ĐẠI HỌC BÁCH KHOA HÀ NỘI

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

----- 🙡 🕮 🙣 -----



**BÁO CÁO PROJECT I**

***Báo cáo tuần 3***

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:

**61. Creat Anagram**

a. Đề bài:

Cho 2 xâu s và t gồm các chữ cái tiếng Anh viết hoa có cùng độ dài. Tìm số chữ cái tối thiểu cần thay thế để có được “anagram” của t từ s.

(S được gọi là chuỗi anagram của t nếu khi sắp xếp lại các chữ cái trong s ta được t )

b. Giải pháp:

Duyệt lần lượt các ký tự của chuỗi t, nếu tìm thấy ký tự đó trong chuỗi s thì xóa ký tự đó trong chuỗi s, khi đó chuỗi s còn lại các ký tự cần phải thay thế

Ví dụ: xâu t: ABCDGABCD

Xâu s: BCDABGADF

Khi duyệt lần lượt các ký tự ở xâu t và tìm các ký tự đó ở xâu s ta sẽ loại bỏ được các ký tự có thể dùng ở s để sắp xếp thành t, vì độ dài của 2 xâu bằng nhau nên số với mỗi ký tự không tìm được ở s sẽ là 1 ký tự phải thay thế (trong ví dụ là C tương ứng với F) nên số ký tự còn lại (tức độ dài còn lại của s) sẽ là số ký tự cần thay thế.

c. Code:

int createAnagram(string s, string t) {

    for(int i=0;i<t.length();i++) {

        int pos = s.find(t[i]);

        if(pos>=0) s.erase(pos,1);

    }

    return s.length();

}

**62. Construct Square**

a. Đề bài:

Cho 1 xâu gồm các chữ cái tiếng Anh viết thường, tìm số chính phương lớn nhất khi thay mỗi chữ cái bằng 1 chữ số và có thể sắp xếp lại vị trí của các chữ số đó với điều kiện chữ số đầu tiên khác 0, mỗi chữ cái chỉ ánh xạ đến 1 số và ngược lại.

b. Giải pháp:

Sử dụng 2 mảng, 1 mảng để lưu số lần xuất hiện của mỗi ký tự của xâu đã cho.

Mảng thứ 2 lưu số lần xuất hiện của mỗi chữ số của các số chính phương thỏa mãn độ dài bằng độ dài của xâu đã cho.

Nếu mảng 1 và mảng 2 có cùng kích thước và chứa các phần tử giống nhau thì thỏa mãn yêu cầu đề bài

c. Code:

vector<int> getCharVector(string s){

    vector<int> buckets(26, 0);

    for(auto c : s)

        buckets[c-'a']++;

    vector<int> result;

    for(auto b : buckets){

        if(b > 0)

            result.push\_back(b);

    }

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

    return result;

}

vector<int> getIntVector(int n){

    vector<int> buckets(10, 0);

    while(n > 0){

        int temp = n % 10;

        n /= 10;

        buckets[temp]++;

    }

    vector<int> result;

    for(auto b : buckets){

        if(b > 0)

            result.push\_back(b);

    }

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

    return result;

}

int constructSquare(std::string s) {

    vector<int> list1 = getCharVector(s);

    int start = sqrt(pow(10, s.length()-1));

    int end = sqrt(pow(10, s.length()));

    for(int m=end; m>=start; m--){

        vector<int> list2 = getIntVector(m\*m);

        if(list1 == list2)

            return m\*m;

    }

    return -1;

}

**63. Numbers Grouping**

a. Đề bài:

Cho 1 mảng các số nguyên. Quy định rằng: nhóm đầu tiên bao gồm các số từ 1 đến 10^4 nhóm 2 từ 10^4+1 đến 2\*10^4, … nhóm 100 sẽ là 99\*10^4+1 đến 100\*10^4.

Các nhóm được viết lần lượt, bao gồm số của nhóm, các số trong nhóm đó, mỗi thứ được viết trên 1 dòng, tính xem cần bao nhiêu dòng để viết mảng đã cho?

b. Giải pháp:

Tính xem có bao nhiêu nhóm có trong mảng rồi cộng với số phần tử có trong mảng.

Cụ thể, sử dụng CTDL Set trong C++, tính ra số nhóm của từng phần tử trong mảng rồi thêm vào set, nếu nhóm đã xuất hiện thì sẽ ko dc thêm vào set và ngược lại.

c. Code:

int numbersGrouping(vector<int> a) {

    set<int> b;

    for(int i=0;i<a.size();i++){

         int k = a.at(i)%10000==0?a.at(i)/10000:a.at(i)/10000+1;

        b.insert(k);

    }

    return a.size()+b.size();

}

**64. Different Squares**

a. Đề bài:

Cho 1 ma trận hình chữ nhật chứa các chữ số, tính xem có bao nhiêu ma trận con khác nhau kích thước 2x2 có trong ma trận đó?

b. Giải pháp:

Duyệt tất cả các ma trận con, sử dụng CTDL Set, nếu là ma trận mới thì sẽ được thêm vào set, còn ma trận đã xuất hiện trong Set sẽ không được thêm vào. Kết quả là kích thước của set.

c. Code:

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

    set<vector<int>> res;

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

        for(int j=0;j<matrix[i].size()-1;j++){

           vector<int> a = {matrix[i][j],matrix[i][j+1],matrix[i+1][j],matrix[i+1][j+1]};

            res.insert(a);

        }

    }

return res.size();

}

**65. Most Frequent Digit Sum**

a. Đề bài:

step(x) được định nghĩa bằng x-s(x) với s(x) là tổng các chữ số của x.

Cho 1 số n, xây dựng dãy bao gồm: { n, step(n), step(step(n)),…0} sau đó thay thế các phần tử x của dãy này bằng s(x) được dãy mới. Xác định phần tử xuất hiện nhiều nhất trong dãy mới, trong trường hợp có nhiều số lần xuất hiện lớn nhất bằng nhau thì lấy phần tử có giá trị lớn nhất.

b. Giải pháp:

Sử dụng CTDL Map với key là s(x) và x là các phần tử của dãy đầu tiên, value là số lần xuất hiện của s(x). Duyệt lần lượt các phần tử trong map, kết quả sẽ là key của cặp <key,value> có value lớn nhất, trong trường hợp có nhiều value bằng nhau thì đưa ra key lớn nhất.

c. Code:

int s(int x){

    int s=0;

    while(x>0){

        s+=x%10;

        x/=10;

    }

    return s;

}

int mostFrequentDigitSum(int n) {

    map<int,int> res;

    while(n){

        res[s(n)]++;

        n=n-s(n);

    }

    pair<int,int> max;

     map<int, int>::iterator it;

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

        if((it->second>max.second)||(it->second==max.second&&it->first>max.first) )

        max = {it->first,it->second};

    }

    return max.first;

}

**66. Number of Clans**

a. Đề bài:

Gọi 2 số nguyên A và B là bạn của nhau nếu mỗi số nguyên trong mảng divisors là ước của cả A và B hoặc không là ước của cả A và B. Nếu 2 số A và B là bạn, chúng được gọi là cùng 1 clan, hỏi có bao nhiêu clan trong các số nguyên từ 1 tới k?

b. Giải pháp:

Duyệt lần lượt các cặp số từ 1 đến k để đếm số cặp là friends, khi đó, số clan sẽ là k-số cặp.

c. Code:

bool isFriends(vector<int> divisors, int a, int b){

    while(!divisors.empty()){

        int i=divisors.back();

        if((a%i==0&&b%i!=0)||(a%i!=0&&b%i==0)) return false;

        divisors.pop\_back();

    }

    return true;

}

int numberOfClans(vector<int> divisors, int k) {

int friendclans=0;

for(int i=1;i<k;i++){

    for(int j=i+1;j<=k;j++){

        if(isFriends(divisors, i, j)) { friendclans++;

        break;}

    }

}

return k-friendclans;

}

**67. House Numbers Sum**

a. Đề bài:

Cho 1 mảng gồm các số nguyên dương, tính tổng các số từ phần tử đầu tiên đến phần tử bằng 0 đầu tiên.

b. Giải pháp:

Duyệt lần lượt mảng để tính tổng đến khi gặp phần tử bằng 0 thì dừng lại

c. Code:

int houseNumbersSum(vector<int> inputArray) {

    int s=0;

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

        if(inputArray.at(i)==0) break;

        else s+=inputArray.at(i);

    }

    return s;

}

**68. All longest Strings**

a. Đề bài:

Cho 1 mảng các string, trả về mảng con chứa các string dài nhất

b. Giải pháp:

Duyệt lần lượt các phần tử trong mảng đã cho, nếu gặp chuỗi dài hơn thì mảng 2 chứa chuỗi đó, nếu gặp chuỗi dài bằng chuỗi của mảng 2 thì thêm vào mảng 2.

c. Code:

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

    vector<string> ans = {inputArray.at(0)};

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

        if(inputArray.at(i).size()>ans.back().size()) ans = {inputArray.at(i)};

        else if(inputArray.at(i).size()==ans.back().size())

        ans.push\_back(inputArray.at(i));

    }

    return ans;

}

**69. House of Cats**

a. Đề bài:

Cho 1 số nguyên dương là tổng số chân của mèo và người, đưa ra mảng các số người có thể có theo thứ tự tăng dần.

b. Giải pháp:

Nếu tổng số chân chia hết cho 4 thì số người có thể bắt đầu từ 0, ngược lại ko chia hết cho 4 thì số người chỉ có thể bắt đầu từ 1, sau đó cứ tăng 2 đv rồi push vào mảng đến khi tổng số chân người vượt qua tổng chân cả người và mèo thì ngừng.

c. Code:

vector<int> houseOfCats(int legs) {

vector<int> ans;

if(legs%4!=0) ans.push\_back(1);

else ans.push\_back(0);

int next = ans.back()+2;

while(next\*2<=legs){

    ans.push\_back(next);

    next = ans.back()+2;

}

return ans;

}

**70. Alphabet Subsequence**

a. Đề bài:

Kiểm tra xem các kí tự trong dãy đã cho có được sắp xếp theo thứ tự alphabet và không có ký tự giống nhau không

b. Giải pháp:

Duyệt lần lượt các ký tự trong chuỗi, nếu ký tự i đứng sau hoặc giống ký tự i+1thì trả về false, nếu không có ký tự nào như thế thì trả về true

c. Code:

bool alphabetSubsequence(string s) {

    for(int i=0;i<s.length()-1;i++){

        if(s[i]>=s[i+1]) return false;

    }

    return true;

}

**71. Minimal Number of Coins**

a. Đề bài:

Cho 1 mảng bao gồm các số là các mệnh giá tiền theo thứ tự tăng dần và 1 số nguyên price là số tiền cần trả. Giả sử bạn có vô hạn tiền, tính số tờ tiền phải dùng để trả ít nhất.

b. Giải pháp:

Sử dụng tờ có mệnh giá lớn nhất đến khi số tiền phải trả còn lại nhỏ hơn mệnh giá đó, lặp lại đến khi số tiền phải trả còn lại bằng 0.

c. Code:

int minimalNumberOfCoins(vector<int> coins, int price) {

   int ans =0;

   while(price>0){

    ans += price/coins.back();

    price = price%coins.back();

    coins.pop\_back();

}

return ans;

}

**72. Add border**

a. Đề bài:

Cho 1 ma trận hình chữ nhật kích thước nxm, thêm đường bao quanh ma trận đó bằng các dấu \*

Ví dụ:

picture = ["abc",

"ded"]

addBorder(picture) = ["\*\*\*\*\*",

"\*abc\*",

"\*ded\*",

"\*\*\*\*\*"]

b. Giải pháp:

Nối thêm các ký tự “\*” vào đầu và cuối mỗi phần tử trong mảng và thêm 2 chuỗi “\*\*\*…\*\*\*” có kích thước m+2 vào đầu và cuối mảng.

c. Code:

vector<string> addBorder(vector<string> picture) {

    int row = picture.size();

    int col = picture.at(0).size();

    vector<string> ans;

    string add ="\*";

    for(int i=1;i<col+2;i++){

        add.append("\*");

    }

    ans.push\_back(add);

    for(int i=0;i<row;i++){

        ans.push\_back("\*"+picture.at(i)+"\*");

    }

    ans.push\_back(add);

    return ans;

}

**73. Switch Lights**

a. Đề bài:

Có N ngọn nến được đặt theo cùng 1 hàng, 1 vài ngọn đã được thắp sáng, các ngọn nến từ 1 đến n có quy luật sau: nếu 1 ngọn nến đang sáng tắt sẽ làm đảo lộn trạng thái của tất cả các ngọn nến đứng trước nó. Hỏi trạng thái các ngọn nến sẽ như thế nào sau khi áp dụng quy luật trên với 1 mảng trạng thái cho trước?

b. Giải pháp:

Tìm xem ở ngọn nến thứ i có bao nhiêu ngọn nến đứng sau phát sáng, nếu đó là số chẵn thì thiết lập trạng thái của ngọn nến hiện tại là tắt, nếu là số lẻ thì trạng thái là sáng.

c. Code:

vector<int> switchLights(vector<int> a) {

    int light=0;

    for(int i=0;i<a.size();i++){

        if(a[i]==1) light++;

    }

    for(int i=0;i<a.size();i++){

        if(a.at(i)==1) light--;

        a.at(i)= (light%2==0)?0:1;

    }

    return a;

**}**

**74. Time Reading**

a. Đề bài

Cho 1 xâu là 1 đoạn văn tiếng Anh, hỏi có bao nhiêu từ không vượt quá số maxLength cho trước?

b. Giải pháp

Duyệt lần lượt các kí tự trong xâu, nếu ký tự ở vị trí i là chữ cái và i-1 không phải thì đó là chữ cái đầu tiên của 1 từ, nếu ký tự ở vị trí i là chữ cái và i+1 không phải chữ cái thì đó là chữ cái cuối cùng của từ đó. Sau đó tính độ dài của từ đó và kiểm tra điều kiện có vượt quá maxLength không, nếu ko vượt qua thì tăng biến đếm thêm 1.

c. Code

bool isAlpha1(char a){

    if(a>='a'&&a<='z') return true;

     if(a>='A'&&a<='Z') return true;

     return false;

}

int timedReading(int maxLength, string text) {

    int count=0;

    int first=0;

 for(int i=0;i<text.length();i++){

     if(isAlpha1(text[i])&&!isAlpha1(text[i-1])) first=i;

     if(isAlpha1(text[i])&&!isAlpha1(text[i+1]))

        if(i-first+1<=maxLength) count++;

 }

 return count;

}

**75. Elections Winners**

a. Đề bài

Cho 1 mảng các số nguyên dương là số phiếu bầu hiện tại của các ứng cử viên, còn k cử tri chưa bỏ phiếu. Hỏi có bao nhiêu ứng cử viên vẫn còn cơ hội chiến thắng (UCV được coi là chiến thắng nếu có số phiếu nhiều nhất, hơn cả người xếp thứ 2).

b. Giải pháp

Tìm max của mảng, tìm xem có bao nhiêu phần tử + k > max (count). Nếu count =0 và có nhiều phần tử có cùng max thì trả về 0, nếu count=0 và chỉ có 1 max thì trả về 1, ngược lại trả về count.

c. Code

int electionsWinners(vector<int> votes, int k) {

    int count =0;

    int max =0;

    int max2;

    for(int i=0;i<votes.size();i++){

        if(votes.at(i)>max) {

            max = votes.at(i);

        }

        else if(votes.at(i)==max)

        max2=max;

    }

     for(int i=0;i<votes.size();i++){

        if(votes.at(i)+k>max)

            count++;

}

if(count == 0&&max==max2) return 0;

if(count == 0&&max!=max2) return 1;

return count;

}

**76. Integer to String of Fixed Width**

a. Đề bài

Cho 1 số nguyên dương và 1 số width, hãy đưa ra 1 string là số có độ dài width tính từ hàng đơn vị của số đã cho

b. Giải pháp

Duyệt lần lượt từ hàng đơn vị của số đã cho để lấy các chữ số rồi thêm vào xâu kết quả, đến khi đã lấy đủ width số thì ngừng.

C. Code

string integerToStringOfFixedWidth(int number, int width) {

    string ans;

    while(width>0){

        ans=(char)((number%10)+'0')+ans;

        number/=10;

        width--;

    }

    return ans;

}

**77. Are Similar?**

a. Đề bài

2 mảng được gọi là similar nếu 1 mảng chứa 1 cặp số mà nếu đổi chỗ cặp số đó ta được mảng kia. Kiểm tra xem 2 mảng đã cho có similar không?

b. Giải pháp

Đếm số vị trí mà phần tử 2 mảng khác nhau. Nếu = 0 tức 2 mảng giống nhau, trả về true, nếu =2, sắp xếp lại 2 mảng rồi so sánh 2 mảng, nếu bằng nhau trả về true. Các trường hợp còn lại trả về false.

c. Code

bool areSimilar(vector<int> a, vector<int> b) {

    int count=0;

    for(int i=0;i<a.size();i++){

        if(a[i]!=b[i]) count++;

    }

    if(count==0) return true;

    if(count==2) {

    sort(a.begin(),a.end());

    sort(b.begin(),b.end());

    if(a==b) return true;

}

return false;

}

**79. Three Split**

a. Đề bài

Cho 1 mảng các số nguyên, tính xem có bao nhiêu cách chia mảng thành 3 mảng con sao cho tổng mỗi mảng bằng nhau

b. Giải pháp

Duyệt lần lượt các phần tử để tính tổng của các mảng con, nếu mỗi lần duyệt qua phần tử kế tiếp mà tổng của mảng con vẫn vậy thì tăng biến đếm thêm 1.

c. Code

function threeSplit(a) {

  var totalSum = a.reduce((total, cur) => total + cur);

  var firstSum = 0;

  var secondSum = 0;

  var ways = 0;

  for (var i = 0; i < a.length - 2; i++) {

    firstSum += a[i];

    secondSum = 0;

    for (var j = i + 1; j < a.length - 1; j++) {

      secondSum += a[j];

      if (firstSum === secondSum && secondSum === totalSum - firstSum - secondSum) {

        ways++;

      }

    }

  }

  return ways;

}

**80. Character Parity**

a. Đề bài

Cho 1 ký tự, kiểm tra ký tự đó có phải là số không? Nếu là số thì là số chẵn hay lẻ?

b. Giải pháp

Nếu ký tự không phải là số thì đưa ra màn hình “not a digit”, ngược lại, nếu chia hết cho 2 thì đưa ra “even”, nếu ko chia hết cho 2 thì đưa ra “odd”.

c. Code

string characterParity(char symbol) {

    if(!isdigit(symbol)) return "not a digit";

    else if((symbol-'0')%2==0) return "even";

    else return "odd";

}

**81. Reflect String**

a. Đề bài

Cho 1 ánh xạ: a – z , b – y , c- x….. y – b , z – a. Hãy biến đổi 1 xâu theo ánh xạ đó.

b. Giải pháp

Gọi sum là tổng của ‘a’ và ‘z’ (cũng là tổng của mỗi cặp ánh xạ), khi đó với mỗi ánh xạ i’ = sum -i

c. Code

string reflectString(string inputString) {

    char sum = 'a'+'z';

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

        inputString[i]=(char) (sum-inputString[i]);

    }

    return inputString;

}