Đề bài

Cho 1 mảng là ma trận hình chữ nhật, hãy đưa ra cột thứ i+1 của ma trận đó.

b. Giải pháp

Ở hàng thứ j, phần tử ở cột thứ i+1 là matrix[j][i]

c. Code

vector<int> solution(vector<vector<int>> matrix, int column) {

    vector<int> col;

    for(int i=0;i<matrix.size();i++){

        col.push\_back(matrix[i][column]);

    }

    return col;

}

**99. Are Isomorphic?**

a. Đề bài

2 mảng được gọi là isomorphic nếu chúng có cùng số hàng và mỗi hàng có cùng số phần tử. Cho 2 mảng, kiểm tra xem chúng có isomorphic không.

b. Giải pháp

Kiểm tra 2 mảng có cùng số hàng không, nếu có, kiểm tra mỗi hàng có cùng số phần tử không.

c. Code

bool solution(vector<vector<int>> array1, vector<vector<int>> array2) {

    if(array1.size()==array2.size()){

        for(int i=0;i<array2.size();i++){

            if(array2[i].size()!=array1[i].size()) return false;

        }

        return true;

    }

    else return false;

}

**100. Reverse On Diagonals**

a. Đề bài

Cho 1 ma trận vuông kích thước nxn, đảo ngược vị các phần tử nằm trên mỗi đường chéo chính của ma trận

b. Giải pháp

Ở mỗi hàng thứ i, đổi chỗ các phần tử (matrix[i][i] và matrix[n-1-i]) và (matrix[i][n-1-i] và matrix[n-1-i][i])

c. Code

vector<vector<int>> solution(vector<vector<int>> matrix) {

    int n = matrix.size();

    for(int i=0;i<n/2;i++){

      swap(matrix[i][i], matrix[n-1-i][n-1-i]);

      swap(matrix[i][n-1-i],matrix[n-1-i][i]);

    }

    return matrix;

}