1. **Chia để trị**
2. **Dãy con liên tục có tổng lớn nhất**

* Ý tưởng: Đặt M=(L+R)/2
  + Nếu dãy con cần tìm nằm ở nửa trái, được bài toán nhỏ hơn, dùng đệ quy để tìm
  + Nếu dãy con cần tìm nằm ở nửa phải, được bài toán nhỏ hơn, dùng đệ quy để tìm
  + Nếu dãy con nằm trùm lên cả hai nửa thì ta tự kiểm tra và duyệt
* Code:

#include<bits/stdc++.h>

using namespace std;

long Max(long \*a, int L, int R){

if(L==R) return a[L];

int M=(L+R)/2;

long ML=Max(a,L,M);

long MR=Max(a,M+1,R);

long TL= a[M], s=a[M];

for(int i=M-1;i>=L;i--){

s+=a[i];

if(TL<s) TL=s;

}

long TR = a[M+1], p=a[M+1];

for(int i=M+2;i<=R;i++){

p+=a[i];

if(TR<p) TR=p;

}

long MM = TR + TL;

return MM>ML && MM>MR ? MM : (ML>MR ? ML : MR);

}

int main()

{

int n;

long \*A;

cin>>n;

A = new long[n+5];

for(int i=1;i<=n;i++){

cin>>A[i];

}

cout<<"Day con lien tuc co tong max la: "<<Max(A, 1, n);

return 0;

}

1. **Đếm số nghịch thế**
   * Ý tưởng: Ta chia bài toán thành 2 phần M=(L+R)/2

* Nếu số nghịch thế nằm bên trái được bài toán nhỏ hơn, dùng đệ quy để tìm
* Nếu số nghịch thế nằm nửa bên phải, được bài toán nhỏ hơn, dùng nghịch thế để tìm
* Số nghịch thế có phần tử nằm nửa bên trái lớn hơn phần tử nằm nửa bên phải
* Chúng ta tính tổng cả 3 trường hợp để tìm ra tất cả các số nghịch thế
* Code:

#include<bits/stdc++.h>

using namespace std;

int x[100005];

long Nghichthe(int \*A, int L, int R){

if(L==R) return 0;

int M=(R+L)/2;

long d = Nghichthe(A,L,M) + Nghichthe(A, M+1, R);

int i=L, j=R;

for(int p=L;p<=R;p++)

if(i<=M && j<=R)

{

if(A[j]==A[i]) d+=M-i+1;

x[p]=A[i]<=A[j]?A[i++]:A[j++];

}

else x[p]=i<=M ? A[i++]:A[j++];

for(int p=L;p<=R;p++) A[p]=x[p];

return d;

}

int main()

{

int n,\*A;

cout<<"Nhap n: "; cin>>n;

A = new int[n+5];

for(int i=0;i<n;i++){

cout<<"A["<<i<<"] = ";

cin>>A[i];

}

cout<<"So nghich the la: "<<Nghichthe(A,1,n);

return 0;

}

1. **Cắt thanh kim loại**
   * Ý tưởng: Xây dựng cấu trúc dữ liệu cây nhị phân, mỗi nút quản lý các đoạn cắt gồm vị trí mút trái và mút phải và độ dài đoạn con dài nhất của nó.
     + Nếu bài toán suy biến: Cây chỉ có nút gốc là [a,b,M] trong đó a là mút trái, b là mút phải, M là đoạn con dài nhất bằng b-a. Chúng ta cắt tại vị trí v nào thì sẽ thêm 2 cây con trái [a, v, v-a] và cây con phải [v, b, b-v].
     + Nếu bài toán chưa suy biến: Nút gốc [a, b, M] đã có 2 cây con, nếu vết cắt tại v<a thì cắt vào cây con trái, chúng ta gọi đệ quy vào cây con trái. Ngược lại v>a thì vào cây con phải, chúng ta gọi đệ quy vào cây con phải, sau đó cập nhật giá trị đọna con dài nhất của gốc dựa vào đoạn con dài nhất của 2 cây con trái và phải.

* Code:

#include"bits/stdc++.h"

using namespace std;

struct Tree {

int a,b,m;

Tree \*Left, \*Right;

Tree(int u,int v) {

a=u;

b=v;

m=b-a;

Left=Right=NULL;

}

};

void add(Tree \*T,int x) {

if(T->Left==NULL) {

T->Left=new Tree(T->a,x);

T->Right=new Tree(x,T->b);

} else {

if(x<T->Left->b) add(T->Left,x);

else add(T->Right,x);

}

T->m=max(T->Left->m,T->Right->m);

}

int main()

{

int m,n,x;

scanf("%d%d",&n,&m);

Tree \*T=new Tree(0,m);

for(int i=0; i<n; i++) {

scanf("%d",&x);

add(T,x);

printf("%d ",T->m);

}

}

1. **Xóa k chữ số của một số để được số lớn nhất**
   * Ý tưởng: Ta xét số nguyên dương nhưng dưới dạng một xâu các chữ số có độ dài n, ta muốn xóa đi k chữ số để được số lớn nhất
     + Trường hợp bài toán suy biến: k=0 không phải xóa gì cả, nếu k= độ dài của xâu thì xóa tất cả
     + Trường hợp chưa suy biến: k>0, ta thực hiện:

B1: Tìm vị trí giá trị lớn nhất đầu tiên trong k+1 chữ số đầu tiên

B2: Ta xét 2 TH

* TH1: Vị trí lớn nhất ở đầu tiên thì ta giữ lấy giá trị này và xuất ra màn hình, sau đó gọi đệ quy giảm đệ trị xóa đi k chữ số ở phần đuôi
* TH2: Vị trí lớn nhất ở vị trí m khi đó ta coi như xóa đi m-1 chữ số đầu và gọi đệ quy để xóa đi k- (m-1) chữ số.
* Code:

#include<bits/stdc++.h>

using namespace std;

void xoa(char \*x,int k) //xoa k phan tu cua xau x

{

if(k==strlen(x)) return;

if(k==0) {cout<<x; return;}

char \*p=x;

for(char \*q=x+1;q<=x+k;q++) if(\*q>\*p) p=q; //con tro p tro vao vi tri max

if(p==x)

{

cout<<char(\*x);

xoa(x+1,k);

}

else xoa(p,k-(p-x));

}

int main()

{

char x[1000];

int k;

cin>>x>>k;

xoa(x,k);

}

1. **Quay lui**
2. **Mọi con đường về 0**
   * Ý tưởng: Số nguyên dương n có thể phân tích thành rất nhiều số m<n, tiếp tục m lại sinh ra các số khác nhỏ hơn m, cứ như vậy tới bước cuối các số sinh ra luôn là số 0. Ta thấy các bước đều cơ bản giống nhau nên ta sẽ duyệt vòng lặp và kiểm tra việc sinh số này, nếu nó đúng thì thêm nó vào một cái cây, nếu sai thì đệ quy quay lại kiểm tra tiếp.
   * Code:

#include<bits/stdc++.h>

using namespace std;

long long zero(int n)

{

if(n==0) return 1;

long long d=0;

for(int a=1;a\*a<=n;a++)

if(n%a==0) d+=zero((a-1)\*(n/a+1));

return d;

}

int main()

{

int n;

cin>>n;

cout<<"So con duong la: "<<zero(n);

}

1. **Sắp xếp balo**
   * Ý tưởng:
   * Code:

#include<bits/stdc++.h>

using namespace std;

class Balo

{

int n,M,a[100],b[100],x[100],pa[100],Max=0;

void TRY(int k,int KT,int GT)

{

if(k==n)

{

if(GT>Max)

{

Max=GT;

for(int i=1;i<=n;i++) pa[i]=x[i];

}

}

else

{

x[k+1]=0; TRY(k+1,KT,GT);

x[k+1]=1; if(KT+a[k+1]<=M) TRY(k+1,KT+a[k+1],GT+b[k+1]);

}

}

public:void sol()

{

cout<<"Nhap n = "; cin>>n;

for(int i=1;i<=n;i++) cin>>a[i]>>b[i];

cout<<"Nhap kich thuoc ba lo M = "; cin>>M;

TRY(0,0,0);

cout<<"Gia tri lon nhat "<<Max;

for(int i=1;i<=n;i++)

if(pa[i]) cout<<"\nchon do vat "<<i<<" kich thuoc "<<a[i]<<" gia tri "<<b[i];

}

};

int main(){Balo B;B.sol();}

1. **Đổi tiền**
   * Ý tưởng: Ta kiểm tra lần lượt các trường hợp với các thông số là tổng loại tiền k, số tiền T, số tờ t…Chia thành 2 trường hợp:

* Nếu tổng loại tiền kiểm tra = tổng loại tiền ban đầu thì kiểm tra và in ra số tờ ít nhất
* Ngược lại, ta dùng đệ quy tiếp tục kiểm tra vị trí k+1, tổng tiền và số tờ tiền đã được cập nhật.
* Code:

#include<bits/stdc++.h>

using namespace std;

class DT

{

int n,m,a[100],x[100],Min=INT\_MAX;

public:

void nhap()

{

cout<<"Nhap so menh gia n = ";

cin>>n;

for(int i=1;i<=n;i++) cin>>a[i];

cout<<"Nhap so tien : "; cin>>m;

}

void TRY(int k,int T,int t)

{

if(k==n)

{

if(T==m && t<Min) Min=t;

return;

}

for(int z=0;z<=(m-T)/a[k+1];z++)

{

x[k+1]=z;

TRY(k+1,T+z\*a[k+1],t+z);

}

}

void sol()

{

nhap();

TRY(0,0,0);

if(Min==INT\_MAX) cout<<"Khong co cach nao doi duoc";

else cout<<"So to it nhat "<<Min;

}

};

int main()

{

DT M; M.sol();

}

1. **Tám hậu**
   * Ý tưởng: Xét tất cả các cột, thử đặt quân hậu 1 vào một cột, với mỗi cách đặt như vậy, xét tất cả các cách đặt quân hậu 2 không bị quân hậu 1 ăn, tiếp tục thử lại 1 cách và xét tiếp các cách đặt quân hậu 3...
     + Khi chọn vị trí cột j cho quân hậu thứ i, ta chọn ô (i,j) không bị các quân hậu đặt trước đó ăn, tức là phải chọn cột j còn tự do, đường chéo chính, phụ còn tự do.
     + Nếu i=n thì được 1 nghiệm, ngược lại gọi đệ quy với quân hậu tiếp theo

* Code:

#include<bits/stdc++.h>

using namespace std;

class Queen

{

int n,x[1000],dem=0;

map<int,bool> R, C,P; //R- danh dau cot, C -duong cheo chinh, P - duong cheo phu

void TRY(int k)

{

if(k==n)

{

printf("\n%-5d",++dem);

for(int i=1;i<=n;i++) printf("(%d,%d) ",i,x[i]);

return;

}

else

for(int t=1;t<=n;t++)

if(R[t]==false && C[k+1-t]==false && P[k+1+t]==false)

{

R[t]=C[k+1-t]=P[k+1+t]=true; //Tien

x[k+1]=t;

TRY(k+1);

R[t]=C[k+1-t]=P[k+1+t]=false; //lui

}

}

public: void sol()

{

cout<<"Nhap n = "; cin>>n; C[0]=true;

TRY(0);

}

};

int main()

{

Queen Q; Q.sol();

}

1. **Đường đi trong đồ thị**
   * Ý tưởng: Ta duyệt đồ thị đề cho, ta có điểm đầu, điểm cuối của đồ thị. Kiểm tra một giá trị k truyền vào nếu giá trị k đấy bằng điểm cuối thì in ra kết quả. Nếu chưa thì duyệt một thùng chứa chứa mảng đồ thị để kiểm tra, nếu chấp nhận thì cập nhật đường đi và đệ quy quay ngược lại để kiểm tra đỉnh tiếp theo.
   * Code

#include<bits/stdc++.h>

using namespace std;

class Graph

{

int n,x[100],d[100],s,f;

vector<int> A[100]; //Danh sach ke

void nhap(string fn)

{

ifstream fin(fn);

fin>>n;

for(int i=1;i<=n;i++)

for(int j=1;j<=n;j++)

{

int x;

fin>>x;

if(x==1) A[i].push\_back(j); //i->j bo sung j vao danh sach ke cua i

}

fin.close();

}

void TRY(int k)

{

if(x[k]==f)

{

for(int i=1;i<k;i++) cout<<x[i]<<"->"; cout<<x[k]<<"\n";

return;

}

for(auto t:A[x[k]]) //C++11

if(d[t]==0)

{

x[k+1]=t;

TRY(k+1);

}

}

public: void sol()

{

nhap("g.txt");

cout<<"Nhap dinh xuat phat s= "; cin>>s;

cout<<"Nhap dinh ket thuc f= "; cin>>f;

x[1]=s;

fill(d,d+n+1,0);d[s]=1;

TRY(1);

}

};

int main() {Graph G; G.sol();}

1. **Giải ô chữ sodoku**
   * Ý tưởng: Duyệt mảng 9x9, nếu ô đấy khác 0 thì đánh dấu và kiểm tra ô đấy theo hàng, cột, zone. Nếu hợp lệ thì điền số và kiểm tra ô tiếp theo, nếu không hợp lệ đệ quy quay lại tăng giá trị lên để kiểm tra tiếp với giá trị khác.
   * Code

#include "stdio.h"

#include "conio.h"

#include "iostream"

#include "math.h"

#include <iomanip>

#define n 9

using namespace std;

class So

{

private :

int A[9][9];

public :

int KiemTra(int k, int t);

void Try(int k);

void NhapFile(char \*fn)

{

FILE \*f = fopen(fn,"rt");

if (!f)

{cout<<"Loi mo file";

return;

}

int i,j;

for( i=0; i<=n-1 ; i++)

for(j = 0; j<=n-1;j++)

fscanf(f,"%d",&A[i][j]);

fclose(f);

}

void Xuat()

{

cout<<"Xuat Ma Tran\n";

for(int i=0; i<=n-1 ; i++)

{

for(int j = 0; j<=n-1;j++)

cout<<A[i][j]<<" ";

cout<<"\n";

}

}

};

int So::KiemTra(int k, int t)

{

int i,j;

for(i=0;i<n;i++)

{

if (A[i][k%n]==t) return 0;

if (A[k/n][i]== t) return 0;

}

int u = k/n, v = k%n;

for(i = u/3\*3;i< u/3\*3+3;i++)

for(j = v/3\*3;j<v/3\*3+3;j++)

if(A[i][j]==t) return 0;

return 1;

}

void So::Try(int k = -1)

{

if(k==80){

Xuat();

getch(); return;

}

k++;

if(A[k/n][k%n]) Try(k);

else

{

for(int t = 1; t<=n ; t++)

if(KiemTra(k,t))

{

A[k/n][k%n]=t;

Try(k);

A[k/n][k%n]=0;

}

}

}

int main()

{

So T;

T.NhapFile("input.txt");

T.Xuat();

T.Try();

getch();

}

1. **Tham lam**
2. **Tô màu đồ thị**
   * Ý tưởng: Đầu tiên ta xét các đỉnh theo thứ tự và gán cho mỗi đỉnh một màu riêng theo nguyên tắc: các đỉnh không kề với đỉnh đang xét (không có cạnh nối trực tiếp) thì được phép tô cùng một màu, cấm tô màu đó cho các đỉnh có cạnh kề với đỉnh đang xét. Thuật toán lặp lại cho đến khi tất cả các đỉnh được tô màu.
   * Code:

#include<bits/stdc++.h>

using namespace std;

class Graph

{

int n,m,a[100][100];

pair<int,int> B[100]; //first ->dinh, second->bac

public:

void nhap(string fn="g.txt")

{

ifstream fin(fn);

fin>>n>>m;

for(int i=1;i<=n;i++)

for(int j=1;j<=n;j++) a[i][j]=0;

for(int i=1;i<=m;i++)

{

int u,v;

fin>>u>>v;

a[u][v]=a[v][u]=1;

}

fin.close();

}

void bac()

{

for(int i=1;i<=n;i++)

{

int t=0;

for(int j=1;j<=n;j++) t+=a[i][j];

B[i]={i,t};

}

}

void tomau()

{

int k=0,d[100]; //d - danh dau cac dinh da to

vector<int> Mau[n+1];

fill(d,d+n+1,0);

bac();

sort(B+1,B+n+1,[](pair<int,int> u,pair<int,int> v){return u.second>v.second;}); //sap bac giam dan

for(int i=1;i<=n;i++)

if(d[B[i].first]==0)

{

d[B[i].first]=++k; //to mau moi

Mau[k].push\_back(B[i].first);

for(int j=i+1;j<=n;j++)

if(d[B[j].first]==0)

{

int ok=1;

for(int z=1;z<=n;z++) if(d[z]==k && a[z][B[j].first]==1) {ok=0;break;}

if(ok) {d[B[j].first]=k; Mau[k].push\_back(B[j].first);}

}

}

cout<<"\nSo mau it nhat "<<k;

for(int i=1;i<=k;i++)

{

cout<<"\nMau "<<i<<" gom : ";

for(int m:Mau[i]) cout<<m<<" ";

}

}

};

int main(){Graph G; G.nhap(); G.tomau();}

1. **Sắp xếp balo**
   * Ý tưởng: Sắp xếp các đồ vật giảm dần theo giá trị, sau đó sắp xếp vào balo đến khi full thì thôi.
   * Code:

#include<bits/stdc++.h>

using namespace std;

struct dovat{int id;double kt,gt;};

bool ss(dovat u,dovat v)

{

double x=u.gt/u.kt,y=v.gt/v.kt;

if(x==y) return u.gt>v.gt;

return x>y;

}

class Balo

{

int n,M;

dovat A[1000];

public:

void nhap()

{

cout<<"Nhap so do vat : "; cin>>n;

for(int i=1;i<=n;i++)

{

A[i].id=i;

cout<<"Do vat "<<i<<" : ";

cin>>A[i].kt>>A[i].gt;

}

cout<<"Kich thuoc ba lo :"; cin>>M;

sort(A+1,A+n+1,ss);

}

void sapxepbalo()

{

int K=0,G=0;

for(int i=1;i<=n;i++)

if(K+A[i].kt<=M)

{

cout<<"\nLay vat "<<A[i].id<<" kich thuoc "<<A[i].kt<<" gia tri "<<A[i].gt;

K+=A[i].kt;

G+=A[i].gt;

}

cout<<"\nTong kich thuoc "<<K<<"\nTong gia tri "<<G;

}

};

int main(){Balo B; B.nhap();B.sapxepbalo();}

1. **Lập lịch**
   * Ý tưởng: Ta tính toán sự chênh lệch thời gian và số công phải bỏ ra để thực hiện công việc đó. Sau đấy sắp xếp các công việc theo thứ tự không tăng về thời gian và công sức. Công việc nào tốn ít thời gian nhất và được nhiều lợi ích nhất thì chọn, hết thời gian đấy lại tiếp tục chọn công việc khác.
   * Code:

#include<bits/stdc++.h>

using namespace std;

struct sk{int id,bd,kt;};

class Lich

{

int n;

sk A[100];

public: void nhap()

{

cout<<"So sk : "; cin>>n;

for(int i=1;i<=n;i++)

{

A[i].id=i;

cout<<"Thoi gian : (bat dau, ket thuc) ";

cin>>A[i].bd>>A[i].kt;

}

sort(A+1,A+n+1,[](sk u,sk v){return u.kt<v.kt;});

}

void laplich()

{

int moc=INT\_MIN,k=0;

for(int i=1;i<=n;i++)

if(A[i].bd>=moc)

{

cout<<"\nChon sk "<<A[i].id<<" bat dau

"<<A[i].bd<<" ket thuc "<<A[i].kt;

moc=A[i].kt;

k++;

}

cout<<"\nSo su kien duoc chon "<<k;

}

};

int main(){Lich L; L.nhap();L.laplich();}

1. **Thuật toán Krusal với cây khung nhỏ nhất**
   * Ý tưởng: Thuật toán dựa trên mô hình xây dựng cây khung nhỏ nhất bằng thuật toán hợp nhất. Để xây dựng tập n-1 cạnh của cây khung nhỏ nhất, Krusal đề nghị cách kết nạp lần lượt vào các cạnh đó theo nguyên tắc:
     + Ưu tiên các cạnh có trọng số nhỏ
     + Kết nạp các cạnh khi nó không tạo chu trình với tập cạnh đã kết nạp trước đó

* Code:

#include<bits/stdc++.h>

using namespace std;

struct canh{int dau,cuoi,trongso;};

struct Graph

{

int n,m,D[1000]; //D - Disjoin - Set

canh A[1000];

void nhap(string fn="cay.txt")

{

ifstream fin(fn);

fin>>n>>m;

for(int i=1;i<=m;i++) fin>>A[i].dau>>A[i].cuoi>>A[i].trongso;

fin.close();

sort(A+1,A+m+1,[](canh u, canh v){return u.trongso<v.trongso;});

}

int goc(int x){return D[x]==0?x:goc(D[x]);}

void kruskal()

{

fill(D,D+n+1,0);

int T=0;

for(int i=1;i<=m;i++)

{

int u=goc(A[i].dau),v=goc(A[i].cuoi);

if(u!=v)

{

T+=A[i].trongso;

cout<<"\nChon canh "<<A[i].dau<<" - "<<A[i].cuoi<<" trong so "<<A[i].trongso;

D[u]=v; //ghep vao 1 cay;

}

}

cout<<"\nTong trong so cay "<<T;

}

};

int main(){Graph G; G.nhap();G.kruskal();}

1. **Quy hoạch động**
2. **Xâu con chung dài nhất**

* Ý tưởng: Ta có thể nhận ra được độ dài xâu con chung dài nhất của 2 xâu A và B bé hơn hoặc bằng min(|A|,|B|). Nếu một trong 2 xâu là rỗng thì xâu con chung là rỗng, bài toán này sẽ phụ thuộc vào i, j. Nếu i, j đều dương thì xét các trường hợp:
  + Nếu ký tự tại A và B giống nhau thì độ dài xâu con chung dài nhất C[i][j] = 1 + C[i-1][j-1]
  + Nếu không thì C[i][j] = max(C[i-1][j], C[i][j-1])
* Code:

#include<bits/stdc++.h>

using namespace std;

struct xcc

{

int n,m,C[100][100];

char x[100],y[100];

void buocthuan()

{

for(int i=1;i<=n;i++) C[i][0]=0;

for(int j=0;j<=m;j++) C[0][j]=0;

for(int i=1;i<=n;i++)

for(int j=1;j<=m;j++)

if(x[i]==y[j]) C[i][j]=1+C[i-1][j-1];

else C[i][j]=max(C[i-1][j],C[i][j-1]);

}

void buocnghich(int i,int j)

{

if(C[i][i]==0) return;

while(C[i][j]==C[i-1][j]) i--;

while(C[i][j]==C[i][j-1]) j--;

buocnghich(i-1,j-1); cout<<x[i];

}

void sol()

{

printf("Nhap x = "); fflush(stdin); scanf("%s",x+1); //Chu y nhap tu x1;

printf("Nhap y = "); fflush(stdin); scanf("%s",y+1); //Chu y nhap tu y1;

n=strlen(x+1); m=strlen(y+1);

buocthuan();

cout<<"\nDo dai xau con chung dai nhat "<<C[n][m]<<"\n";

cout<<"\nXau la : "; buocnghich(n,m);

}

};

int main(){xcc X; X.sol();}

1. **Cắt hình vuông ít nhất**
   * Ý tưởng:
   * Code:

#include<bits/stdc++.h>

using namespace std;

struct CHV

{

int n,m,C[100][100];

void buocthuan()

{

for(int i=1;i<=n;i++) C[i][1]=i;

for(int j=1;j<=m;j++) C[1][j]=j;

for(int i=2;i<=n;i++)

for(int j=2;j<=m;j++)

if(i==j) C[i][j]=1;

else

{

C[i][j]=i\*j;

for(int k=1;k<=i/2;k++) if(C[i][j]>C[k][j]+C[i-k][j])C[i][j]=C[k][j]+C[i-k][j];

for(int k=1;k<=j/2;k++) if(C[i][j]>C[i][k]+C[i][j-k])C[i][j]=C[i][k]+C[i][j-k];

}

}

void buocnghich(int i,int j,string p="\n")

{

if(i==j) cout<<p<<i<<"\*"<<j;

else

{

for(int k=1;k<=i/2;k++) if(C[i][j]==C[k][j]+C[i-k][j])

{

buocnghich(k,j,p+" ");

cout<<p<<i<<"\*"<<j;

buocnghich(i-k,j,p+" ");

return;

}

for(int k=1;k<=j/2;k++) if(C[i][j]==C[i][k]+C[i][j-k])

{

buocnghich(i,k,p+" ");

cout<<p<<i<<"\*"<<j;

buocnghich(i,j-k,p+" ");

return;

}

}

}

void sol()

{

cout<<"Nhap n, m : "; cin>>n>>m;

buocthuan();

cout<<"\n\nSo hinh vuong it nhat "<<C[n][m];

buocnghich(n,m);

}

};

int main(){CHV H;H.sol();}

1. **Thuật toán Floyd**
   * Ý tưởng: Nếu có đường đi từ i tới k và từ k tới j nhỏ hơn đường đi hiện tại từ i tới j thì ta sẽ cập nhật đường đi từ i tới j thành đường đi từ **i tới k cộng với từ k tới j**. Ta gọi k là đỉnh trung gian của i và j. Như vậy sau khi thực hiện thuật toán, sẽ có một số cạnh “ảo” được sinh ra, tức là các cạnh không nối trực tiếp giữa hai đỉnh.
   * Code:

#include<bits/stdc++.h>

using namespace std;

int inf=1e9;

struct graph

{

int n,a[100][100],p[100][100]; //p luu so mu

void nhap(string fn){

ifstream fin(fn);

fin>>n;

for(int i=1;i<=n;i++)

for(int j=1;j<=n;j++)

{

fin>>a[i][j]; if(a[i][j]==-1) a[i][j]=inf;

p[i][j]=0;

}

fin.close();

}

void buocthuan(){

for(int k=1;k<=n;k++)

for(int i=1;i<=n;i++)

for(int j=1;j<=n;j++)

if(a[i][j]>a[i][k]+a[k][j]) {a[i][j]=a[i][k]+a[k][j];p[i][j]=k;}

}

void buocnghich(int i,int j){

if(p[i][j]==0) cout<<i;

else

{

buocnghich(i,p[i][j]);

cout<<"->";

buocnghich(p[i][j],j);

}

}

void sol() {

nhap("g.txt");

buocthuan();

for(int i=1;i<=n;i++)

for(int j=1;j<=n;j++)

if(j!=i)

{

if(a[i][j]==inf) cout<<"\nKhong di duoc "<<i<<" den "<<j;

else

{

cout<<"\nDuong di min "<<i<<" den "<<j<<" la "<<a[i][j]<<" : ";

buocnghich(i,j); cout<<"->"<<j;

}

}

}

};

int main(){graph G;G.sol();}

1. **Tam giác số**
   * Ý tưởng: Thực hiện quy hoạch động đặt C[i][j] là tổng lớn nhất của mọi đường đi từ đỉnh tới ô (i,j) ta có. Tại đỉnh: C[1][1] = A[1][1], trên 2 cạnh bên

C[i][1]=C[i-1][1] + A[i][1]

C[i][i]=C[i-1][i-1]+A[i][i]

Các vị trí còn lại:

C[i][j]=A[i][j] + Max(C[i-1][j-1], C[i-1][j])

Cuối cùng ta đi tìm max của hàng n ma trận C.

* Code:

#include<bits/stdc++.h>

using namespace std;

struct TGS

{

int n,a[100][100],C[100][100];

void nhap(){

cout<<"n = "; cin>>n;

for(int i=1;i<=n;i++)

for(int j=1;j<=i;j++){cout<<"a["<<i<<","<<j<<"]= ";cin>>a[i][j];}

}

void xuat(){

for(int i=1;i<=n;i++)

{

for(int j=1;j<=i;j++) cout<<setw(7)<<C[i][j];

cout<<endl;

}

}

void buocthuan(){

C[1][1]=a[1][1];

for(int i=2;i<=n;i++)

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

if(j==1) C[i][j]=C[i-1][j]+a[i][j];

else if(j==i) C[i][j]=C[i-1][j-1]+a[i][j];

else C[i][j]=max(C[i-1][j-1],C[i-1][j])+a[i][j];

}

void buocnghich(int i,int j){

if(i==1) {cout<<a[i][j];return;}

if(j==1) {buocnghich(i-1,j); cout<<"+"<<a[i][j];return;}

if(j==i) {buocnghich(i-1,j-1); cout<<"+"<<a[i][j];return;}

if(C[i-1][j-1]>C[i-1][j]) {buocnghich(i-1,j-1); cout<<"+"<<a[i][j];return;}

buocnghich(i-1,j); cout<<"+"<<a[i][j];

}

void sol(){

nhap();

buocthuan();

cout<<"\nMa tran QHD\n\n";

xuat();

int m=1;

for(int j=1;j<=n;j++) if(C[n][m]<C[n][j])m=j;

cout<<"\n\nTong lon nhat :"<<C[n][m]<<"\n";

buocnghich(n,m);

}

};

int main(){TGS T;T.sol();}