ĐẠI HỌC BÁCH KHOA HÀ NỘI

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

----- 🙡 🕮 🙣 -----



**BÁO CÁO PROJECT I**

***Báo cáo tuần 5***

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 101 đến câu 125

**101. Swap Diagonals**

a. Đề bài

Cho ma trận vuông nxn (n>=1), đổi vị trí các phần tử nằm trên 2 đường chéo chính của ma trận

b. Giải pháp

Duyệt tất cả các hàng của ma trận (từ 0 đến n-1) , ở hàng thứ i, đổi chỗ 2 phần tử [i][i] và [i][n-1-i]

c. Code

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

    int n = matrix[0].size();

    if(n==1) return matrix;

    else {

        for(int i=0;i<n;i++){

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

        }

        return matrix;

    }

}

**102. Crossing Sum**

a. Đề bài

Cho 1 ma trận hình chữ nhật, tính tổng tất cả các phần tử nằm trên hàng thứ a và cột thứ b

b. Giải pháp

Tính tổng các phần tử của hàng a cộng với tổng các phần tử cột b rồi trừ đi phần tử nằm trên điểm giao

c. Code

int solution(vector<vector<int>> matrix, int a, int b) {

    int sum=0;

    int col = matrix[0].size();

    int row = matrix.size();

    for(int i=0;i<col;i++){

        sum+=matrix[a][i];

    }

    for(int i=0;i<row;i++){

        sum+=matrix[i][b];

    }

    sum-=matrix[a][b];

    return sum;

}

**103. Draw Rectangle**

a. Đề bài

Cho 1 ma trận hình chữ nhật, vẽ 1 hình chữ nhật con có tọa độ 2 đỉnh nằm trên đường chéo thứ 1 là (x1,y1) và (x2,y2) theo cách sau:

VD: canvas = [['a', 'a', 'a', 'a', 'a', 'a', 'a', 'a'],

['a', 'a', 'a', 'a', 'a', 'a', 'a', 'a'],

['a', 'a', 'a', 'a', 'a', 'a', 'a', 'a'],

['b', 'b', 'b', 'b', 'b', 'b', 'b', 'b'],

['b', 'b', 'b', 'b', 'b', 'b', 'b', 'b']]

rectangle = [1, 1, 4, 3]

[['a', 'a', 'a', 'a', 'a', 'a', 'a', 'a'],

['a', **'\*'**, **'-'**, **'-'**, **'\*'**, 'a', 'a', 'a'],

['a', **'|'**, 'a', 'a', **'|'**, 'a', 'a', 'a'],

['b', **'\*'**, **'-'**, **'-'**, **'\*'**, 'b', 'b', 'b'],

['b', 'b', 'b', 'b', 'b', 'b', 'b', 'b']]

b. Giải pháp

Duyệt lần lượt các phần tử nằm trên 2 cột và 2 hàng rồi thay đổi thành ký tự phù hợp

c. Code

vector<vector<char>> solution(vector<vector<char>> canvas, vector<int> rectangle) {

    int y1 = rectangle[0], x1=rectangle[1], y2= rectangle[2],x2= rectangle[3];

    canvas[x1][y1]='\*';

    canvas[x1][y2]='\*';

    canvas[x2][y1]='\*';

    canvas[x2][y2]='\*';

    for(int i=y1+1;i<y2;i++) canvas[x1][i] ='-';

    for(int i=y1+1;i<y2;i++) canvas[x2][i] ='-';

    for(int i=x1+1;i<x2;i++) canvas[i][y1] ='|';

    for(int i=x1+1;i<x2;i++) canvas[i][y2] ='|';

    return canvas;

}

**104. Volleyball Positions**

a. Đề bài

Cho 1 ma trận 4x3 có các vị trí 1, 2,3,4,5,6:

0 3 0

4 0 2

0 6 0

5 0 1

Và cho thuật toán như sau: phần tử tại vị trí 1 được chuyển tới vị trí 6, vị trí 2 chuyển tới 1, 3 tới 2, 4 tới 3, 5 tới 4 và 6 tới 5.

Cho ma trận là kết quả của k lần áp dụng thuật toán trên, tìm ma trận ban đầu?

b. Giải pháp

Ta chỉ xét số dư của k chia 6 vì với k= bội của 6 thì các phần tử được hoán đổi 1 vòng và quay trở về vị trí ban đầu. Với mỗi đơn vị của k, ta áp dụng ngược lại thuật toán trên và sau k lần ta được ma trận ban đầu.

c. Code

vector<vector<string>> solution(vector<vector<string>> formation, int k) {

    k=k%6;

    for(int i=1;i<=k;i++){

         formation[3][2].swap(formation[2][1]);

         formation[3][0].swap(formation[2][1]);

         formation[1][0].swap(formation[3][0]);

         formation[1][0].swap(formation[0][1]);

         formation[1][2].swap(formation[0][1]);

    }

    return formation;

}

**105. Star Rotation**

a. Đề bài

1 hình sao của 1 ma trận được xác định bằng 2 đường chéo chính, cột giữa, hàng giữa của 1 ma trận. Cho 1 ma trận (2k+1)x(2k+1), tọa độ của tâm hình sao và chiều rộng của hình sao nằm trong ma trận đó và 1 số nguyên k, hãy đưa ra ma trận mới khi xoay hình sao đó 1 góc 45k (độ) sang phải.

b. Giải pháp

Ta sẽ thay đổi vị trí các phần tử nằm trên 2 đường chéo chính, cột giữa, hàng giữa của ma trận chứa hình sao cho nhau, các phần tử còn lại giữ nguyên.

Ví dụ

matrix = [[**1**, 0, 0, **2**, 0, 0, **3**],

[0, **1**, 0, **2**, 0, **3**, 0],

[0, 0, **1**, **2**, **3**, 0, 0],

[**8**, **8**, **8**, **9**, **4**, **4**, **4**],

[0, 0, **7**, **6**, **5**, 0, 0],

[0, **7**, 0, **6**, 0, **5**, 0],

[**7**, 0, 0, **6**, 0, 0, **5**]]

Với mỗi lần xoay 45 độ, các phần tử 1 sẽ chuyển đến vị trí của phần tử 2, 2->3, 3->4,…,8->1. Ta chỉ xét 45k%360 lần xoay vì khi xoay 360 độ mảng không đổi.

c. Code

vector<vector<int>> solution(vector<vector<int>> matrix, int width, vector<int> center, int t) {

    int x=center[0], y= center[1];

     t=t%8;

     for(int i=1;i<=t;i++){

         for(int j=1;j<=width/2;j++){

             int temp = matrix[x-j][y-j];

             matrix[x-j][y-j] = matrix[x][y-j];

             matrix[x][y-j] = matrix[x+j][y-j];

             matrix[x+j][y-j] = matrix[x+j][y];

             matrix[x+j][y]=matrix[x+j][y+j];

             matrix[x+j][y+j]=matrix[x][y+j];

             matrix[x][y+j]=matrix[x-j][y+j];

             matrix[x-j][y+j]=matrix[x-j][y];

             matrix[x-j][y]=temp;

         }

     }

     return matrix;

}

**106. Sudoku**

a. Đề bài

Cho 1 ma trận 9x9, kiểm tra xem ma trận có phải sudoku không? (mỗi hàng, mỗi cột, mỗi ma trận con 3x3(không đè lên nhau) chỉ chứa các số 1 -> 9, các số không lặp lại)

b. Giải pháp

Duyệt qua từng hàng, cột và ma trận 3x3 để kiểm tra có phải sudoku không

c. Code

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

    for (int i = 0; i < 9; i++) {

        int a = 0, b = 0, c = 0;

        for (int j = 0; j < 9; j++) {

            a ^= 1 << grid[i][j];

            b ^= 1 << grid[j][i];

            c ^= 1 << grid[i - i % 3 + j / 3][i % 3 \* 3 + j % 3];

        }

        if (a != 1022 || b != 1022 || c != 1022)

            return false;

    }

    return true;

}

**107. Minesweeper**

a. Đề bài

Cho 1 ma trận kiểu bool thể hiện các ô có mìn (true) và không có mìn (false) của game Minesweeper. Hãy tìm 1 mảng với mỗi phần tử thể hiện số mìn của các ô xung quanh nó.

b. Giải pháp

Tạo 1 ma trận kích thước bằng ma trận đã cho và các phần tử bằng 0. Duyệt ma trận đã cho, nếu có mìn thì tăng các ô xung quanh lên 1.

c. Code

void check(vector<vector<int>> &res ,int i, int j, int row, int col){

    if(i>=0&&i<=(row-1)&&j>=0&&j<=(col-1)) res[i][j]++;

}

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

    vector<vector<int>> res;

    int row = matrix.size();

    int col = matrix[0].size();

    vector<int> sub;

    for(int i=0;i<col;i++){

        sub.push\_back(0);

    }

    for(int i=0;i<row;i++){

        res.push\_back(sub);

    }

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

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

            if(matrix[i][j]) {

                check(res,i-1,j-1,row,col);

                check(res,i-1,j,row,col);

                check(res,i-1,j+1,row,col);

                check(res,i,j-1,row,col);

                check(res,i,j+1,row,col);

                check(res,i+1,j-1,row,col);

                check(res,i+1,j+1,row,col);

                check(res,i+1,j,row,col);

            }

        }

    }

    return res;

}

**108. Box Bur**

a. Đề bài

Cho 1 ma trận kích thước mxn>= 3x3. Hãy tìm 1 ma trận con (m-1)x(n-1) với mỗi phần tử là trung bình cộng của các phần tử trong ma trận con 3x3 của ma trận đã cho ở các vị trí tương ứng.

b. Giải pháp

Với mỗi vị trí của ma trận kết quả sẽ bằng trung bình cộng của các phần tử trong ma trận 3x3 ở vị trí tương ứng. Sau đó thêm vào mảng kết quả.

c. Code

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

    vector<vector<int>> ans;

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

        vector<int> sub;

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

            int element = (image[i-1][j-1]+image[i-1][j]+image[i-1][j+1]+image[i][j-1]+image[i][j]+image[i][j+1]+image[i+1][j-1]+image[i+1][j]+image[i+1][j+1])/9;

            sub.push\_back(element);

        }

        ans.push\_back(sub);

    }

    return ans;

}

**109. Contours Shifting**

a. Đề bài

Cho 1 ma trận hình chữ nhật, đường bao 0 chứa tất cả các phần tử nằm ngoài cùng của ma trận, đường bao 1 chứa các phần tử nằm ngoài cùng thứ 2, … Hãy áp dụng thuật toán sau vào ma trận:

* Nếu đường bao là số chẵn thì dịch tất cả các phần tử của đường bao đó theo chiều kim đồng hồ
* Nếu đường bao là số lẻ thì dịch tất cả các phần tử của đường bao đó theo chiều ngược chiều kim đồng hồ

b. Giải pháp

Duyệt từng đường bao của ma trận và dịch chuyển các phần tử

c. Code

void changeReverse(vector<vector<int>> &matrix, int firstRow, int lastRow, int firstCol, int lastCol){

    if(firstCol==lastCol||firstRow==lastRow){

        for(int i=firstCol;i<lastCol;i++){

            swap(matrix[firstRow][i],matrix[firstRow][i+1]);

        }

        for(int i=firstRow;i<lastRow;i++){

            swap(matrix[i][firstCol],matrix[i+1][firstCol]);

        }

    }else {

        for (int i = firstCol; i < lastCol; i++) {

            swap(matrix[firstRow][i], matrix[firstRow][i + 1]);

        }

        for (int i = firstRow; i < lastRow; i++) {

            swap(matrix[i][lastCol], matrix[i + 1][lastCol]);

        }

        for (int i = lastCol; i > firstCol; i--) {

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

        }

        for (int i = lastRow; i > firstRow + 1; i--) {

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

        }

    }

}

void change(vector<vector<int>> &matrix, int firstRow, int lastRow, int firstCol, int lastCol){

    if(firstCol==lastCol||firstRow==lastRow){

        for(int i=lastRow;i>firstRow;i--){

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

        }

        for(int i=lastCol;i>firstCol;i--){

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

        }

    }else{

    for(int i=firstRow;i<lastRow;i++){

        swap(matrix[i][firstCol],matrix[i+1][firstCol]);

    }

    for(int i=firstCol;i<lastCol;i++){

        swap(matrix[lastRow][i],matrix[lastRow][i+1]);

    }

    if(firstCol==lastCol||firstRow==lastRow);else{

        for(int i=lastRow;i>firstRow;i--){

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

        }

        for(int i=lastCol;i>firstCol+1;i--){

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

        }

    }

    }

}

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

    int row = matrix.size();

    int col = matrix[0].size();

    int i =0;

    int end = ceil((float)min(row,col)/2);

    int firstRow =0,firstCol=0;

    int lastRow=row-1, lastCol=col-1;

    while(i<end){

        if(i%2==0)   change(matrix, firstRow+i, lastRow-i, firstCol+i, lastCol-i);

        else changeReverse(matrix, firstRow+i, lastRow-i, firstCol+i, lastCol-i);

        i++;

    }

    return matrix;

}

**110. Polygon Perimeter**

a. Đề bài

Cho 1 mảng kiểu bool, mỗi phần tử là 1 ô vuông và được tô đen nếu phần tử đó là true. Tính chu vi của hình được tô đen

b. Giải pháp

Đếm số lượng phần tử được tô đen (s) và số lượng phần tử được tô đen gần phần tử đó và có cạnh chung(m). Chu vi của hình được tô đen sẽ bằng 4xs-m.

c. Code

bool check2(vector<vector<bool>> &matrix, int i, int j){

    if(i<0||i>=matrix.size()||j<0||j>=matrix[0].size()) ;

    else if(matrix[i][j]) return true;

    return false;

}

int check(vector<vector<bool>> &matrix, int i, int j){

    int k=0;

    if(check2(matrix,i-1,j)) k++;

    if(check2(matrix,i,j-1)) k++;

    if(check2(matrix,i,j+1)) k++;

    if(check2(matrix,i+1,j)) k++;

    return k;

}

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

    int s=0,m=0;

    for(int i=0;i<matrix.size();i++){

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

            if(matrix[i][j]) {

                ++s;

                m+=check(matrix,i,j);

            }

        }

    }

    return 4\*s-m;

}

**111. Gravitation**

a. Đề bài

Cho 1 mảng gồm các xâu có độ dài bằng nhau gồm các ký tự ‘#’ và ‘.’, ‘#’ biểu thị cho 1 viên đá. Thả các viên đá tử đỉnh, các viên có vận tốc bằng nhau và không đổi, 1 viên gọi là bất động nếu nó chạm đáy hoặc bị chặn lại bởi 1 viên bất động khác, tìm xem những cột nào bất động trước.

Ví dụ:

rows = ["#..##",

".##.#",

".#.##",

"....."]

Graphical user interface

Description automatically generated with medium confidence

b. Giải pháp

Duyệt qua tất cả các cột của ma trận, tính toán thời gian mỗi cột trở lên bất động và xem những cột nào có thời gian bất động nhỏ nhất.

c. Code

 vector<int> solution(vector<string> rows) {

    vector<int> res;

    int min =999999;

    res.push\_back(999999);

    for(int i=0;i<rows[0].size();i++){

        int s=0;

        bool check = false;

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

            if(rows[j][i]=='#') check = true;

            if(check&&rows[j][i]=='.') s++;

        }

        if(s<min) {

            min=s;

            while(!res.empty()) res.pop\_back();

            res.push\_back(i);

        }else

        if(s==min) res.push\_back(i);

    }

  return res;

}

**112. Is information consistent?**

a. Đề bài

Cho 1 ma trận, mỗi phần tử tại 1 hàng là lời khai của từng nhân chứng khi được hỏi A có tội không, với ‘-1’ là không có tội, ‘0’ là không biết, ‘1’ là có tội. Các hàng là lời khai của những người đó ở các thời điểm khác. Kiểm tra xem những lời khai có bị mâu thuẫn không? (mâu thuẫn khi 1 người vừa trả lời là có tội và không có tội)

b. Giải pháp

Duyệt lời khai của từng người, nếu xuất hiện cả 1 và -1 thì là mâu thuẫn, ngược lại thì không

c. Code

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

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

        bool guilty = false;

        bool innocent = false;

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

            if(evidences[i][j]==-1) innocent=true;

            if(evidences[i][j]==1) guilty=true;

        }

        if(guilty&&innocent) return false;

    }

    return true;

}

**114. Shuffled Array**

a. Đề bài

Cho một mảng có N+1 phần tử bao gồm N phần tử và tổng của N phần tử đó được đặt ở vị trí bất kỳ. Hãy trả về mảng N phần tử không chứa tổng được sắp xếp theo thứ tự tăng dần

b. Giải pháp

Tìm ra phần tử tổng của N phần tử ban đầu rồi xóa phần tử đó ra khỏi mảng. Sau đó sắp xếp lại mảng.

c. Code

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

   int sum =0;

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

       sum+=shuffled[i];

   }

   sum/=2;

    shuffled.erase(find(shuffled.begin(),shuffled.end(),sum));

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

    return shuffled;

}

**115. Sort by Height**

a. Đề bài

Cho 1 mảng chứa 1 số phần tử bằng -1 và các phần tử khác (>0). Sắp xếp mảng theo thứ tự tăng dần và không thay đổi vị trí các phần tử -1.

b. Giải pháp

Sử dụng thuật toán sắp chọn với các phần tử khác -1.

c. Code

vector<int> solution(vector<int> a) {

    for(int i=0;i<a.size();i++){

        if(a[i]!=-1){

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

                if(a[j]!=-1){

                    if(a[j]<a[i]) swap(a[i],a[j]);

                }

            }

        }

    }

    return a;

}

**116. Sort by Length**

a. Đề bài

Cho 1 mảng gồm các xâu, sắp xếp mảng theo thứ tự độ dài tăng dần, nếu 2 xâu có độ dài bằng nhau thì vị trí tương sau khi sắp xếp phải giống như lúc trước sắp xếp

b. Giải pháp

Sử dụng thuật toán stable sort

c. Code

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

    stable\_sort(a.begin(), a.end(), [](string i, string j) -> bool { return i.size() < j.size(); });

    return a;

}

**117. Boxes Packing**

a. Đề bài

Cho 3 mảng gồm các phần tử là kích thước tương ứng (chiều dài, chiều rộng, chiều cộng) của mỗi hộp. Kiểm tra xem có thể nhét các hộp đó vào nhau không, sao cho mỗi hộp i chỉ chứa nhiều nhất 1 hộp con i-1 ( tức là hộp con này nằm trực tiếp trong hộp i, không nằm trong hộp con nhỏ hơn i) ? 1 hộp có thể nằm trong 1 hộp khác nếu cả 3 kích thước của hộp đó nhỏ hơn 3 kích thước của hộp kia, hộp có thể xoay.

b. Giải pháp

Chuẩn hóa lại kích thước bằng cách sắp xếp lại các kích thước của mỗi hộp theo thứ tự giảm dần. Sau đó sắp xếp lại thứ tự các hộp theo thể tích tăng dần.

Duyệt lần lượt từng hộp, nếu kích thước tương ứng của hộp sau nhỏ hơn hộp trước thì trả về false. Nếu duyệt hết các hộp thì trả về true.

c. Code

bool solution(vector<int> length,vector<int> width,vector<int> height) {

    vector<vector<int>> vol;

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

    {

        vector<int> a({length[i],width[i],height[i]});

        sort(a.begin(),a.end());

        vol.push\_back(a);

    }

    sort(vol.begin(),

         vol.end(),[](const auto& vol1, const auto& vol2){

        return (vol1[0] < vol2[0])&&(vol1[1] < vol2[1])&&(vol1[2] < vol2[2]);

        });

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

        for (int j = 0; j < 3; j++){

            if (vol[i][j] <= vol[i-1][j])

                return false;

        }

    }

    return true;

}

**118. Maxium Sum**

a. Đề bài

Cho 1 mảng a gồm các số nguyên, và 1 mảng q kích thước nx2 gồm các cặp [l,r] (l,r là số nguyên, l<=r) chỉ mảng con của a từ vị trí l đến r. Hãy sắp xếp lại mảng a sao cho tổng các phần tử trong các mảng con của a được quy định trong q là lớn nhất và trả về giá trị đó.

b. Giải pháp

Sắp xếp mảng đã cho theo thứ tự tăng dần

Tính xem trong các đoạn [l,r] thì tần suất xuất hiện của những vị trí là bao nhiêu, sau đó sắp xếp theo thứ tự tăng dần, khi đó tổng lớn nhất sẽ là tích của tần số lớn nhất x số lớn nhất trong mảng đã cho + tần số lớn thứ 2 x số lớn thứ 2 trong mảng đã cho +…

c. Code

int solution(vector<int> a, vector<vector<int>> q) {

   vector<int> res;

   int sum=0;

    for(int i=0;i<a.size();i++){

        int count=0;

    for(int j=0;j<q.size();j++){

        if(i>=q[j][0]&&i<=q[j][1]) count++;

    }

    res.push\_back(count);

    }

    sort(a.begin(),a.end());

    sort(res.begin(),res.end());

    int i\_res = res.size()-1;

    int i\_a = a.size()-1;

    while(i\_res>=0){

        sum+=res[i\_res]\*a[i\_a];

        i\_res--;

        i\_a--;

    }

    return sum;

}

**119. Rows Rearranging**

a. Đề bài

Cho 1 ma trận hình chữ nhật, kiểm tra xem có thể thay đổi vị trí các hàng của ma trận để tất cả các cột được sắp xếp theo thứ tự tăng dần không

b. Giải pháp

Sắp xếp ma trận theo các hàng (nếu phần tử thứ 1 bằng nhau thì kiểm tra phần tử thứ 2, …)

Sau đó duyệt lại tất cả các cột và kiểm tra các phần tử có được sắp xếp theo thứ tự tăng dần không

c. Code

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

    for(int i=0;i<matrix.size();i++){

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

        if(matrix[i][0]>matrix[j][0]) swap(matrix[i],matrix[j]);

        else if(matrix[i][0]==matrix[j][0]) {

            if(matrix[i][1]>matrix[j][1]) swap(matrix[i],matrix[j]);

            else if(matrix[i][1]==matrix[j][1])

            if(matrix[i][2]>matrix[j][2]) swap(matrix[i],matrix[j]);

        }

    }

    }

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

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

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

        }

    }

    return true;

}

**120. Digit Difference Sort**

a. Đề bài

Cho 1 mảng các số nguyên, định nghĩa difference là hiệu giữa chữ số lớn nhất và chữ số nhỏ nhất của 1 số, sắp xếp mảng đã cho theo thứ tự tăng dần difference, nếu 2 difference bằng nhau thì thứ tự phải giống với vị trí tương đối lúc chưa sắp xếp

b. Giải pháp

Sử dụng hàm sắp xếp stable\_sort để so sánh các phần tử theo difference và giữ nguyên vị trí tương đối của chúng nếu có cặp difference bằng nhau.

c. Code

int difference(int x){

    int max =x%10;

    int min =x%10;

    x/=10;

    while(x>0){

        int a = x%10;

        if(max<a) max = a;

        if(min>a) min = a;

        x/=10;

    }

    return max-min;

}

vector<int> solution(vector<int> a) {

    stable\_sort(a.begin(),a.end(),[](int x1,int x2){

        return difference(x1)<=difference(x2);

    });

    return a;

}

**121. Unique Digit Products**

a. Đề bài

Cho 1 mảng các số nguyên, định nghĩa product(x) là tích tất cả các chữ số của số nguyên x. Hỏi có bao nhiêu product khác nhau của từng số nguyên trong mảng đã cho.

b. Giải pháp

Sử dụng cấu trúc dữ liệu Set, thêm từng giá trị product của mỗi phần tử trong mảng vào set. Trả về kích thước của Set

c. Code

int product(int x){

    int res =1;

    while(x>0){

        int a = x%10;

        res\*=a;

        x/=10;

    }

    return res;

}

int solution(vector<int> a) {

    set<int> res;

    for(int i=0;i<a.size();i++){

        res.insert(product(a[i]));

    }

    return res.size();

}

**122. Bishop and Pawn**

a. Đề bài

Cho vị trí của 1 con tượng và 1 con tốt trên bàn cờ vua. Kiểm tra xem con tượng có thể ăn được con tốt không

b. Giải pháp

Để con tượng có thể ăn được con tốt thì con tốt phải nằm trên đường chéo của con tượng, khi đó con tốt và con tượng là 2 đỉnh nằm trên 1 đường chéo của 1 hình vuông, tức khoảng cách hoành độ của 2 quân bằng khoảng cách tung độ của 2 quân.

c. Code

bool solution(string bishop, string pawn) {

    if(abs( (int)(pawn[0]-bishop[0]) ) == abs( (int) (pawn[1]-bishop[1]) ) ) return true;

    return false;

}

**123. Chess Knight Moves**

a. Đề bài

Cho vị trí của 1 con mã trên bàn cờ vua, tính xem có bao nhiêu vị trí mà con mã có thể đi trong nước tiếp theo.

b. Giải pháp

Kiểm tra từng vị trí con mã có thể đi, nếu vị trí thỏa mãn nằm trong bàn cờ thì tăng biến đếm thêm 1.

c. Code

int solution(string cell) {

    int count =0;

    char x = cell[0];

    char y = cell[1];

    if(x-2>='a'&&y+1<='8') count++;

    if(x-1>='a'&&y+2<='8') count++;

    if(x+1<='h'&&y+2<='8') count++;

    if(x+2<='h'&&y+1<='8') count++;

    if(x+2<='h'&&y-1>='1') count++;

    if(x+1<='h'&&y-2>='1') count++;

    if(x-1>='a'&&y-2>='1') count++;

    if(x-2>='a'&&y-1>='1') count++;

    return count;

}

**124. Bishop Diagonal**

a. Đề bài

Cho vị trí của 2 con tượng trong bàn cờ vua, nếu chúng cùng nằm trên 1 đường chéo thì di chuyển vị trí của chúng về cuối 2 đầu đường chéo, nếu không thì giữ nguyên. Trả về vị trí của 2 con tượng sau đó

b. Giải pháp

Kiểm tra xem 2 con tượng có cùng nằm trên 1 đường chéo không, nếu có thì xét vị trí của 2 con tượng rồi di chuyển 2 con tượng về 2 đầu của đường chéo

c. Code

vector<string> solution(string b1, string b2) {

    if (b2 < b1) swap(b1,b2);

    if (b1[0]-b2[0]==b1[1]-b2[1]){

        while (b1[0] > 'a' && b1[1] > '1'){

            b1[0]--;

            b1[1]--;

        }

        while (b2[0] < 'h' && b2[1] < '8'){

            b2[0]++;

            b2[1]++;

        }

    }

    else if (b1[0]-b2[0]==b2[1]-b1[1]){

        while (b1[0] > 'a' && b1[1] < '8'){

            b1[0]--;

            b1[1]++;

        }

        while (b2[0] < 'h' && b2[1] > '1'){

            b2[0]++;

            b2[1]--;

        }

    }

    return {b1,b2};

}

**125. Whose Turn?**

a. Đề bài

Cho 2 con mã đen và con mã trắng ở các vị trí xuất phát của nó trong bàn cờ vua ( mã trắng ở b1, g1 mã đen ở b8, g8). Cho 1 xâu gồm vị trí mới của 4 con mã sau 1 số nước đi, giả sử bên trắng đi trước, hỏi lượt đi tiếp theo là của bên nào?

b. Giải pháp

Sau mỗi lượt đi của con mã, tổng tọa độ của nó theo hàng và cột sẽ tăng hoặc giảm 1 số lẻ. Ban đầu tổng tọa độ của 4 con mã là 1 số chẵn (bên trắng đi trước) , vì vậy nếu tổng tọa độ của 4 con mã là số chẵn thì lượt đi tiếp theo là của bên trắng, nếu tổng tọa độ là số lẻ thì lượt tiếp theo là của bên đen.

c. Code

bool solution(std::string p)

{

    return (p[0] + p[1] + p[3] + p[4] + p[6] + p[7] + p[9] + p[10]) % 2 == 0;

}