ĐẠI HỌC BÁCH KHOA HÀ NỘI

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

----- 🙡 🕮 🙣 -----



**BÁO CÁO PROJECT I**

***Báo cáo tuần 8***

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:

Từ câu 150 đến câu 154

Phần Intro: từ câu 1 đến câu 10

**150. Cells Joining**

a. Đề bài

Cho 1 vector string thể các hàng và cột và chứa các ký tự trong mỗi ô. Ví dụ

table = ["+----+--+-----+----+",

"|abcd|56|!@#$%|qwer|",

"|1234|78|^&=()|tyui|",

"+----+--+-----+----+",

"|zxcv|90|77777|stop|",

"+----+--+-----+----+",

"|asdf|~~|ghjkl|100$|",

"+----+--+-----+----+"]

Cho mảng Coords gồm 2 phần tử là tọa độ của 2 vị trí {hàng, cột} của bảng trên. Hãy hợp nhất các hàng và cột từ vị trí 1 đến vị trí 2 của bảng.

Ví dụ với Coords = [[2,0],[1,1]]. Hợp nhất từ vị trí hàng 2 cột 0 đến hàng 1 cột 1

Được:

["+----+--+-----+----+",

"|abcd|56|!@#$%|qwer|",

"|1234|78|^&=()|tyui|",

"+----+--+-----+----+",

"|zxcv 90|77777|stop|",

"| +-----+----+",

"|asdf ~~|ghjkl|100$|",

"+-------+-----+----+"]

b. Giải pháp

Ta sẽ xác định vị trí của dấu + trong bảng tương ứng với thứ tự được cho trong mảng Coords để xác định được vị trí cần hợp nhất trong bảng đã cho

Sau khi đã xác định được các ô cần hợp nhất, ta sẽ duyệt lần lượt từng hàng, từng cột ở giữa các ô đó (không tính vị trí biên) , nếu là các ký tự phân cách (+,|,-) thì thay thế bằng khoảng trắng. Nếu các ký tự ở vị trí biên trái cùng hoặc phải cùng thì thay thế bằng |.

c. Code

std::vector<std::string> solution(std::vector<std::string> t, std::vector<std::vector<int>> co) {

    vector<int> r,c;

    int x=0,y=0,topRow,botRow,leftCol,rightCol;

    for(auto e:t) {

        if(e[0]=='+') r.push\_back(x);

        x++;

        }

    for(auto e:t[0]) {

        if(e=='+') c.push\_back(y);

        y++;

        }

    topRow=r[min(co[0][0],co[1][0])];

    botRow=r[max(co[0][0],co[1][0])+1];

    leftCol=c[min(co[0][1],co[1][1])];

    rightCol=c[max(co[0][1],co[1][1])+1];

    for(x=topRow+1;x<botRow;x++) {

        for(y=leftCol+1;y<rightCol;y++) {

            if(t[x][y]=='-'||t[x][y]=='|'||t[x][y]=='+') t[x][y]=' ';

            if(topRow==0) t[topRow][y]='-';

            if(botRow==t.size()-1) t[botRow][y]='-';

        }

        if(leftCol==0) t[x][leftCol]='|';

        if(rightCol==t[0].length()-1) t[x][rightCol]='|';

    }

    return t;

}

**151. First Operation Character**

a. Đề bài

Cho 1 biểu thức toán học gồm các toán hạng và toán tử. Xác định xem toán tử nào được tính toán đầu tiên?

Ví dụ

expr = "(2 + 2) \* 2" => solution(expr) = 3

Do dấu + nằm trong ngoặc nên được tính đầu tiên và nằm ở vị trí thứ 3 trong xâu nên đầu ra là 3.

b. Giải pháp

Duyệt lần lượt từ trái sang phải xâu đó và xác định biểu thức chứa dấu ngoặc nào được tính đầu tiên (tức cặp đóng mở ngoặc trong cùng) và lưu kiểu toán tử nằm trong cặp ngoặc đó cũng như vị trí của toán tử đó. Cặp ngoặc nào nằm trong nhất sẽ được tính đầu tiên, trả về vị trí của toán tử trong ngoặc đó. Nếu có nhiều hơn 1 cặp ngoặc nằm trong nhất thì cặp ngoặc chứa toán tử \* và / sẽ được tính đầu tiên, khi đó trả về vị trí của toán tử trong cặp ngoặc đó.

c. Code

int solution(std::string expr) {

    int b = 0, p = 0, pmul, padd;

    int mul = -1, add = -1;

    for (char c : expr) {

        if (c == '(') {

            b++;

        } else if (c == ')') {

            b--;

        } else if (c == '\*' && mul < b) {

            mul = b;

            pmul = p;

        } else if (c == '+' && add < b) {

            add = b;

            padd = p;

        }

        p++;

    }

    return add > mul ? padd : pmul;

}

**152. Count Elements**

a. Đề bài

Cho 1 xâu inputString, xác định xem trong xâu đó có bao nhiêu biến dữ liệu nguyên thủy.

Ví dụ inputString = "[ "one", 2, "three" ]" => output: 3

b. Giải pháp

Sử dụng regex để tách xâu ra thành các mảng nếu:

* Có cặp đóng mở ngoặc kép “ “
* Là 1 chuỗi số
* Là true / false

Khi đó kết quả là độ dài của mảng

c. Code

function solution(inputString) {

    return (inputString.match(/".\*?"|-?\d+|true|false/g)||[]).length

}

**153. Tree Bottom**

a. Đề bài

Cho 1 xâu biểu diễn 1 cây nhị phân có dạng (value left\_subtree right\_subtree)

Ví dụ: với (2 (7 (2 () ()) (6 (5 () ()) (11 () ()))) (5 () (9 (4 () ()) ())))

Shape

Description automatically generated with low confidenceCây nhị phân sẽ là

Với xâu đã cho, hãy xác định các node nằm ở mức thấp nhất của cây, đưa ra theo thứ tự từ trái qua phải. Ở ví dụ trên, kết quả sẽ là [5,11,4]

b. Giải pháp

Duyệt lần lượt từ trái qua phải từng ký tự của xâu đã cho, nếu gặp ‘(‘ thì tăng mức hiện tại thêm 1, gặp ‘)’ thì giảm mức hiện tại xuống 1, nếu gặp ký tự số thì nối vào 1 string lưu số đến khi gặp khoảng trắng thì chuyển string đó sang số. Nếu mức hiện tại đang bằng với mức thấp nhất thì thêm số đó vào mảng lưu kết quả, nếu mức hiện tại lớn hơn mức thấp nhất thì cập nhật mức thấp nhất và mảng lưu kết quả là mảng mới chỉ chứa số đó.

c. Code

function solution(t) {

    var n = t.length;

    var ans;

    var lans = -1;

    var level = 0;

    var number;

    for (var i = 0; i < n; ++i) {

        var c = t[i];

        if (c === '(') {

            ++level;

            number = '';

        } else if (c === ')') {

            --level;

        } else if (c === ' ') {

            if (number && number.length > 0) {

                var parsedNumber = Number(number);

                if (lans === level) {

                    ans.push(parsedNumber);

                } else if (lans < level) {

                    lans = level;

                    ans = [parsedNumber];

                }

                number = '';

            }

        } else {

            number += c;

        }

    }

    return ans;

}

**154. Befunge-93**

a. Đề bài

Cho 1 mảng 2 chiều gồm các ký tự, gọi là chương trình Befunge-93. Chương trình Befunge-93 bao gồm các câu lệnh như sau:

* Câu lệnh điều hướng:

+ >: đi sang phải

+ <: đi sang trái

+ v: đi xuống

+ ^: đi lên

+ #: bỏ qua ô tiếp theo

* Câu lệnh điều kiện:

+ \_: pop 1 giá trị, đi sang phải nếu giá trị đó = 0 và trái nếu ngược lại

+ | : pop 1 giá trị, đi xuống nếu giá trị đó =0 và lên nếu ngược lại

* Toán tử toán học:

+ +: pop a và b rồi push a+b

+ -: pop a và b rồi push b-a

+ \*: pop a và b rồi push a\*b

+ /: pop a và b rồi push b/a (số nguyên)

+ %: pop a và b rồi push b%a

* Toán tử logic:

+ !: pop a, nếu a =0 push 1, ngược lại push 0

+ `: pop a và b, push 1 nếu b>a và 0 nếu ngược lại

* Các câu lệnh stack:

+ : : thêm vào stack giá trị giống với đỉnh stack

+ \: đổi vị trí 2 giá trị ở đỉnh stack

+ $: lấy giá trị ở đỉnh stack và hủy bỏ

* Các câu lệnh output:

+ .: Lấy giá trị và xuất dưới dạng số nguyên theo sau là khoảng trắng

+ ,: lấy giá trị và xuất dưới dạng ký tự ASCII

* Các số 0-9: push số đó vào stack
* “ : bắt đầu string mode, push tất cả các ký tự ASCII cho đến khi gặp “ tiếp theo
* \_ (khoảng trắng): không làm gì
* @: kết thúc chương trình

Nếu stack rỗng và cần đưa ra gía trị thì đưa ra 0.

Lưu ý: Chương trình chỉ chạy tối đa 1e5 lệnh hoặc đưa ra tối đa 100 ký tự

b. Giải pháp

Duyệt từ trái qua phải, từ trên xuống dưới mảng xâu đã cho, nếu gặp các ký tự thuộc lệnh điều hướng thì đi tới các ô theo lệnh đó, nếu là số thì push vào stack, nếu gặp ký tự “ thì bật biến flag và bắt đầu push các ký tự đến khi gặp “ tiếp theo. Thực hiện các lệnh của toán tử như theo đề bài yêu cầu. Nếu gặp @ hoặc đầu ra đã đủ 100 ký tự hoặc thực hiện 1e5 lệnh thì kết thúc chương trình.

c. Code

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

    const int DI[] = {0, -1, 0, 1};

    const int DJ[] = {1, 0, -1, 0};

    const int m = p.size(), n = p[0].size();

    int i = 0, j = 0, d = 0;

    std::vector<int> stack;

    std::ostringstream out;

    auto Push = [&stack](int x) { stack.push\_back(x); };

    auto Pop = [&stack]() {

        if (stack.empty()) return 0;

        int x = stack.back();

        stack.pop\_back();

        return x;

    };

    auto Top = [&stack]() { return stack.empty() ? 0 : stack.back(); };

    auto Step = [&i, &j, &d, m, n, DI, DJ]() {

        i = (i + DI[d] + m) % m;

        j = (j + DJ[d] + n) % n;

    };

    bool string\_mode = false;

    for (int cmd = 0; cmd < 100000 && out.str().size() < 100; ++cmd) {

        if (string\_mode) {

            if (p[i][j] == '"') string\_mode = false;

            else Push(p[i][j]);

        } else if (isdigit(p[i][j])) {

            Push(p[i][j] - '0');

        } else switch (p[i][j]) {

            case '>': d = 0; break;

            case '<': d = 2; break;

            case 'v': d = 3; break;

            case '^': d = 1; break;

            case '#': Step(); break;

            case '\_': d = (Pop() ? 2 : 0); break;

            case '|': d = (Pop() ? 1 : 3); break;

            case '+': { int a = Pop(), b = Pop(); Push(b + a); } break;

            case '-': { int a = Pop(), b = Pop(); Push(b - a); } break;

            case '\*': { int a = Pop(), b = Pop(); Push(b \* a); } break;

            case '/': { int a = Pop(), b = Pop(); Push(b / a); } break;

            case '%': { int a = Pop(), b = Pop(); Push(b % a); } break;

            case '!': Push(!Pop()); break;

            case '`': { int a = Pop(), b = Pop(); Push(b > a ? 1 : 0); } break;

            case ':': Push(Top()); break;

            case '\\': { int a = Pop(), b = Pop(); Push(a); Push(b); } break;

            case '$': Pop(); break;

            case '.': out << Pop() << " "; break;

            case ',': out << (char) Pop(); break;

            case '"': string\_mode = true; break;

            case '@': goto hell;

        }

        Step();

    }

    hell:

    return out.str().substr(0, 100);

}

**II-INTRO**

**1.Add**

a.Đề bài

Viết hàm tính tổng 2 số

b. Code

int solution(int param1, int param2) {

    return param1+param2;

}

**2. Century From Year**

a. Đề bài

Cho 1 năm, tính xem năm đó thuộc thế kỷ bao nhiêu

b. Code

int solution(int year) {

    if(year%100==0) return year/100;

    else return year/100+1;

}

**3. checkPalindrome**

a. Đề bài

Cho 1 xâu, kiểm tra xâu đó có phải palindrome không

b. Giải pháp

Duyệt từ phần tử đầu đến phần tử giữa của xâu, nếu mỗi phần tử i đều giống phần tử ở vị trí n-1-I (n là độ dài xâu) thì trả về true, ngược lại false

c. Code

bool solution(string inputString) {

    int size = inputString.size();

    for(int i = 0;i<size/2;i++){

        if(inputString[i]!=inputString[size-1-i]) return false;

    }

    return true;

}

**4. adjacentElementsProducts**

a. Đề bài

Cho 1 mảng số nguyên, tìm cặp 2 phần tử liên tiếp có tích lớn nhất

b. Giải pháp

Duyệt từ phần tử đầu đến phần tử gần cuối và tính tích của các cặp phần tử rồi so sánh

c. Code

int solution(vector<int> inputArray) {

    int prod=-999999;

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

    if(prod<inputArray[i]\*inputArray[i+1]) prod = inputArray[i]\*inputArray[i+1];

    }

    return prod;

}

**5. shapeArea**

a. Đề bài

Cho 1 hình n-interesting polygon, xác định diện tích của hình đó (theo đơn vị ô vuông).

Hình n-interesting polygon xác định như sau:

Chart, bar chart

Description automatically generated

b. Giải pháp

Nếu n = 1 thì trả về 1, với n = N thì diện tích của hình cạnh n = diện tích hình cạnh (n-1) + 4(n-1)

c. Code

int solution(int n) {

    if(n==1) return 1;

    else return solution(n-1)+4\*(n-1);

}

**6. Make Array Consecutive 2**

a. Đề bài

Cho 1 mảng gồm các số nguyên, cần thêm vào mảng đó bao nhiêu phần tử để thu được 1 mảng có thể sắp xếp thành dãy các số nguyên liên tiếp

b. Giải pháp

Sắp xếp mảng rồi tìm tổng khoảng cách giữa các cặp phần tử liên tiếp

c. Code

int solution(vector<int> statues) {

    sort(statues.begin(),statues.end());

    int sum = 0;

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

        sum+=statues[i+1] - statues[i] -1;

    }

    return sum;

}

**7. almostIncreasingSequence**

a. Đề bài

Cho 1 mảng các số nguyên, kiểm tra xem có thể bỏ 1 phần tử để tạo thành dãy tăng dần không (không sắp xếp lại)

b. Giải pháp

Duyệt lần lượt từ đầu đến cuối mảng, nếu phần tử i >= phần tử i+1 thì kiểm tra khi loại bỏ có tạo thành dãy tăng không, nếu có >1 lần như vậy thì trả về false, ngược lại true

c. Code

bool solution(std::vector<int> sequence) {

    int count = 0;

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

        if(sequence[i] >= sequence[i+1]){

            count++;

            // 3-6-2-4

            if(sequence[i-1]>=sequence[i+1] && i>=1 && sequence[i]>=sequence[i+2] && i+2<sequence.size()){

                count++;

            }

        }

    }

    if(count>1){

        return false;

    }else{

        return true;

    }

}

**8. maxtrix Elements Sum**

a. Đề bài

Cho 1 ma trận hình chữ nhật , tính tổng các phần tử của ma trận với điều kiện các ô bằng 0 và đứng dưới ô =0 trong cột không được tính

b. Giải pháp

Duyệt từng cột của ma trận, ở mỗi cột duyệt theo từng hàng và tính tổng các phần tử, chuyển sang cột khác khi gặp phần tử = 0 hoặc hết hàng

c. Code

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

    int total =0;

    for(int j = 0; j<matrix[0].size();j++){

        int i = 0;

        while(i<matrix.size()&&matrix[i][j]!=0){

            total+=matrix[i][j];

            i++;

        }

    }

    return total;

}

**9. All Longest Strings**

a. Đề bài

Cho 1 mảng gồm các xâu, hãy đưa ra các xâu có độ dài lớn nhất

b. Giải pháp

Tính độ dài của xâu dài nhất rồi đưa ra các xâu có cùng độ dài đó

c. Code

vector<string> solution(vector<string> inputArray) {

    vector<string> res;

    int len = 0;

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

        if(inputArray[i].length()>len){

            len = inputArray[i].length();

        }

    }

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

         if(inputArray[i].length()==len){

             res.push\_back(inputArray[i]);

         }

}

return res;

}

**10. commonCharacterCount**

a. Đề bài

Cho 2 xâu, tính xem có bao nhiêu ký tự giống nhau giữa 2 xâu

b. Giải pháp

Sử dụng 1 mảng để lưu số lượng từng loại ký tự (a, b ,c …) của xâu 1, sau đó duyệt lần lượt từng ký tự của xâu 2, nếu có ký tự đó có trong xâu 1 thì tăng biến đếm lên 1 và giảm số lượng trong mảng đi 1. Kết quả trả về là biến đếm

c. Code

int solution(string st1, string st2)

{

    int rt = 0;

    int countSt[26] = {0};

    for (char &c : st1)

    {

        countSt[c - 'a']++;

    }

    for (char &c : st2)

    {

        if (countSt[c - 'a'])

        {

            countSt[c - 'a']--;

            rt++;

        }

    }

    return rt;

}