FPHÂN TÍCH THIẾT KẾ THUẬT TOÁN

(Thầy PHẠM XUÂN TÍCH)

**Chương I : Tổng quan thuật toán**

1. **Thuật toán**
* Khái niệm: Thuật toán là 1 dãy các thao tác biến đổi từ dữ liệu vào thành dữ liệu ra sau hữu hạn bước.



VD:

* Thuật từ Euclid tìm ước chung lớn nhất (UCLN)

UCLN(a, b) = UCLN(a-b, b)

* Thuật chia Euclid mở rộng:

d = (a, b) => £ x, y Z

chứng minh : ax + by = d;

nếu d =1 => x = a-1 mod b (nghịch dảo model)

Code minh họa thuật từ Euclid:

int UCLN(int a, int b){

 while(a !=b){

 if(a > b) {

 a - =b;

 b- =a;

 }

 }

 return a;

Code minh họa thuật toán chia Euclid:

int UCLN(int a, int b){

 while(b){

 int r = a % b;

 a = b;

 b = r;

 }

 return a;

}

1. **Đặc trưng của thuật toán**
2. **Tính xác định**

Mỗi thuật toán được gắn với một bài toán cụ thể được xác định đầu vào và đầu ra.

VD: ax2 + bx + c =0

BT1: Tìm nghiệm phân số của phương trình khi a,b,c là

1. Các phân số.
2. a,b,c € R.
3. a,b,c € C.

Code minh họa phần là số phức (Giải phương trình trên trường hữu tỷ):

#include <complex>

#include <cmath> //khai can so phuc

#include <iostream>

using namespace std;

typedef complex<float> sp;

int main(){

 sp a,b,c,d,x1,x2;

 sp k(0, 0);//ham tao

 cout<<"Nhap a "; cin>>a.real() >>a.imag();

 cout<<"Nhap b" ; cin>>b.real()>>b.imag();

 cout<<"Nhap c" ; cin>>c.real()>>c.imag();

 if( a == k ){

 if(b == k){

 cout<<"Vo so nghiem";

 else cout<<"Vo nghiem";

 }

 else cout<< -c/b;

 }

 else {

 d = sqrt(b\*b - 4\*a\*c);

 if(d == 0) cout<< "Nghiem kep" << -b/sp(2,0)/a;

 else

 {

 x1 = (-b - d)/sp(2,0) /a;

 x2 = (-b + d)/sp(2,0)/a;

 }

 }

}

1. **Tính cấu trúc**
* Mỗi thuật toán đều tuân thủ các cấu trúc chặt chẽ.
* Các bước theo trình tự nhất định, bước nào trước thực hiện trước.

VD: Sàng nguyên tố Eratostheno

BT: Cho n € Z+ . Tìm tất cả các số nguyên tố <= n

(GT: GoldBack: “Mọi số tự nhiên lớn hơn hoặc bằng 6 đều là tổng của 3 số nguyên tố”).

*BT Thêm 1: Cho n là 1 số tự nhiên (1<= n <= 105). Hãy đếm mọi cách phân tích n thành 3 số nguyên tố.*

*BT Thêm 2: Cho 1<n <=m <= 10^9. Đếm xem từ m -> n có bao nhiêu số nguyên tố với 0< m – n< 10 ^5.*

Gợi ý:

 B1: sang nguyên tố từ 2 ->

 B2: Tìm bội [n,m]

\*\*\*\*\* Thuật toán sàng nguyên tố

* Các bước thực hiện:

B1: 2,3,4……….n lên sàng

B2: Duyệt từ 2 -> n

Nếu nó trên sàng thì là số nguyên tố và loại các bội của nó khỏi sàng.

* Ý tưởng thuật toán :

+ Sàng nguyên tố là thuật toán do Eratosthenes đưa ra để tìm các [số nguyên tố](http://www.nguyenvanquan7826.com/2014/12/08/cac-thuat-toan-kiem-tra-nguyen-algorithm-check-prime-number/). Nó có đặc điểm khác với thuật toán khác là kiểm tra các số nguyên tố theo kiểu sàng lọc, xét tất cả những số cần kiểm, những số nào không phải là số nguyên tố thì bỏ đi. Thuật toán thích hợp cho bài toán tìm tất cả các số nguyên tố trong khoảng [a, b] mà đặc biệt hiệu quả khi khoảng cách giữa a, b là rất lớn.

+ Để tìm các số không phải số nguyên tố chỉ cần dựa vào các số nguyên tố ban đầu, VD số 2 là số nguyên tố thì các số chia hết cho 2 chắc chắn không phải là số nguyên tố, số 3 là số nguyên tố thì tất cả các số là bội của 3 đều bị loại bỏ, cứ như vậy những số được giữ lại là các số nguyên tố.

Code minh họa:

#include<stdio.h>

#include<iostream>

using namespace std;

void sang(int n,int \*p)

{

 bool \*s= new bool[n+5];

 p= new int[n+5];

 for(int i=1;i<=n;s[i+=2]=1);

 s[2]=1;

 int t=0;

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

 if(s[i]==1)

 {

 p[++t]=i;

 cout<<p[t]<<"\t";

 for(int j=i\*i;j<=n;j+=i)

 s[j]=0;

 }

}

int main()

{

 int n;

 int \*p;

 cin>>n;

 cout<<"so nguyen to la: "<<endl;

 sang(n,p);

}

1. **Tính dừng**
* Thuật toán dừng sau hữu hạn bước.

VD: Tính UCLN(a, b)

int gcd( int a, int b){

 while (b){

 int r =a%b;

 a =b;

 b=r;

 }

 return a;

}

1. **Tính đa dạng**
* Mỗi bài toán có nhiều cách giải khác nhau(có nhiều thuật toán) mong muốn xác định được thuật toán tốt nhất.

VD: Các thuật toán sắp xếp:

+ Quick sort

+ Merge short

+ Heap short

+ Bubble sort

+ Insert sort

+ Select sort

+ comb sort

+ Radix sort

+ Couting sort

Bài toán: Cho a1,a2,…..,an [1;k]

B1: Đếm tần suất tăng dần:

+ bao nhiêu số 0 🡪 d[0]

+ bao nhiêu số 1 🡪 d[1]

+……………………….

+bao nhiêu số k 🡪 d[k]

B2: Viết lại a:

a1 , a2, a 3, ………….,ak. (ví dụ có 3 số 0)

0 0 0

void sort(int n, int \*a, int k){

 int \*d = new int[k+5]; //khởi gán đếm

 for(int i= 0; i<=k; d[ i++]=0)

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

 int t =0;

 for(int i = 0; i<=k ; i++) //viết lại mảng

 for(int j = 1; i<= d[i] ; j++)

 a[++t]=i;

}

*Bài tập làm thêm*

1. *Cho a1,a2,…..,an[n;k] code sắp xếp Counting sort.*
2. *Sắp xếp xâu: Cho xâu x chữ hoa có độ dài<=10^6. Sắp xếp các kí tự tang dần.*
3. **Tính phổ dụng**
* Một tư tưởng thuật toán hóa có thể áp dụng giải nhiều bài toán

VD: Tìm kiếm nhị phân

Bài toán: Cho dãy a1,a2,…..,an đã sắp xếp tang dần. Cho x chỉ ra 1 vị trí của x tăng dần.

Code minh họa:

 int Find(int n, int\*a, int x){

 int L =1, R =n ;

 while(L<=R){

 int M = (L + R)/2;

 if(x==a[M]) return M;

 if(x < a[M] ) R= M – 1;

 else L = M + 1;

}

return 0;

}

*Bài tập thêm: Cho a1,a2,…..,an tang dần và x.*

1. *Tìm phần tử gần với x nhất.*
2. *Tìm phần tử nhở hơn nvaf gần với x nhất.*

Bài toán: Cho f C[a,b].

f(a)\*f(b) <=0

Tìm x [a,b] để f(x) = 0.

Code:

double chiadoi(double a, double b,double f(double), double ep =0,0001)

{

 while (b – a > ep ){

 double c = (b + a)/2;

 if( f(a)\* f(c) <= 0 ) b= c;

 else a =c;

}

return (b +a)/2;

}

VD: sin x = 0 [3,3.5]

Code nối tiếp trên:

double f(double x){

 return pow(x,x) -10 ;

}

int main(){

 cout<<”pi = ” << chiadoi(3,3.5,sin,0.0001);

 cout<<”Nghiem ” << chiadoi(2,3,7);

}

1. **Tính hình thức hóa**
* Thuật toán chỉ để sử dụng nếu biểu diễn bằng ngôn ngữ hình thức và ngôn ngữ lập trình.

VD:

\*\*\* ý tưởng: Tính diện tích hình giới nội

****

Với x=a, x=b, y=0, y =f(x).

* Phương pháp hình thang

Phân đoạn [a,b] thành những đoạn bằng nhau

 h= (b -a)/n

Biểu diễn các điểm chia:

 x0 = a

 x1= a – h

 x2 = a +h + h = a+2h

 …………………

 xi = a +ih

 xn = a +nh =b

**

=>0 +*f*1) +(*f1* + *f2*)+ … + (*fn-1 + fn*)]

= [f0 + 2f1 + 2f2 + … + 2fn-1 + fn]

I =h[+ (f1 + f2 + … + fn-1 )+ ] //diện tích hình thang

I =

Code minh họa:

double hinhthang (double a,double b,double f(double),int n)

{

 double h =(b – a)/n;

 double S = 0;

for (int i =1; i < n; i++) S += f(a +i \* h);

return (S + f(a)/2 + f(b)/2) \* h;

}

VD:

Code minh họa:

double bp(double x){return x \*x ;}

double ff(double x){return pow(x,x);}

double f(double x) {return 4/(1+ x\*x );}

int main(){

 cout<<”I1= ”<<”Hinh thang(0,1,bp,1000);

 cout<<”I2= ”<<”Hinh thang(0,1,ff,10000);

 cout<<”Pi= ”<<”Hinh thang(0,1,f,1000000);

}

1. **Độ phức tạp**
2. **Khái niệm**

Độ phức tạp của 1 thuật toán là nguồn tài nguyên mà thuật toán đó sử dụng. Gồm tài nguyên về không gian và tài nguyên về thời gian.

+ Tài nguyên về không gian: là số byte

Chứa các dữ liệu

\* Input

 \* Out put

 \* Các biến mạch trung gian mà thuật toán đó sử dụng

+ Tài nguyên về thời gian: số phép toán cơ bản mà thuật toán tính toán.

+ bit: >>,<<,&,|,^,!

+ so sánh: >, < , >=, <= ,==, !=

+ logic: &&, ||, !

+ số học: +, - ,\* ,/

+ gán: =, ++,--

+tham chiếu: [ ]

+ cấp phát, giải phóng : return.

VD: Tính số Fibonaci:

Code minh họa:

long fib(int n){

 long F= new long[n+1];

 F[0] = F[1] = 1;

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

 F[k] = F[k-1] + F[k-2];

 long x= F[n];

delete F;

return x;

}

Độ phức tạp:

* Không gian:

 K(n) = sizeof(n) +sizeof(F) + sizeof(k) + size of(x);

 = 4 +(n+1)\*4 +4 +4

 = 4n +16 = O(n)

T(n) = 4 +4 +9n -7 +2 +1 +1 = 9n +5 =O(n)

\*\*\*\*\*Đánh giá độ phức tạp chỉ quan tâm đến thời gian và ít quan tâm đến không gian.

Có 2 cách đánh giá:

+ Lí thuyết là đếm bước.

+ Thức nghiệm là chạy và đo thời gian.

*Bài tập thêm: Xét không gian và thời gian của đoạn code sau:*

*long fib(int n){*

 *long a=1 ,b=1, c;*

 *for(int k=2; k <= n; k++ ) //for(;n>=2;n--)*

*{*

 *c= a + b;*

 *a = b;*

 *b = c;*

*3*

 *return b;*

*}*

// có thể xét không có k.

1. **Các kí hiệu đánh giá: O, Ω, ϴ.**

Định nghĩa: Giả sử f : R+ 🡪 R+ hàm xác định dương.

-Cận trên: O(f(n)) = {g(n) | Ǝ n0 đủ lớn Ǝ c R+ 0ta có g(n) c f(n)}

- Cận dưới: Ω(f(n)) = {g(n) | Ǝ n0 đủ lớn Ǝ c R+ 0 ta có c f(n) g(n)}

- Cận chặt: ϴ(f(n)) = {g(n) | Ǝ n0 đủ lớn Ǝ c1,c20 ta có :

C1 f(n) g(n)<c2f(n)}

VD: Cận trên

 3n2 + 2n +5 = O(n3)Vì tồn tại n0 =1 ,c =10 : 3n2 + 2n +5 3

8n + 100= O(n3) Vì tồn tại n0 =1 ,c =108 : 8n +100 3

2) vì tồn tại n0 =1 ,c =1 : n2/2 +n/2 < cn2

log2n = O() vì tồn tại n0 = 64, c=1: log2 n 0

VD: Cận dưới:

2)

Vì tồn tại n0 =1 , c= 1/8:

n2/2 +n/2 >c 2n2

⬄ n2/2 +n/2 >n2

VD: Cận chặt:

2)

Vì tồn tại n0 = 1, c1 =1, c2 = 1:

c1n2 c2n2

\*\*\*\*\* Chú ý:

- Khi đánh giá độ phức tạp thì dựa vào kích cỡ Input

- Khi đánh giá tiệm cận là đánh giá khi kích cỡ Input đủ lớn

VD: = (log2 n)

c1 log2 n log3 n 2log2 n ;

với log3 n = log3 2 \* log2 n

1. **Tính chất**

**\*\*\* Chuyển đổi**

**-** Nếu f =O(g) thì g = Ω (f)

- Nếu f = Ω (g) thì g = O(f)

**\*\*\* Phản xạ**

f = O(c.f)

f = Ω(c.f)

f =

**\*\*\* Đối xứng**

f = => g =

**\*\*\* Tính chất bắc cầu**

Nếu f = O(g) và g =O(h) thì f =O(h)

 f= Ω(g) và g = Ω (h) thì f = Ω (h)

 f = (g) và g = thì f =

\*\*\* Giao

(f) = O(f) Ω(f)

VD:

void Sort(int n, int \*a){

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

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

 if(a[i] >a[j]){

 int t = a[i];

 a[i] = a[j];

 a[j] =t;

 }

}

Cách tính độ phức tạp:

+) i=1, j: 2 🡪n :

min: 5(n -1) +2 +1 = 5(n-1) +3

max: 12(n -1) +2 +1 = 12(n-1) +3

+) i =2, j: 3 🡪 n:

min: 5(n-2) +3

max: 12(n-2) +3

…………………….

+) i= n-1; j =n:

min = 5 \*1 +3

max = 12 \*1 +3

=> 5(1 +2 +3 + … +n-1) + 3(n-1) +2 T(n) 12(1+2 +3 + … +n-1)+3(n-1)+2

⬄+ 3n - 1 T(n) + 3n - 1

⬄+ 3n - 1 T(n) 6n2- 3n - 1 6n2

=> T(n) = (n2)

1. **Đánh giá độ phức tạp của thuật toán**
2. **Đánh giá thuật toán lặp**

+ Quy tắc cộng

Nếu thuật toán A được biểu diễn bới chương trình P gồm 2 đoạn chương trình P1 và P2 rời nhau thì:

TA(n) = TP(n) = TP1(n) + TP2(n) =Max(TP1(n),TP2(n))

+ Quy tắc nhân:

Nếu thuật toán A được biểu diễn bởi chương trình P gồm 2 đoạn chương trình P1 và P2 lồng nhau thì:

TA(n) = TP(n) =TP1(n) \* TP2(n)

VD: Bài toán nội suy:

BT: Cho các mốc x0,x1,x2,…xn. Có bảng sau:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| X | x0 | x1 | ….. | xn |
| f(x) | y0 | y1 | …… | yn |

Nội suy bằng đa thức Langrandge

=>

=> y = +y0

= y1 – y0 + y0

= y1 + y0 (1 - )

 =y1 + y0

3 mốc :

|  |  |  |  |
| --- | --- | --- | --- |
| X | x0 | x1 | x2 |
| Y | y0 | y1 | y2 |

y = y0 +y1 +y0

**Tổng quát:**

|  |
| --- |
| Y = (với i) |

Chương trình:

void Nhap(){

 cout<<”Nhap n =”;

 cin>>n;

 x = new float[n+1];

 y = new float[n+1];

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

 cout<< “x[ ” << i <<” ] = ”; cin>>x[i];

 cout<<”y[ ” << i<< “] =” ; cin >>y[i];

 }

}

 float Langrandge(int n, float \*x, float \*y, float X){

 float L =0;

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

float P =1;

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

 if(j!=i) p= p\*(x – x[j])\*(x[i] – x[j]);

 L += P \* y[i];

 }return L;

 }

int main(){

 int n;

 float \*x,\*y,X,Y;

 Nhap(n,x,y);

 cout<< “X=” ; cin >> X;

cout << “Y=” ; cin >>Y;

 cout<< Langrandge(n,x,X,Y);

 }

Độ phức tạp T(n) = TNhap(n) + Tlangrandge(n) =ϴ(n) +ϴ(n2) = ϴ(n2)

1. **Đánh giá thuật toán Đệ Quy**

VD1: Tính n!

 n! =

Code minh họa:

long gt(int n)

{

 if(n == 0) return 1;

 return gt(n-1) \*n;

}

Độ phức tạp:T(n) =

=> Tổng quát: T(n) =

T(n) = T(n-1) + c

 = T(n-2) +c +c

 = T(n-3) + 3c

 = ……………..

 = T(n-n) + nc

 = co +nc = (n)

VD2: tính độ phức tạp của đoạn code sau:

int MULT(int x,int n){

 if(n == 0) return 0;

 if(n == 1) return x;

 if(n % 2 == 0) return MULT(x\*x, n/2);

 return x \* MULT(x\*2,n/2);

}

Độ phức tạp:

Gọi T(n) ==

T(n) = T( + c

 = T ( + 2c

 ………………

 = T( + hc = T(1) + c log2 n = c0 + c log2 n

=> T(1) =(log n) ; h= log2 n

VD3: Tính độ phức tạp:

float(int n, float x, float y)

{

 if(n <= 1) return 1;

 float S =0 ;

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

 float T=1;

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

 T \*= (y – j \* x);

 S+= T \* x/y;

}

 return S+F(n/2,x,y/2) + F(n/4,x/2,y);

 }

}

Tính độ phức tạp:

T(n) =

T(n) =+ cn2

 = Cn2 T(n) 16/11 Cn2

C1 C2

=> T(n) =(n2)

VD4: Tính độ phức tạp

T(n) =

T(n) Cn

.

.

.

T(1) = T(n/2h)

 C nlog n =(nlog n)

=> h = nlog n

**c. Định lí Master**

T(n) =

- TH1: Nếu loga b >k thì:

T(n) =(nloga b )

- TH2: Nếu loga b = k thì:

T(n) =(nklog n )

- TH3: TH2: Nếu loga b < k thì:

T(n) =(nk)

\*\* logab =1 =k =>T(n) =(nlog n )

VD1:Xác đinh đô phức tạp

T(n) =

log4 3 < 3 => T(n) = (n3)

VD2:Xác định độ phức tạp

T(n) =

k = 0

a = 1

b =3/2

=> log3/2 1 = 0 = k

=> T(n) = (log n) (TH2)

**CHƯƠNG 2: PHÂN TÍCH THIẾT KẾ THUẬT TOÁN LẶP VÀ ĐỆ QUY**

**Quy trình phân tích thiết kế thuật toán**

B1: Phân tích bài toán Input và diều kiện bài toán

 Output và điều kiện của Output

B2: Xây dựng ý tưởng

- Thiết kế

- Cài đặt

B3: Tính đúng đắn

- Bất biến vòng lặp

- Thiết lập bất biến

- Duy trì bất biến

B4: Thoát

- Bất thường

- Bình thường

- Kết thúc

B5: Truy hồi

- Đệ quy

- Truy hồi

B6: Đánh giá Độ phức tạp

VD1: Cho n Z , x,y R

S =

Đặt t = sin x khi đó ta có:

S =

Đặt T(n) =

=> S= với T(n) =

1. **Đặc tả bài toán**
2. Input n Z , x,y R
3. Output

- Không tính được

- Tính được S=

1. **Các bước căn bản**

Đặt T(n) =

Pn = (-1)n

Qn = tn

Rn = yn

+ B0:

+ B1 :

+ B2:

.

.

.

+ Bi:

.

.

.

+ Bn:

Thiết kế:

Input n Z+ , x,y R

X =0 || y =0

 **+**

 **-**

Out Không tính được

P = Q =1

T =0; t = sin x

I = 1,2,3…..n

2016 – T/xy2

T + = P \*

P =-P

Q \* = t

R \* = y

Out

R

 **+ -**

Cài đặt:

Bool f(int a, float x, float y, float &S)

{

 if( x == 0 || y == 0) return 0;

 float t = sin x, p = 1, Q =1, R = 1, T =0;

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

P = -P;

Q \* = t;

R \* = y;

 if(R = = i) return 0;

 T += P \*(Q +1)/(R –i);

 }

 S = 2016 – T/x/y/y;

 if( S < 0) return 0;

 S = sqrt(S);

 retrun 1;

 }

1. **Tính đúng đắn**
2. **Bất biến vòng lặp**

Bi:

1. **Thiết lập**

B0:

Duy trì bất biến

Bi -1 => Bi

 Bi

1. **Thoát**

**+** Bất thường khi

+ Bình thường :

+ Kết thúc:

1. **Truy hồi**

 n= 0 thì:

+ Bn: n=0

+ Bn: n> 0

Cài đặt đệ quy:

bool t(int n, float x, float y, float & P,, float & Q, float & R, float &T)

{

 if(n = = 0){

 P =Q =R=1;

 T = 0;

 return 1;

 }

bool OK = t(n-1,x,y,P,Q,R,T);

 if(! OK) return false;

 P =- P;

 Q\* = t;

 R \* = y;

 if(R == n) return false;

 T += P\*(Q+1)/(R-n);

 return true;

}

1. **Đánh giá**
2. Thuật toán lặp do 1 vòng lặp
3. Thuật toán đệ quy

T(n) =

T(n) = T(n-1) +C

 = T(n-2) +2C

 = ………..

 = T(n-n) +nc =

**CHƯƠNG III: ĐỆ QUY**

1. **Đệ quy chia để trị(devide and conquew)**

- Chia để trị triển khai thuật toán Top – Down

- Chia bài toán lớn thành những bài toán con

- Dùng đệ quy để giải các bài toán con

- Liên kết các kết quả bài toán con để được kết quả của bài toán lớn

VD: Bài toán mọi con đường về 0

Số m = a\*b(a< = b)

Sinh ra n =(a -1)(b+1)



Cho m hỏi có bao nhiêu cách đi được đến 0

Code minh họa:

long Zero(int n){

 if(n == 0) return 1;

 long d=0;

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

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

 return d;

}

*Bài tập thêm 1:Cho m,n đếm số đường đi từ m ->n*

*Bài tập thêm 2: Dùng mảng đánh dấu để kiểm soát những con đường đã đếm thì không phải tính lại.*

1. **Cấu trúc chia để trị**

- Suy biến giải quyết luôn

- Chia để trị

+ Chia bài toán lớn thành những bài toán con

+ Trị: dung đệ quy

+ Liên kết kết quả các bài toán con được kết quả bài toán lớn

VD1: Chia nhóm: Có n người chia thành 2 nhóm và mỗi nhóm chếnh nhau đứng k người. Hỏi từ n người ban đầu chia được bao nhiêu nhóm?

- Phân tích: ⬄

Code minh họa:

long Nhom(long n,int k){

 if(n<= k || (n +k )% 2) return 1;

 return Nhom((n +k)/2, k) + Nhom((n -k)/2, k);

}

VD2:Bài toán có n mục tiêu cần quan sát trên mặt phẳng tọa độMỗi 1 camera thì quan sát được hoặc là song song với Ox hoặc là song song với trục Oy.

Code minh họa

#include<stdio.h>

#include<conio.h>

#include<list>

typedef struct Toado{

 long x,y;

 Toado (long a, long b){

 x =a; y =b;

 }

}Toado;

 typedef list<Toado> LIST;

void Nhap(LIST &L){

 int n;

 scanf("%d",&n);

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

 long u,v;

 scanf("%ld%ld",&n,&v);

 L.pust\_back(Toado(n,v));

 }

}

long Camera(LIST L){

 if(L.size() <= -1) return L.size();

 LIST L1,L2;

 Toado P = L.front();

 for(list <Toado>::iterator id = L.begin(); id!=L.end(); id++)

 {

 if(\*id.x!= P.x) L1.push\_back(\*id);

 if(\*id.y!= P.y) L2.pust\_back(\*id);

 }

 long c1 = Camera(L1);

 long c2 = Camera(L2);

 return(c1<c2?c1:c2) + 1;

}

int main(){

 LIST L;

 Nhap(L);

 Camera(L)

}

**b.Một số bài chia để trị**

\*\*\*Bài toán 1:Lũy thừa

Cho a là số thực , n là số nguyên dương, tính an

an=

Code minh họa:

double Power(double a, int n){

 if(n = =0) return 1;

 double t = Power(a, n/2);

 t = t\*t;

 return n% 2? a\*t : t;

}

Tính độ phức tạp:

T(n) – độ phức tạp thuật toán

T(n) =

TH1: Theo master: a=1, b=2, k =0.

loga b = 0 =k

TH2: T(n) =ϴ(log n)

VD2: Cho số:

Fn =

Phương trình đặc trưng:

r2 =r +1 ⬄ r2  - r – 1 = 0

⬄

=>Fn=

=>

⬄⬄⬄

⬄ Fn = (

Code minh họa:

long long Fib(int n){

 long double k =sqrt(S);

 return Power((1-k)/2, n+1) +Power((1-k)/2, n+1)

}

*Bài tập thêm1: Tính an mod m bằng phương pháp chia để trị.*

*Dựa vào (a\*b) mod m =(a mod m)(b mod m) mod m.*

*Bài tạp thêm 2: Cho x,n +*

*(1 + x + … + xn) mod (109 + 7)*

\*\*\*Bài toán: tìm hiểu nghịch đảo modul

Số Fibonaci:

Định lí:

 =

Chứng minh:

 =

 =

 = =

= =

Code minh họa:

void F(intn, long &a,long &b){

 if(n == 0){a =1; b = 0; return;}

 if(n == 1){a=b=1;return;}

 F(n/2,a,b);

 long c =a;

 a =a2 + b2;

b =2cb - b2;

 if(n%2) {

 a= a+b;

 b =a –b;

 }

}

**c.Chia để trị trên mảng**

1. Tìm kiếm nhị phân

Bài toán 1: Cho dãy a1an đã sắp xếp tăng dần.cho số x. Hỏi x có trong dãy không?

\*\*\*Ý tưởng: Tổng quát tìm x trong dãy aL,aL + 1, … ,aR-1,aR.

+ Suy biến L>R ta không có x.

+ chia để trị:

 \* Chia k= (L+R)/2

 \* Trị và lên kết nếu: ak =x 🡪 trả về k

 ak> x 🡪 Tìm aL……ak

ak> x 🡪 Tìm ak……aR

Code minh họa:

int Find(int L, int R, int \*a, int x){

 if(L>R) return 0;

 int k =(L+R)/2;

 if(a[k] == x) return k;

 if(a[k] >x ) return Find(L,k,1,a,x);

 return Find(k+1,R,a,x);

}

Độ phức tạp:

T(n) =

T(n) =ϴ(log n)

Bài toán 2: Tổng 1 dãy

Code minh họa:

int sum(int L, int R, int \*a){

 if(L>R) return 0;

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

 return sum(L,(L+R)/2, a) + sum((L+R)/2,R,a);

}

void Nhap(int L, int R, int \*a){

 if(L>=R) return 1;

 int K =(L+R)/2;

 Nhap(n,K-1,a);

 cout<<"a[" <<k<< "]=";

 cin>> a[k];

 Nhap(K+1,R,a);

}

bool Tang(int L,int R, int \*a){

 if(L >= R) return 1;

 int K =(L+R)/2;

 if(a[K]>a[K+1]) return false;

 if(! Tang(L,K-1,a)) return false;

 return Tang(K+1,R,a);

}

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

 if(L == R) return 1;

 int K =(L+R)/2;

 int m1= Max(L,K,a);

 int m2 = Max(k+1,R,a);

 return m1>m2 ? m1 :m2;

}

2.Chia để trị

a. Thuật toán sắp xếp MerSort

Bài tóan: Cho a1,…..,an phần tử so sánh được với nhau sắp xếp tang dần.

Ý tưởng :

Tổng quát: Sắp xếp aL,aL+1,…,aR.

+Suy biến: L>R -> xong L R

+ Chia để trị: a

 1.Chia k= (L+R)/2

 2.Trị dùng đệ quy

 Sắp xếp aL …………..ak.

 Sắp xếp ak+1………….aR

 3.Liên kết: Trộn 2 dãy tăng a1,…..ak Và ak+1………..aR

Minh họa hình ảnh:



#include<iostream>

using namespace std;

int x[100005];

void Mergesort(int \*a, int L, int R){

 int k=(L+R)/2;

 int i = L, j = k+1;

 if(L>=R)return;

 Mergesort(a,L,k);

 Mergesort(a,k+1,R);

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

 if(i<=k && j<= R) x[id] =a[i] <a[j]?a[i++]:a[j++];

 else x[id] = i <= k ? a[i++]:a[j++];

 for(int id = L; id<=R; id++){

 a[id] = x[id];

 }

}

int main(){

 int n;

 int \*A;

 cout<<"\nNhap so phan tu:";

 cin>>n;

 A=new int[n+5];

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

 cout<<"\nA["<<i<<"]=";

 cin>>A[i];

 }

 Mergesort(A,1,n);

 cout<<"\nDay sau khi sap xep la: \n";

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

 {

 cout<<A[i]<<"\t";

 }

}Đánh giá độ phức tạp:T(n) =

TH1: Theo định lí mater loga b = log2 2 =1 =k;

TH2: T(n) = Ѳ(nlog n)

b.Đếm số nghịch thế

Đếm số nghịch thế:

VD: 7 4 2 8 1 6 3

 5 3 1 0 0 1 0

Một nghịch thế là i < j mà a[i] > a[j] trong 1 dãy a1,….an

Tổng quát: Đếm số nghịch thế aL,aL+1,…,aR.

+ Suy biến: L>=R return 0;

+ Chia:



Số nghịch thế là tổng nghịch thế trái + phải +(trái và phải)



Code minh họa

#include<iostream>

using namespace std;

int x[100005];

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

 if(L>=R) return 0;

 int d= 0;

 int k =(L+R)/2;

 long d1 = NghichThe(a,L,k);

 long d2 = NghichThe(a,k+1,R);

 int i = L,j=k+1;

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

 if(i<=k && j<= R) {

 x[id] =a[i] <a[j]?a[i++]:a[j++];

 if(a[i]>a[j]) d+=k-i+1;

 }

 else x[id] = i <= k ? a[i++]:a[j++];

 for(int id = L; id<=R; id++){

 a[id] = x[id];

 }

 return d1+d2+d;

}

int main(){

 int n;

 int \*A;

 cout<<"\nNhap so phan tu:";

 cin>>n;

 A=new int[n+5];

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

 cout<<"\nA["<<i<<"]=";

 cin>>A[i];

 }

 cout<<"\nSo nghich the la:"<<NghichThe(A,1,n);

}

*Bài tập thêm: i < j => a[i] <a[j]*

c.Thuật toán Quicksort

Bài toán 1:Cho a1,…an. Sắp xếp tang dần.

Tổng quát: Sắp xếp aL,aL+1,…,aR.

+Suy biến: L>R -> xong

 1.Chia:

+ Chọn x bất kì trong aL,….aR.

+ Phân hoạch:



2.Trị

Gọi đệ quy sắp xếp aL,………..ak

Và ak+1,………..aR

3.Liên kết không làm gì.

Code minh họa:

#include<iostream>

using namespace std;

int Patition(int \*a,int L,int R){

 swap(a[L],a[(L+R)/2]);

 int x = a[L];

 int i = L;

 for(int j= L+1; j<=R;j++)

 if(a[j]<x){

 i++;

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

 }

 swap(a[L],a[i]);

 return i;

}

void QuickSort(int \*a,int L,int R){

 if(L >= R ) return ;

 int p =Patition(a,L,R);

 QuickSort(a,L,p-1);

 QuickSort(a,p+1,R);

}

int main(){

 int n;

 int \*A;

 cout<<"\nNhap so phan tu:";

 cin>>n;

 A=new int[n+5];

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

 cout<<"\nA["<<i<<"]=";

 cin>>A[i];

 }

 QuickSort(A,1,n);

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

 cout<<A[i]<<"\t";

}

\*\*Đánh giá độ phức tạp:

+ Tốt nhất: T(n) = 2T(n/2) + Cn

T(n) =Ѳ (n log n)

+ Xấu nhất: T(n) = T(n-1) +Cn

 = T(n-2) + Cn + C(n-1)

 = ….

 = T(0) + C1 +C2+….+Cn

 = Co +

 = Ѳ (n2)

+ Trung bình:

T(n) = T() +T() +Cn

 = Ѳ (n n) = Ѳ (n log n)

Bài toán 2:Tìm phần tử thứ K

Cho a1…..an (1<=k <= n).

Chỉ ra phần tử thứ k.

4 7 2 8 1 6

K=3 => 4

Tổng quát: Tìm phần tử thứ k trong aL……..aR

+ Suy biến L= R && k=1 => aL

Ý tưởng:

1.Chia phân hoạch:

2.Nếu p-L+1 =k trả về ap

 Nếu p-L+1 > k đệ quy phần tử k từ aL………ap-1

 Nếu p-L+1 > k đệ quy phần tử k từ ap+1………aR.

Code minh họa:

#include<iostream>

using namespace std;

int Patition(int \*a,int L,int R){

 swap(a[L],a[(L+R)/2]);

 int x = a[L];

 int i = L;

 for(int j= L+1; j<=R;j++)

 if(a[j]<x){

 i++;

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

 }

 swap(a[L],a[i]);

 return i;

}

int Find(int \*a, int L,int R,int k){

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

 int p= Patition(a,L,R);

 if(k == p-L+1) return a[p];

 if(k<p-L+1) return Find(a,L,p-1,k);

 else return Find(a,p+1,R,k-p+L-1);

}

int main(){

 int n,k;

 int \*A;

 cout<<"\nNhap so phan tu:";

 cin>>n;

 A=new int[n+5];

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

 cout<<"\nA["<<i<<"]=";

 cin>>A[i];

 }

 cout<<"\nNhap k = ";

 cin>>k;

 cout<<Find(A,1,n,k);

}

\*\*Độ phức tạp:

+ Tốt nhất: T(n) = T(n/2) +Cn

 T(n) = Cn

+ Xấu nhất: T(n) =Ѳ(n2)

Bài tập 3: Dãy con liên tục có tổng max

Cho dãy số nguyên a1,a2…..an Z+. Tìm dãy con liên tục có tổng max.



Tổng quát: aL,aL+1……..aR.

Suy biến: L = R =>aL

Chia:



k= (L+R)/2

Trị: Đệ quy M1 = Tổng lớn nhất aL………ak

M1 = Tổng lớn nhất ak+1………aR

Code minh họa:

#include<iostream>

using namespace std;

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

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

 int k=(L+R)/2;

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

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

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

 for(int i = k-1;i>=L;i--)

 {

 s+=a[i];

 if(TL <s) TL =s;

 }

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

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

 p+= a[i];

 if(TR<p) TR = p;

 }

 long M = TL+TR;

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

}

int main(){

 int n;

 long \*A;

 cout<<"\nNhap so phan tu:";

 cin>>n;

 A = new long[n+5];

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

 cout<<"\nA["<<i<<"]=";

 cin>>A[i];

 }

 cout<<"\nDay con lien tuc co tong MAx la: "<<Max(A,1,n);

}

*Bài tập thêm:Tìm dãy con có tổng min,max.Vị trí max từ…?*

1. **Quay lui (BackTracking)**

****

VD: Mọi con đường về 0

m = a\*b -> n=(a-1)(b+1)

* Cho n in ra mọi con đường về 0.

Phân tích: cho x1 -> x2 ->…..->xm

Sao cho 1. x1 = n

 2.xm = 0

 3.xi -> xi  + 1 (với xi =a\*b, xi+1 =(a-1)(b+1)

Code minh họa:

#include<iostream>

using namespace std;

void Try(int \*x, int k){

 if(x[k] == 0){

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

 cout<<"-> "<<x[i];

 cout<<endl;

 }

 else{

 for(int a=1;a\*a <= x[k];a++)

 if(x[k]%a ==0){

 int t =(a-1)\*(x[k]/a +1);

 x[k+1] = t;

 Try(x,k+1);

 }

 }

}

int main(){

 int x[100],n;

 cin>>n;

 x[1] = n;

 Try(x,1);

}

*Bài tập thêm 1:Hãy tìm mọi con đường từ n ->m*

*Bài tập thêm 2: Hãy tìm 1 đường từ m -> n*

1. **Cấu trúc thuật toán quay lui**

Thuật toán quay lui giải các bài toán có dạng tìm kết quả là một vector x1,x2,………xm.

Giả sử đi được từ x1,x2,………xk

+ Nếu tới đích => m kết quả.

+ Nếu chưa tới truy các khả năng đi tiếp cho x[k + 1] và đệ quy đi tiếp.

Minh họa:

Procedure Try(x1,x2,………xk)

Begin

 if(finish(x1,x2,………xk))

 then write(x1,x2,………xk)

 else

 foreach + in Next (x1,x2,………xk)

 do

 Begin

 X[k+1] = t;

 TRY(x1,x2,………xk);

 End

End.

VD: Sinh các dãy nhị phân độ dài n

n =3 => 000

 001

 ….

 111

Phân tích: x1,x2,………xn  sao cho:

1. xi
2. Dài n.

Code minh họa:

#include<iostream>

using namespace std;

void TRY(int \*x, int k, int n){

 if(k == n){

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

 cout<<x[i];

 cout<<endl;

 return;

 }

 for(int t=0;t<=1;t++){

 x[k+1] =t;

 TRY(x,k+1,n);

 }

}

int main(){

 int x[10005],n;

 cin>>n;

 TRY(x,0,n);

}

*Bài tập thêm:*

1. *In ra dãy nhị phân có độ dài n*

*+ 2 số đầu là: 11\_ \_ \_*

*+ -> Try(x,2,n)*

1. *Sinh dãy nhị phân dài n không có 2 số 1 liền nhau*
2. *Dãy b phân thêm Try(int \*x, int k,int n,int b =2)*
3. *Tập con: A={cho,ga,meo} in ra các tập con của a*
4. **Quay lui(BackTracking)**

Bài toán 1:Bài toán 8 con hậu

+ Bài toán đặt 8 con hậu lên bàn cờ 8 x 8 mà không con nào ăn con nào

+ Bài toán tổng quát: n con hậu trên bàn cờ n\*n

+ Ý tưởng quay lui

Hậu 1 hàng 1 cột x1

Hậu 2 hàng 2 cột x2

…………..

Hậu n hàng n cột xn

Tìm x1,x2,………xn

Sao cho: 1. 1n

 2.

 3.|

 Code minh họa:

Cách 1:

#include<conio.h>

#include<stdio.h>

#define nl printf("\n")

void Print(int \*x, int k){

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

 printf("(%d,%d)",i,x[i]);

 nl;

 }

}

bool Datduoc(int \*x,int k,int t){

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

 if(t == x[i]) return 0;

 if(k+1 -1 == t - x[i]) return 0;

 if(k+1 -i == x[i] -t) return 0;

 }

 return 1;

}

void Try(int \*x,int k,int n){

 if(k == n) {

 Print(x,k);

 return;

 }

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

 if(Datduoc(x,k,t)){

 x[k+1]=t;

 Try(x,k+1,n);

 }

}

int main(){

 int n, x[100], c[200],p[200],d[100];

 scanf("%d",&n);

 Try(x,0,n);

 getch();

}

Cách 2:

#include<stdio.h>

#include<conio.h>

#define nl printf("\n")

void Print(int \*x,int k)

{

 int static d=1;

 printf("%2d : ",d++);

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

 nl;

}

bool TRY(int \*x,int k,int n,int \*d,int \*c,int \*p)

{

 if(k==n) {Print(x,k);return 1;}

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

 if (d[t]==0 && c[k+1-t]==0 && p[k+1+t]==0)

 {

 x[k+1]=t;

 d[t]=c[k+1-t]= p[k+1+t]=1; //tien

 if(TRY(x,k+1,n,d,c,p)) return 1;

 d[t]=c[k+1-t]= p[k+1+t]=0; //lui

 }

 return false;

}

int main()

{

 int n,x[100],c[200],p[200],d[100];

 scanf("%d",&n);

 for(int i=0;i<200;i++) d[i]=c[i]=p[i]=0;

 TRY(x,0,n,d,c+100,p);

 getch();

}

Bài toán 2: phân tích số

Nhập n phân tích tổng các số nguyên dương thành dãy tang dần

S = 1 + 1 +1 +1+1

= 1+ 1 +1 + 2

= 2 + 3 = 5

+ Ý tưởng: x1,x2,………xk,xk+1

Đặt T = x1,x2,………xk

xk+1 = n – t

* T +t + u = n (với u<= t)
* T + t <= n – t
* xk<= t<=n-T –t

Code minh họa:

#include<stdio.h>

#include<conio.h>

#define nl printf("\n")

void Print(int \*x,int k,int n){

 printf("%d = ",n);

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

 printf("%d",x[k]);

 nl;

}

void TRY(int \*x,int k,int T,int n){

 for(int t =x[k];t<= (n -T)/2;t++){

 x[k+1] = t;

 TRY(x,k+1,T+t,n);

 }

 x[k+1]= n -T;

 Print(x,k+1,n);

}

int main(){

 int n,x[100];

 scanf("%d",&n);

 x[0]=1;

 TRY(x,0,0,n);

}

Bài toán 3: Leo thang

Cho n bậc thang , một người bước 1 bước , 2 bước hoặc 3 bước. Hỏi số cách leo n bậc thang

Code minh họa:

#include<stdio.h>

#include<conio.h>

#define nl printf("\n")

int dem=0;

void Print(int \*x,int k,int n)

{

 dem++;

 printf("%d = ",n);

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

 printf("%d",x[k]);

 nl;

}

void TRY(int \*x,int k,int T,int n,int m)

{

 if(k==m-1) {x[k+1]=n-T;Print(x,k+1,n); return;}

 for(int t=0; T+t<=n;t++)

 {

 x[k+1]=t;

 TRY(x,k+1,T+t,n,m);

 }

}

int main()

{

 int n,x[100],m;

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

 TRY(x,0,0,n,m);

 printf("\n\nCo ta ca %d nghiem",dem);

 getch();

}

*Bài toán Thêm: Tìm đường đi từ f -> S*

*Cho đồ thị G = (V,E) . s,f V . Tìm đường đi từ f -> s*

*Phân tích: x1 -> x2 ->………-> xn*

*Sao cho 1. xi V*

 *2.x1 = S*

 *3.xn = f*

 *4. xi  xj*

*5.( xi, xi+1)*

*1.Tìm chu trình đi từ 1 đỉnh tới 1 đỉnh*

*2.Nhập S. tìm chu trình từ f -> S*

*3.Tìm chu trình hanilton qua tất cả các đỉnh 1 lần*

*4.Tìm đường đi ngắn nhất bằng quay lui*

Bài toán 4: Tìm đường đi trong mê cung.Cho mê cung là 1 ma trận 0 -1

1. -Vật cản

0 – Đường đi

VD: 7 8

 1 0 1 1 0 1 0 1

 0 0 0 0 0 0 0 0

 1 1 0 1 1 0 1 0

 0 1 0 0 0 0 0 0

 1 1 1 0 1 1 0 1

 1 0 0 0 0 0 0 1

 1 1 0 1 1 1 1 1

Code minh họa:

#include<stdio.h>

#include<conio.h>

#define FOR(i,a,b) for(int i=a;i<=b;i++)

#define nl printf("\n")

#define M(v) M[v.x][v.y]

struct Toado

{

 int x,y;

};

class MC

{

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

 public:

 void LoadFile(char \*);

 void Output();

 void TRY(Toado \*X,int k,Toado f);

 void Process();

};

void MC::Process()

{

 Toado f,s,X[100000];

 printf("Nhap vao xuat phat : "); scanf("%d%d",&s.x,&s.y);

 printf("Nhap vao ket thuc : "); scanf("%d%d",&f.x,&f.y);

 X[1]=s;

 M(X[1])=1;

 TRY(X,1,f);

 M(X[1])=0;

}

int hh[]={ 0, 0, 1,-1};

int hc[]={ 1,-1, 0, 0};

void MC::TRY(Toado \*X,int k,Toado f)

{

 if(X[k].x==f.x && X[k].y==f.y)

 {

 FOR(i,1,k-1) printf("(%d,%d)->",X[i].x,X[i].y);

 printf("(%d,%d)\n",X[k].x,X[k].y);

 return;

 }

 FOR(t,0,3)

 {

 Toado v=X[k];

 v.x+=hh[t];

 v.y+=hc[t];

 if(M(v)==0) //M[v.x][v.y]

 {

 X[k+1]=v;

 M(v)=1;

 TRY(X,k+1,f);

 M(v)=0; //lui

 }

 }

}

void MC::Output()

{

 FOR(i,0,n+1)

 {

 FOR(j,0,m+1) printf("%d ",M[i][j]);

 nl;

 }

}

void MC::LoadFile(char \*fn)

{

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

 fscanf(f,"%d%d",&n,&m);

 FOR(i,1,n)

 FOR(j,1,m) fscanf(f,"%d",&M[i][j]);

 FOR(i,0,n+1) M[i][0]=M[i][m+1]=1;

 FOR(j,0,m+1) M[0][j]=M[n+1][j]=1;

}

int main()

{

 MC G;

 G.LoadFile("mc.txt"); // Ma trận mc.txt đăth trong file .txt

 G.Output();

 G.Process();

 getch();

}

**Chương 4: Thiết kế tham lam**

**1.Phương pháp tham lam**

-Thuật toán tham lam giải các bài tập có yếu tố tối ưu

-Dùng kinh nghiệm để so sánh,đánh giá nhằm tìm lời giải

VD1: Cho một hình chữnhật kích thước cạnh nguyên dương n\*m .Cắt thành các hình vuông cạnh nguyên sao cho số hình vuông là ít nhất

Code :

Int CHV(int n,int m)

{

If(n==0 || m==0) return 0 ;

I(n>m) return n/m+CHV(n/m,m) ;

 Else return m/n+CHV(n,m/n);

}

VD2: Bài toán đổi tiền

Cho n giá trị a1,a2,a3…..an

Mã hiện có vô hạn tờ

Đổi M tiền .Tìm cách đổi sao cho số tờ ít nhất.

*Ví dụ:* 1 3 4 11

Đổi 19 =11+4+2=> 3 tờ

***BTVN:*** *Quay lui bài tiền :*

 *x1\*a1+x2\*a2+….+xn\*an =M*

*Giải phương trình sao cho tổng tiền min*

**2.Tham lam theo nguyên lý thứ tự**

**2.1 Bài toán sắp xếp balo**

Bài toán: Cho n đồ vật có kích thước và giá trị lần lượt :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Giá trị | A1 | A2 | …. | An |
| Kích thước | B1 | B2 | …. | Bn |

Balo có kích thước là m.Tìm cách để đồ vật vào balo sao cho kích thước <=M,Tổng giá trị lớn nhất.

VD:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Giá trị | 70 | 80 | 90 | 40 | 40 | 50 |
| Kích thước | 30 | 40 | 80 | 20 | 25 | 30 |

M=100

+Bước 1: Sắp xếp theo giảm tỉ lệ giá trị /kích thước

Trong trường hợp tỷ lệ = nhau sẽ ưu tiên lấy cái to trước ,bé sau

+Bước 2: Lấy lần lượt nếu bỏ vừa balo thì lấy

Code:

//Tham lam balo

#include<iostream>

#include<algorithm>

using namespace std;

struct DoVat{

 int kt,gt,id;

 float tl;

};

void Nhap(int &n,DoVat \*D){

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

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

 D[i].id=i;

 D[i].tl=D[i].gt/(float)D[i].kt;

 }

}

void Xuat(int n,DoVat \*D){

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

 cout<<"\nVat "<<D[i].id<<" co gia tri :"<<D[i].gt<<" va kich thuoc :"<<D[i].kt<<" co ti le : "<<D[i].tl;

}

bool ss(DoVat u,DoVat v){

 return u.tl>v.tl;

}

void Balo(int n,DoVat \*D,int M){

 int k=0,G=0;

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

 if(k+D[i].kt<=M){

 cout<<"\nChon vat "<<D[i].id<<" co kich thuoc : "<<D[i].kt<<" co gia tri : "<<D[i].gt;

 k+=D[i].kt;

 G+=D[i].gt;

 }

 cout<<"\nTong kich thuoc "<<k;

}

int main(){

 int n,M;

 DoVat D[100];

 cout<<"\nNhap kich thuoc balo : ";

 cin>>M;

 cout<<"\nNhap so do vat n = ";

 cin>>n;

 Nhap(n,D);

 cout<<"\nTruoc khi sap xep : ";Xuat(n,D);

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

 cout<<"\nSau khi sap xep: ";Xuat(n,D);

 Balo(n,D,M);

}

**2.2 Bài toán đóng thùng**

Bài toán: Cho n đồ vật có kích thước a1,a2…an

Thùng có kích thước M

Đóng thùng các đồ vật vào thùng sao cho số thùng ít nhất

Ví dụ:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Id | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| Kích thước | 7 | 4 | 2 | 8 | 1 | 6 | 8 | 2 | 4 | 5 | 3 |

M=10

+Bước 1: Sắp xếp theo kích thước giảm dần :

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Id | 4 | 7 | 1 | 6 | 10 | 2 | 9 | 11 | 3 | 8 | 5 |
| Kích thước | 8 | 8 | 7 | 6 | 5 | 4 | 4 | 3 | 2 | 2 | 1 |

+Bước 2: Đóng thùng

Code:

#include<stdio.h>

#include<conio.h>

#include<algorithm>

using namespace std;

struct dovat

{

 int kt,id;

};

void Nhap(int &n,dovat \*D)

{

 printf("Nhap so do vat n = "); scanf("%d",&n);

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

 {

 scanf("%d",&D[i].kt);

 D[i].id=i;

 }

}

void Xuat(int n,dovat \*D)

{

 for(int i=1;i<=n;i++) printf("%6d",D[i].id);

 printf("\n");

 for(int i=1;i<=n;i++) printf("%6d",D[i].kt);

}

bool ss(dovat u,dovat v) //u>v

{

 return u.kt>v.kt;

}

void Dongthung(int n,dovat \*D,int M)

{

 int d[100],T[100],k=0;

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

 {

 int p=0;

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

 if(T[j]+D[i].kt<=M)

 {

 if(p==0) p=j;

 else if(T[p]<T[j])p=j;

 }

 if(p==0){k++; T[k]=D[i].kt;d[D[i].id]=k;}

 else

 {

 T[p]+=D[i].kt;

 d[D[i].id]=p;

 }

 }

 printf("\nSo thung it nhat la %d\n",k);

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

 {

 printf("\nThung %d chua %d gom : ",j,T[j]);

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

 if(d[i]==j) printf("%d ",i);

 }

}

int main()

{

 int n,M;

 dovat D[100];

 Nhap(n,D);

 printf("\nTruoc khi sap xep :\n");

 Xuat(n,D);

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

 printf("\nSau khi sap xep :\n");

 Xuat(n,D);

 printf("\nNhap vao kich thuoc thung M = ");

 scanf("%d",&M);

 Dongthung(n,D,M);

 getch();

}

**2.3 Phân công công việc**

Bài tập: Có n việc mỗi việc chỉ có một người làm và mất thời gian a1,a2,3….an. Có m nguời.Hãy tính cách hoàn thành tất cả các công việc trong thời gian sớm nhất.

VD:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Id | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| Time | 4 | 7 | 2 | 8 | 4 | 8 | 3 | 2 | 4 | 9 | 3 | 6 |

Ý tưởng tham lam:

+Bước 1: Sắp xếp các công việc theo thời gian giảm dần

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Id | 10 | 4 | 6 | 2 | 12 | 1 | 5 | 9 | 7 | 11 | 3 | 8 |
| Time | 9 | 8 | 8 | 7 | 6 | 4 | 4 | 4 | 3 | 3 | 2 | 2 |

+Bước 2: Phân công công việc

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Phân công công việc | Phân công | T1 | T2 | T3 |
|  |  | 0 | 0 | 0 |
| Xét 10 9 | 1 | 9 | 0 | 0 |
| Xét 4 8 | 2 | 9 | 8 | 0 |
| Xét 6 8 | 3 | 9 | 8 | 8 |
| Xét 2 7 | 2 | 9 | 15 | 8 |
| Xét 12 6 | 3 | 9 | 15 | 14 |
| Xét 1 4 | 1 | 13 | 15 | 14 |
| Xét 5 4 | 1 | 17 | 15 | 14 |
| Xét 9 4 | 3 | 17 | 15 | 18 |
| Xét 7 3 | 2 | 17 | 18 | 18 |
| Xét 11 3 | 1 | 20 | 18 | 18 |
| Xét 3 2 | 2 | 20 | 20 | 18 |
| Xét 8 2 | 3 | 20 | 20 | 20 |

Kết luận:

* Hoàn thành 20 đơn vị thời gian
* Phân công:

Người 1 : 1,5,10,11( Tổng time: 20)

Người 2: 2,3,4,7(Tổng time: 20)

Người 3: 3,6,8,9,12(Tổng time:20)

*Code:*

#include<iostream>

#include<algorithm>

using namespace std;

struct Viec{

 int id,tg;

};

class PCCV{

 int n,m;

 Viec \*A;

 public:

 void Nhap();

 void PhanCong();

};

bool sosanh(Viec u,Viec v){

 return u.tg>v.tg;

}

void PCCV::Nhap(){

 cout<<"So cong viec : ";cin>>n;

 A= new Viec[n+5];

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

 A[i].id =i;

 cout<<"\nThoi gian A["<<i<<"] =";

 cin>>A[i].tg;

 }

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

 cout<<"\nSo nguoi: ";cin>>m;

}

void PCCV::PhanCong(){

 int \*t = new int [m+5];

 int \*p= new int [n+5];

 for(int i=1;i<=m;i++) t[i]=0;

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

 int k=1;

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

 if(t[k]>t[j]) k=j;

 t[k]+=A[i].tg;

 p[A[i].id] =k;

 }

 int max=t[1];

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

 if(max<t[i]) max = t[i];

 cout<<"\nThoi gian hoan thanh viec som nhat la : "<<max;

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

 cout<<"\nThoi gian hoan thanh xong viec thu "<<j<<" la "<<t[j]<<" gom: ";

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

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

 }

}

int main(){

 PCCV p;

 p.Nhap();

 p.PhanCong();

}

**2.4 Lập lịch**

Bài tập: Có n sự kiện có thời gian bắt đầu, thời gian kết thúc lần lượt là :

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Sự kiện | 1 | 2 | 3 | … | … | n |
| Bắt đầu | A1 | A2 | A3 | … | … | An |
| Kết thúc | B1 | B2 | B3 | … | … | Bn |

Có một hội trường.Hãy tổ chức sự kiện sao cho số sự kiện được tổ chức là nhiều nhất.

Cách 1: Tham lam Bi-Ai

Cách 2: Tham lam theo thời gian bắt đầu

Cách 3:Sự kiện ít xung đột

* Cách 1,2,3 bị loại

Cách 4:Tham lam theo thời gian kết thúc sớm

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Sự kiện | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| Time Bắt đầu | 4 | 12 | 11 | 3 | 2 | 13 | 4 | 2 | 17 | 15 | 7 | 20 | 18 | 8 |
| Time Kết thúc | 7 | 18 | 16 | 9 | 7 | 14 | 5 | 10 | 19 | 20 | 18 | 24 | 19 | 11 |

+Bước 1: Sắp xếp theo thời gian kết thúc sớm

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Sự kiện | 7 | 1 | 5 | 4 | 8 | 14 | 6 | 3 | 2 | 11 | 9 | 13 | 10 | 12 |
| Time Bắt đầu | 4 | 4 | 2 | 3 | 2 | 8 | 13 | 11 | 12 | 7 | 17 | 18 | 15 | 20 |
| Time Kết thúc | 5 | 7 | 7 | 9 | 10 | 11 | 14 | 16 | 18 | 18 | 19 | 19 | 20 | 24 |

+Bước 2: Tham lam

Time Bắt đầu(Timebd),Time Kết thúc(Timekt)

Mốc time ban đầu : M= 4(Time bắt đầu của sự kiện 7)

7 4 5 ->**Chọn** (vì Time bắt đầu sk 7 =4) M=5

1 4 7 -> Loại(vì 1 <4) M=5

5 2 7 ->Loại(vì 2<4) M=5

4 3 9 ->Loại(3<5) M=5

8 2 10->Loại(2<5) M=5

14 8 11->**Chọn**(vì 8>5) M=11

6 13 14 ->**Chọn** (vì 13>11) M=14

3 11 16 ->Loại(11<14) M=14

2 12 18 ->Loại(12<14) M=14

11 7 18->Loại(7<14) M=14

9 17 19->**Chọn**(17>14) M=19

13 18 19 ->Loại(18<19) M=19

10 15 20 ->Loại(15<19) M=19

12 20 24 ->**Chọn**(20>19) M=24

*Code:*

//lap lich hoi truong

#include<iostream>

#include<algorithm>

using namespace std;

struct SuKien{

 int id,timeS,timeF;

};

class HoiTruong{

 int n;

 SuKien \*S;

 public:

 void Nhap();

 void LapLich();

};

bool ss(SuKien u,SuKien v){

 return u.timeF<v.timeF;

}

void HoiTruong::Nhap(){

 cout<<"\nSo su kien: ";

 cin>>n;

 S=new SuKien[n+5];

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

 S[i].id=i;

 cout<<"\nThoi gian bat dau S["<<i<<"]: ";cin>>S[i].timeS;

 cout<<"\nThoi gian ket thuc S["<<i<<"]: ";cin>>S[i].timeF;

 }

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

 for(int i=1;i<=n;i++) cout<<S[i].id;

}

void HoiTruong::LapLich(){

 int k=0,t=S[1].timeS;

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

 if(S[i].timeS>=t){

 cout<<"\nChon su kien "<<S[i].id<<" co time bd: "<<S[i].timeS<<" time kt: "<<S[i].timeF;

 k++;

 t=S[i].timeF;

 }

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

}

int main(){

 HoiTruong H;

 H.Nhap();

 H.LapLich();

}

**3.Tham lam trên đồ thị**

**3.1 Tô màu đồ thị**

Bài tập : Cho đồ thị G = (V,E) vô hướng,không trọng số.Hãy tìm cách tô màu các đỉnh của đồ thị sao cho hai đỉnh kề nhau có màu khác nhau,sao cho số màu là ít nhất(Sắc số của đồ thị)

Tạo 1 ma trận :

*Code:*

#include<iostream>

#include<algorithm>

using namespace std;

struct Dinh{

 int id,bac;

};

class Graph{

 int n,A[100][100];

 Dinh D[100];

 public :

 void Nhap();

 void TinhBac();

 void ToMau();

};

void Graph::Nhap(){

 cin>>n;

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

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

 cin>>A[i][j];

// TinhBac();

}

bool ss(Dinh u,Dinh v){

 return u.bac > v.bac;

}

void Graph::TinhBac(){

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

 D[i].id = i;

 D[i].bac=0;

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

 D[i].bac +=A[i][j];

 }

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

}

void Graph::ToMau(){

 int To[100],k=0;

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

 TinhBac();

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

 if(To[D[i].id]==0)

 {

 k++;

 To[D[i].id]=k;

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

 if(To[D[j].id]==0){

 int ok=1;

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

 if(To[D[u].id]==k && A[D[j].id][D[u].id]==1){

 ok=0;

 break;

 }

 if(ok) To[D[j].id]=k;

 }

 }

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

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

 cout<<"\nDinh "<<i<<"co: "<<To[i];

}

int main(){

 Graph p;

 freopen("tomau.txt","r",stdin);

 p.Nhap();

 p.ToMau();

}

**3.2 Thuật toán Kruskal tìm cây khung min**

Bài tập: Cho đồ thị G=(V,E) vô hướng,liên thông có trọng số trên cạnh.Tìm cây khung min

 9 9

 2 2 8 7 8

 2 7

 4 1

 3 6 1

 5 1

Ý tưởng: Tham lam theo cạnh

+Bước 1 : Sắp xếp các cạnh theo trọng số tang dần

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Cạnh  | 68 | 69 | 89 | 14 | 15 | 45 | 47 | 48 | 78 | 58 | 35 | 56 | 25 | 36 | 12 | 23 |
| Trọng số | 1 | 1 | 1 | 2 | 2 | 2 | 3 | 4 | 5 | 6 | 7 | 7 | 8 | 8 | 9 | 9 |

+Bước 2: Lấy dần các cạnh sao cho không tạo thành chu trình

Mảng d (Mảng đánh dấu kiểm tra tạo thành chu trình hay chưa,nếu tạo thành chu trình đánh dấu x)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Đỉnh | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| d | -1 | -2 | -3 | -4 | -5 | -6 | -7 | -8 | -9 |
| Xét 68 | -1 | -2 | -3 | -4 | -5 | 1 | -7 | 1 | -9 |
| Xét 69 | -1 | -1 | -3 | -4 | -5 | 1 | -7 | 1 | 1 |
| Xét 89 | x | x | x | x | x | x | x | x | x |
| Xét 14 | 2 | -2 | -3 | 2 | -5 | 1 | -7 | 1 | 1 |
| Xét 15 | 2 | -2 | -3 | 2 | 2 | 1 | -7 | 1 | 1 |
| Xét 45 | x | x | x | x | x | x | x | x | x |
| Xét 47 | 2 | -2 | -3 | 2 | 2 | 1 | 2 | 1 | 1 |
| Xét 48 | 2 | -2 | -3 | 2 | 2 | 2 | 2 | 2 | 2 |
| Xét 78 | x | x | x | x | x | x | x | x | x |
| Xét 58 | x | x | x | x | x | x | x | x | x |
| Xét 35 | 2 | -2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| Xét 56 | x | x | x | x | x | x | x | x | x |
| Xét 25 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |

Dừng !!!! Đã tìm thấy cây khung min.

1-4,1-5,2-5,3-5,4-7,4-8,6-8,6-9

Code:

#include<iostream>

#include<algorithm>

using namespace std;

struct Canh{

 int D,C,T;//d: dau,c: cuoi,t: trong so

};

class Graph{

 int n,m; //m: so canh,n: so dinh

 Canh \*E;

 public :

 void Nhap();

 void Kruskal();

};

bool ss(Canh u,Canh v){

 return u.T <v.T;

}

void Graph::Nhap(){

 cin>>n>>m;

 E = new Canh[m+5];

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

 cin>>E[i].D>>E[i].C>>E[i].T;

 sort(E+1,E+m+1,ss);

}

void Graph::Kruskal(){

 int \*d = new int [n+5];

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

 int k=0,TS=0;

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

 if(d[E[j].D]!=d[E[j].C]){

cout<<"\nChon canh : "<<E[j].D<<"-"<<E[j].C<<" ts: "<<E[j].T;

TS+=E[j].T;

if(d[E[j].D]<0 && d[E[j].C]<0)

d[E[j].D] =d[E[j].C]= ++k;

else if(d[E[j].D]>0 && d[E[j].C]>0){

int u=d[E[j].C];

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

if(d[i]==u)

d[i]=d[E[i].D];

}

else {

if(d[E[j].D]<0)

d[E[j].D]=d[E[j].C];

 else d[E[j].C]=d[E[j].D];

}

}

cout<<"\nTrong so cua cay khung la "<<TS;

}

int main(){

 Graph G;

 freopen("CayKhung.txt","r",stdin);

 G.Nhap();

 G.Kruskal();

}

**3.3 Thuật toán Prim tìm cây khung min**

Bài tập: Cho đồ thị G=(V,E) vô hướng,liên thông có trọng số trên cạnh.Tìm cây khung min

 9 9

 2 2 8 7 8

 2 7

 4 1

 3 6 1

 5 1

Ý tưởng: Tham lam theo đỉnh

Sử dụng Queue

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 0 |  |  |  |  |  |  |  |  |
| - | (9,1) |  | (2,1) | (2,1) |  |  |  |  |
| - | (9,1) |  | - | (2,1) |  | (3,4) | (4,4) |  |
| - | (8,5) | (7,5) | - | - | (7,5) | (3,4) | (4,4) |  |
| - | (8,5) | (7,5) | - | - | (7,5) | - | (4,4) |  |
| - | (8,5) | (7,5) | - | - | (7,3) | - | - | (1,8) |
| - | (8,5) | (7,5) | - | - | (1,9) | - | - | - |
| - | (8,5) | (7,5) | - | - | - | - | - | - |
| - | (8,5) | - | - | - | - | - | - | - |

Minh họa:

 9

 2=>Chọn

 3

 4

 7

 8

 7

 3=>Chọn

 4

Tương tự…

Code:

#include<iostream>

using namespace std;

#define FOR(i,a,b) for(int i=a;i<=b;i++)

class Graph{

 int n,A[100][100];

 public :

 void Nhap();

 void Prim();

};

void Graph::Nhap(){

 cin>>n;

 FOR(i,1,n)

 FOR(j,1,n)

 cin>>A[i][j];

}

int Min(int \*Q,int \*L,int n) //tim min trong nhung cai duoc xep hang

{

 int p=0;

 FOR(i,1,n)

 if(Q[i]==1){

 if(p==0) p=i;

 if(L[p]>L[i]) p=i;

 }

 return p;

}

void Graph::Prim(){

 int L[100],Q[100],P[100],u;

 FOR(i,1,n) {

 Q[i]=0;L[i]=-1;

 }

 L[1]=0;Q[1]=1;

 while(u=Min(Q,L,n)){

 Q[u]=-1;

 FOR(i,1,n)

 if(A[u][i]>0 && Q[i]!=-1){ //xét nh?ng cai lay chua dk lay ra khoi hang hoac chua duoc xep

 if(L[i]==-1 || L[i]>=A[u][i]){

 L[i]=A[u][i];

 P[i]=u;

 Q[i]=1;

 }

 }

 }

 int T=0;

 FOR(i,1,n) T+=L[i];

 cout<<"\nTrong so cua cay: "<<T;

 FOR(i,2,n) cout<<"\nChon canh "<<i<<" - "<<P[i]<<" ts: "<<L[i];

}

int main(){

 Graph G;

 freopen("CayKhungPrim.txt","r",stdin);

 G.Nhap();

 G.Prim();

}

**3.4 Thuật toán Djikstra tìm cây khung min**

VD: Cho đồ thị G=(V,E) có trọng số dương.Tìm đường đi min từ 1 điểm tới các đỉnh còn lại

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 0 |  |  |  |  |  |  |  | ∞ |
| - | (9,1) |  | (2,1) | (2,1) |  |  |  |  |
| - | (9,1) |  | - | (2,1) |  | (5,4) | (6,4) |  |
| - | (9,1) | (9,5) | - | - | (9,5) | (5,4) | (6,4) |  |
| - | (9,1) | (9,5) | - | - | (9,5) | - | (6,4) |  |
| - | (9,1) | (9,5) | - | - | (7,5) | - | - | (7,8) |
| - | (9,1) | (9,5) | - | - | - | - | - | (7,8) |
| - | (9,1) | (9,5) | - | - | - | - | - | - |
| - | (9,1) | - | - | - | - | - | - | - |

#include<iostream>

using namespace std;

#define FOR(i,a,b) for(int i=a;i<=b;i++)

class Graph{

int n,A[100][100];

public :

void Nhap();

void Prim();

};

void Graph::Nhap(){

cin>>n;

FOR(i,1,n)

FOR(j,1,n)

cin>>A[i][j];

}

int Min(int \*Q,int \*L,int n) //tim min trong nhung cai duoc xep hang

{

 int p=0;

 FOR(i,1,n)

 if(Q[i]==1){

if(p==0) p=i;

if(L[p]>L[i]) p=i;

 }

 return p;

}

void Graph::Prim(){

int L[100],Q[100],P[100],u;

FOR(i,1,n) {

Q[i]=0;L[i]=-1;

}

L[1]=0;Q[1]=1;

while(u=Min(Q,L,n)){

Q[u]=-1;

FOR(i,1,n)

if(A[u][i]>0 && Q[i]!=-1){ //xét nh?ng cai lay chua dk lay ra khoi hang hoac chua duoc xep

if(L[i]==-1 || L[i]>=A[u][i]){

L[i]=A[u][i];

P[i]=u;

Q[i]=1;}

 }

 }

int T=0;

FOR(i,1,n) T+=L[i];

cout<<"\nTrong so cua cay: "<<T;

FOR(i,2,n) cout<<"\nChon canh "<<i<<" - "<<P[i]<<" ts: "<<L[i];

}

int main(){

 Graph G;

 freopen("CayKhungPrim.txt","r",stdin);

 G.Nhap();

 G.Prim();

}

File txt :

9

0 9 -1 2 2 -1 -1 -1 -1

9 0 9 -1 8 -1 -1 -1 -1

-1 9 0 -1 7 8 -1 -1 -1

2 -1 -1 0 2 -1 3 4 -1

2 8 7 2 0 7 -1 6 -1

-1 -1 8 -1 7 0 -1 1 1

-1 -1 -1 3 -1 -1 0 5 -1

-1 -1 -1 4 6 1 5 0 1

-1 -1 -1 -1 -1 1 -1 1 0

**Chương 5: Quy hoạch động(Dynamic programming)**

**1.Phương phát quy hoạch động**

-Phương pháp: Là phương pháp thiết kế thuật toán theo kiểu dưới lên(Bottom-up)

-Để giải bài toán lớn,giải các bài toán con quy hoạch dần dần,giải các bài toán lớn hơn dựa vào bài toán nhỏ hơn.

VD1: Bài toán lát gạch

Cho sân 2\*n

Gạch 2\*1

Có bao nhiêu cách lát sân ?

Ý tưởng : Đặt - số cách lát sân 2\*n

Ta có

=1

=2

=3

…..

=

Code: C[1]=1,C[2]=2;

 for(int i=3;i<=n;i++)

C[i]=C[i-1]+C[i-2];

Cout<<C[i];

VD2: Bài toán leo thang

Có n bậc thang.Một người 1 bước lên 1,2,3 bước.Hỏi số cách lên được n bậc?

Ý tưởng: Đặt - số cách leo thang n bậc

=1

=2

=4

=7

…

=++

VD3: Có bao nhiêu xâu nhị phân độ dài n mà không chứa 101

Ý tưởng: (Quy Hoạch Động )

–số xâu nhị phân độ dài n không chứa 101

=0

=2

=4

7

=12

Đặt A[n] là xâu nhị phân độ dài n không chứa 101 nhưng kết thúc ở 0

B[n] ] là xâu nhị phân độ dài n không chứa 101 nhưng kết thúc ở 1

Ta có :

=

=

=

=>=

VD4: Cho a1,a2,a3…an

Tìm dãy con liên tục có tổng max 4 -7 2 -8 1 2 -4 5 -6 2

Ý tưởng:

Đặt C[i] – Tổng max kết thúc bởi i

C[i]=Max()

= =4

= Max(,+)

= Max(+++,)

= Max(,+)

Code:

int main(){

int n,\*a,C,Max;

cin>>n;

a=new int [n+1];

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

Max=c=a[1];

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

C=C>0?c+a[i]:a[i];

if(Max<c) Max=c;

}

cout<<Max;

}

**2.Một số bài toán Quy Hoạch Động**

**2.1 Bài toán đổi tiền**

BT: Cho n mệnh giá tiền A1,A2,…An

Muốn đổi số tiền M sao cho số tờ tiền ít nhất

VD:

Mệnh giá 4 7 3 5 2 6 1

M=20

Đặt là số cách ít nhất khi đổi j tiền sử dụng các mệnh giá a1…an

= 0

=

=

Minh họa:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| C | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| \* | 0 | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ |
| 4 | 0 | ∞ | ∞ | ∞ | 1 | ∞ | ∞ | ∞ | 2 | ∞ | ∞ | ∞ | 3 | ∞ | ∞ | ∞ | 4 | ∞ | ∞ | ∞ | 5 |
| 7 | 0 | ∞ | ∞ | ∞ | 1 | ∞ | ∞ | 1 | 2 | ∞ | ∞ | 2 | 3 | ∞ | 2 | 3 | 4 | ∞ | 3 | 4 | 5 |
| 3 | 0 | ∞ | ∞ | 1 | 1 | ∞ | 2 | 1 | 2 | 3 | 2 | 2 | 3 | 3 | 2 | 3 | 4 | 3 | 3 | 4 | 4 |
| 5 | 0 | ∞ | ∞ | 1 | 1 | 1 | 2 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 3 | 3 | 3 | 3 | 3 | 4 |
| 2 | 0 | ∞ | 1 | 1 | 1 | 1 | 2 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 3 | 3 | 3 | 3 | 3 | 4 |
| 6 | 0 | ∞ | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 3 | 3 | 3 | 3 | 3 | 3 |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 3 | 3 | 3 | 3 | 3 | 3 |

* Số tờ tiền ít nhất là 3
* 20 = 2\*7 +6

Code:

#include<stdio.h>

#include<conio.h>

#include<iostream>

using namespace std;

class Money

{

 int n,\*a,\*\*C,M;

 public:

 void nhap();

 void BuocThuan();

 void BuocNghich(int n,int M);

 void xuly();

};

void Money::nhap()

{

 printf("Nhap so menh gia n = "); scanf("%d",&n);

 a=new int[n+5];

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

 {

 printf("Menh gia a[%d] = ",i);scanf("%d",&a[i]);

 }

 printf("Nhap so tien muon doi M = "); scanf("%d",&M);

}

void Money::BuocThuan()

{

 C=new int \*[n+5];

 for(int i=0;i<=n;i++) C[i]=new int [M+5];

 for(int j=1;j<=M;j++) C[0][j]=M+1;

 C[0][0]=0;

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

 for(int j=0;j<=M;j++)

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

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

}

void Money::BuocNghich(int n,int M)

{

 if(C[n][M]==0) return ;

 while(C[n][M]==C[n-1][M]) n--;

 BuocNghich(n,M-a[n]);

 printf("%d+",a[n]);

}

void Money::xuly()

{

 nhap();

 BuocThuan();

 if(C[n][M]==M+1) printf("\nKhong doi duoc tien ");

 else

 {

 printf("\nSo to tien it nhat %d",C[n][M]);

 printf("\n%d = ",M);

 BuocNghich(n,M);

 printf("\b ");

 }

}

int main()

{

 Money M;

 M.xuly();

 getch();

}

**2.2 Bài toán sắp xếp balo không lặp**

BT: Cho n đồ vật có kích thước và giá trị

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| KT | A1 | A2 | .. | .. | An |
| GT | B1 | B2 | .. | .. | Bn |

 có kích thước M tìm cách cho các vật vào balo sao cho tsổng giá trị max

VD :

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ID | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| KT | 3 | 2 | 4 | 5 | 2 | 3 | 4 |
| GT | 7 | 6 | 8 | 9 | 4 | 5 | 7 |

Ý tưởng quy hoạch động :

Đặt là giá trị lớn nhất khi xếp đồ vật từ 1-> I vào balo j

Minh họa:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | C | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| gt | kt | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 7 | 3 | 0 | 0 | 0 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 |
| 6 | 2 | 0 | 0 | 6 | 7 | 7 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 |
| 8 | 4 | 0 | 0 | 6 | 7 | 8 | 13 | 14 | 15 | 15 | 21 | 21 | 21 | 21 | 21 | 21 | 21 | 21 |
| 9 | 5 | 0 | 0 | 6 | 7 | 8 | 13 | 14 | 15 | 16 | 21 | 22 | 23 | 24 | 24 | 30 | 30 | 30 |
| 4 | 2 | 0 | 0 | 6 | 7 | 10 | 13 | 14 | 17 | 18 | 21 | 22 | 25 | 26 | 27 | 30 | 30 | 34 |
| 5 | 3 | 0 | 0 | 6 | 7 | 10 | 13 | 14 | 17 | 18 | 21 | 22 | 25 | 26 | 27 | 30 | 31 | 34 |
| 7 | 4 | 0 | 0 | 6 | 7 | 10 | 13 | 14 | 17 | 18 | 21 | 22 | 25 | 26 | 28 | 30 | 32 | 34 |

* Tổng giá trị max : 34

**2.3 Bài toán cắt hình vuông**

Bài tập: Cho hình chữ nhật cỡ n\*m . Hãy cắt thành cách hình vuông cạnh nguyên sao cho số hình là ít nhất

Ý tưởng: Quy hoạch động

Đặt là số hình vuông ít nhất khi cắt hình vuông i\*j

Ta có :

=Min()

Minh họa: n=9,m=10

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| C | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 2 | 2 | 1 | 3 | 2 | 4 | 3 | 5 | 4 | 6 | 5 |
| 3 | 3 | 3 | 1 | 4 | 4 | 2 | 5 | 5 | 3 | 6 |
| 4 | 4 | 2 | 4 | 1 | 5 | 3 | 5 | 2 | 6 | 4 |
| 5 | 5 | 4 | 4 | 5 | 1 | 5 | 5 | 5 | 6 | 2 |
| 6 | 6 | 3 | 2 | 3 | 5 | 2 | 5 | 4 | 3 | 4 |
| 7 | 7 | 5 | 5 | 5 | 5 | 5 | 1 | 7 | 6 | 6 |
| 8 | 8 | 4 | 5 | 2 | 5 | 4 | 7 | 1 | 7 | 5 |
| 9 | 9 | 6 | 3 | 6 | 6 | 3 | 6 | 7 | 1 | 6 |

* Số hình vuông ít nhất là 6
* Truy Vết

 9\*10

 4\*10 5\*10

4\*2 4\*8 5\*5 5\*5

2\*2 2\*2 4\*4 4\*4

Code :

#include<stdio.h>

#include<string>

#include<conio.h>

#include<iostream>

int C[105][105];

using namespace std;

void BuocNghich(string x,int n,int m)

{

 cout<<x<<n<<"\*"<<m<<endl;

 if(C[n][m]>1)

 {

 for(int k=1;k<=n/2;k++)

 if(C[n][m]==C[k][m]+C[n-k][m])

 {

 BuocNghich(x+" ",k,m);

 BuocNghich(x+" ",n-k,m);

 return;

 }

 for(int k=1;k<=m/2;k++)

 if(C[n][m]==C[n][k]+C[n][m-k])

 {

 BuocNghich(x+" ",n,k);

 BuocNghich(x+" ",n,m-k);

 return;

 }

 }

}

int main()

{

 int n,m;

 printf("\nNhap n va m : "); scanf("%d%d",&n,&m);

 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];

 }

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

 string x="";

 BuocNghich(x,n,m);

 getch();

}

**2.4 Xâu con chung dài nhất**

Bài tập: Cho một xâu kí tự . Một xâu con được lấy bằng cách lấy các kí tự xâu đó giữ nguyên thứ tự :

X= ACBACX

Cho 2 xâu x,y ,tìm ra phần xâu con chung dài nhất x và y

 x= AXTGAXXTAG

 y= GTAXGAXTTAX

* AXGAXTA

Ý tưởng quy hoạch động: Đặt là độ dài xâu con chung dài nhất của và

Ta có:

Minh họa:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| C | \* | G | T | A | X | G | A | X | T | T | A | X |
| \* | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| A | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| X | 0 | 0 | 0 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| T | 0 | 0 | 1 | 1 | 2 | 2 | 2 | 2 | 3 | 3 | 3 | 3 |
| G | 0 | 1 | 1 | 1 | 1 | 3 | 2 | 2 | 3 | 3 | 3 | 3 |
| A | 0 | 1 | 1 | 2 | 3 | 3 | 4 | 4 | 4 | 4 | 4 | 4 |
| X | 0 | 1 | 1 | 2 | 3 | 3 | 4 | 5 | 5 | 5 | 5 | 5 |
| X | 0 | 1 | 1 | 2 | 3 | 3 | 4 | 5 | 5 | 5 | 5 | 6 |
| T | 0 | 1 | 2 | 2 | 3 | 3 | 4 | 5 | 6 | 6 | 6 | 6 |
| A | 0 | 1 | 2 | 3 | 3 | 3 | 4 | 5 | 6 | 6 | 7 | 7 |
| G | 0 | 1 | 2 | 3 | 3 | 4 | 4 | 5 | 6 | 6 | 7 | 7 |

* AXGAXTA

Code:

#include<stdio.h>

#include<conio.h>

#include<string.h>

void BuocNghich(char \*x,int n,int m,int C[105][105])

{

 if(C[n][m]==0) return;

 while(C[n][m]==C[n-1][m]) n--;

 while(C[n][m]==C[n][m-1]) m--;

 BuocNghich(x,n-1,m-1,C);

 printf("%c",x[n]);

}

int main()

{

 int n,m,C[105][105];

 char x[105],y[105];

 printf("Nhap xau x : ");fflush(stdin); scanf("%s",x+1);

 n=strlen(x+1);

 printf("Nhap xau y : ");fflush(stdin); scanf("%s",y+1);

 m=strlen(y+1);

 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]= C[i-1][j-1]+1;

 else C[i][j]=C[i-1][j]>C[i][j-1]?C[i-1][j]:C[i][j-1];

 printf("\nDo dai xau con chung dai nhat la %d\n",C[n][m]);

 BuocNghich(x,n,m,C);

 getch();

}

* 1. **Tam giác số**

Bài toán:

 4

 7 2

 1 6 8

 3 4 2 3

 5 6 4 2 1

3 7 2 1 4 5

Tìm đường đi từ đỉnh xuống chân núi sao cho đường đi có tổng max

Ý tưởng quy hoạch động :

Đặt là giá trị max thu được khi đi từ A11->Aij

Ta có :

+…… =

= +

= Max(+ )

Minh họa :

A

4

7 2

1 6 8

3 4 2 3

5 6 4 2 1

3 7 2 1 4 5

C

4

11 6

12 17 14

15 21 19 17

20 27 25 21 18

23 **34** 29 26 25 23

 => Max = 34

Truy vết :

(1,1) -> (2,1) -> (3,2) -> (4,2) -> (5,2) -> (6,2)

Code:

#include<stdio.h>

#include<conio.h>

class TGS

{

 int n,C[105][105],A[105][105];

 public:

 void nhap();

 void BuocThuan();

 void BuocNghich(int n,int m);

};

void TGS::nhap()

{

 printf("Nhap n = "); scanf("%d",&n);

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

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

 {

 printf("A[%d][%d] = ",i,j);

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

 }

}

int Max(int a,int b){return a>b?a:b;}

void TGS::BuocThuan()

{

 C[1][1]=A[1][1];

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

 {

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

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

 }

 for(int i=3;i<=n;i++)

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

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

 int p=1;

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

 if(C[n][j]>C[n][p])p=j;

 printf("\n\nGia tri lon nhat la %d\n",C[n][p]);

 BuocNghich(n,p);

}

void TGS::BuocNghich(int n,int p)

{

 if(n>1)

 {

 if(p==1 || C[n-1][p]>C[n-1][p-1]) BuocNghich(n-1,p);

 else BuocNghich(n-1,p-1);

 }

 printf("(%d,%d)->",n,p);

}

int main()

{

 TGS T;

 T.nhap();

 T.BuocThuan();

 getch();

}