**Tema Greedy-8.noiembrie.2012**

1.Se citesc două numere naturale n şi m (n >= m), apoi se citeşte un şir de n numere întregi. Să se determine o submulţime de elemente din şir cu proprietatea că suma elementelor din mulţime este maxima.

#include<iostream>

using namespace std;

float a[100], b[100];

int i, m, n;

void gredy(){

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

 if(a[i]>=0){

 m++;

 b[m]=a[i];

 }

}

int main(){

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

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

 cin>>a[i];

 }

 gredy();

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

 cout<<b[i]<<" ";

 }

2.Se citeşte un şir de n numere întregi. Să se aleagă din şir un număr maxim de elemente astfel încât produsul lor să fie maxim. De exemplu, pentru n = 5 şi şirul 3, –4, 2, –5, –8 se vor alege 3, 2, –5, –8.

#include<iostream>

using namespace std;

int v[30],n,j=0,m, nr\_elment,v\_minus[39] ;

int ordonare()

{

 int i ,aux;

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

 {

 if (v[i]<0)

 { j++;

 v\_minus[j]=v[i];

 }

 }

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

 {

 if(v\_minus[i]>v\_minus[i+1])

 {

 aux=v\_minus[i];

 v\_minus[i]=v\_minus[i+1];

 v\_minus[i+1]=aux;

 }

 }

}

int greedy()

{

 int i ;

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

 if (v[i]>0)

 cout<<v[i]<<" ";

 if(j!=1)

 {

 if(j%2==0)

 {

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

 cout<<v\_minus[i]<<" ";

 }

 else

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

 cout<<v\_minus[i]<<" ";

 }

 }

}

int main()

{

 cin>>n;

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

 cin>>v[i];

 ordonare();

 greedy();

}

3.Se citesc n intervale de forma [ai, bi), i = 1..n, unde capetele intervalelor sunt numere naturale. Să se aleagă un număr maxim de intervale astfel încât oricare două intervale alese să nu se intersecteze. De exemplu, pentru n=6 şi intervalele [8, 10), [2, 20), [4, 8), [12, 16), [3, 18), [17, 18) se aleg intervalele [4, 8), [8, 10), [12, 16), [17, 18)

#include <iostream>

using namespace std;

int n,x[3][100],g,i;

void sortare(int g[3][100])

{

 int schimb,i,aux;

 do

 {

 schimb=0;

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

 if(g[1][i]>g[1][i+1])

 {

 aux=g[1][i];

 g[1][i]=g[1][i+1];

 g[1][i+1]=aux;

 aux=g[2][i];

 g[2][i]=g[2][i+1];

 g[2][i+1]=aux;

 schimb=1;

 }

 }while(schimb);

}

int main()

{

 cin>>n;

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

 {

 cin>>x[1][i];

 cin>>x[2][i];

 }

 sortare(x);

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

 {

 if(x[2][i]<=x[2][i+1])

 cout<<"["<<x[1][i]<<","<<x[2][i]<<") ";

 }

 if(x[2][n-1]<=x[2][n])

 cout<<"["<<x[1][n]<<","<<x[2][n]<<") ";

 return 0;

}

4.Se citeste de la tastatura un vector de n numere intregi distincte si un numar k. Sa se determine o submultime cu numar maxim de elemente a acestuia. In submultime trebuie sa fie cel mult k numere prime, a caror suma sa fie strict mai mare decat suma numerelor neprime selectate.

#include <iostream>

using namespace std;

int n,k,i,j,v[100],prime[100],a,neprim=0,sumprim=0;

void sortare(int g[100]){

 int schimb,i,aux;

 do

{

 schimb=0;

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

 if(g[i]>g[i+1])

 {

 aux=g[i];

 g[i]=g[i+1];

 g[i+1]=aux;

 schimb=1;

 }

 }while(schimb);

}

int prim(int n)

{

 int ok=0,i;

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

 {

 if(n%i==0)

 ok++;

 }

 if(ok==0||n==2)

 return 0;

 else

 return 1;

}

int main()

{

 cout<<"n=";

 cin>>n;

 cout<<"k=";

 cin>>k;

 a=1;

 i=1;

 do

 {

 cin>>v[i];

 if(prim(v[i])==0)

 {

 prime[a]=v[i];

 a++;

 }

 else

 neprim=neprim+v[i];

 i++;

 }while(i!=n+1);

 i=1;

 do

 {

 if(sumprim<=neprim)

 {

 cout<<prime[i]<<" ";

 sumprim=sumprim+prime[i];

 }

 i++;

 }while(i!=k+1);

 return 0;

}

5.Se dau n numere întregi nenule B={b1, b2, …, bn} si m numere întregi nenule a1, a2, …, am. (m<n). Să se determine o submulţime a mulţimii B care maximizează valoarea expresiei
E=a1\*x1 + a2\*x2 + … + am\*xm

#include<iostream>

using namespace std;

int a[30], b[30], m, n, i, e;

void sort(int k, int x[20]){

 int schimb, aux;

 do{

 schimb=0;

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

 if(x[i]>x[i+1]){

 aux=x[i];

 x[i]+x[i+1];

 x[i+1]=aux;

 schimb=1;

 }

 }while(schimb);

}

int main(){

 cout<<"m= "; cin>>m;

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

 cin>>a[i];

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

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

 cin>>b[i];

 sort(m, a);

 sort(n, b);

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

 e+=a[i]+b[n-m+1];

 cout<<"emax: "<<e;

}

6.Se consideră un număr nelimitat de rucsaci de acelaşi tip, cu fiecare putându-se transporta o greutate maximă G şi n<=1000 obiecte de greutăţi g1, g2, …, gn. Să se determine numărul minim de rucsaci necesari pentru a transporta toate obiectele.

#include<iostream>

#include<fstream>

using namespace std;

int a[100],n,G;

void sort(int g[100])

{

 int schimb,i,aux;

 do

 {

 schimb=0;

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

 if(g[i]>g[i+1])

 {

 aux=g[i];

 g[i]=g[i+1];

 g[i+1]=aux;

 schimb=1;

 }

 }while(schimb);

}

void citire()

{

int i;

ifstream f("date.in");

f>>n;

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

 f>>a[i];

f>>G;

}

int main()

{

int s=0,nr=1,i,j;

citire();

sort(a);

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

 if(a[i]>G)

 a[i]=0;

for(i=n;i>0;i--)

{

if(a[i]!=0)

 {

cout<<"Rucsacul "<<n-i+1<<" contine:";

 s=a[i];

 cout<<a[i]<<" ";

 a[i]=0;

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

{

if(a[j]!=0)

 {

if(s+a[j]<=G)

 {

s=s+a[j];

 cout<<a[j]<<" ";

 a[j]=0;

 }

 else if(s+a[j]>G)

 {

nr++;

 s=0;

 cout<<endl;

 j=i;

 }

 }

}

 }

}

cout<<endl<<endl<<nr<<" rucsaci";

}

7.Se dau două şiruri de lungime N şi un număr K (N<1000 şi K<1000). Cele două şiruri au doar elementele 1 şi –1. Scopul este de a transforma primul şir în al doilea. Singura operaţie permisă este de a selecta o secvenţă de K elemente alăturate şi să le inversăm semnul la toate numerele cuprinse în această zonă. Dacă este posibil, se va afişa numărul minim de operaţii şi apoi poziţia de început a fiecărei operaţii efectuate.

#include<iostream>

#include<fstream>

using namespace std;

int n,k,v[10],a[10],b[10];

void citire()

{

ifstream f("date.in");

int i;

f>>n>>k;

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

 f>>a[i];

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

 f>>b[i];

}

int verifica()

{

int ok=1,i;

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

 if(a[i]!=b[i])

 ok=0;

return ok;

}

void aplica(int i)

{

int j;

for(j=i;j<i+k;j++)

 a[j]=-a[j];

}

int main()

{

int i,j,k=0;

citire();

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

 if(a[i]!=b[i])

 {

aplica(i);

 v[k]=i;

 k++;

 if(verifica()==1)

 i=n+1;

 else i=1;

 }

cout<<k<<" operatii"<<endl;

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

 cout<<v[i]<<" ";

}

8.Într-un acvariu se găsesc n peşti (n<=100) peşti carnivori, numerotaţi de la 1 la n. Ştiind pentru fiecare peşte care sunt peştii pe care-i poate mânca, să se determine ordinea în care peştii se mănâncă astfel încât în final să rămână în acvariu cât mai puţini.

#include <iostream>

#include <cstdio>

using namespace std;

int n,y;

int x[120][120],v[120];

void ales( int pe ){

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

 {

 if ( x[pe][i])

 {

 x[pe][i]=0;

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

 {

 x[j][0]-=x[j][i];

 x[j][i]=0;

 }

 ales ( i );

 cout << pe << "->" << i<<endl;

 return;

 }

 }}

void peste()

{

 int min=120;

 int mini=120;

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

 {

 if ( min > v[i] && x[i][0] )

 {

 min=v[i];

 mini=i;

 }}

 if ( min == 120 ) return;

 x[mini][0]=0;

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

 {

 x[i][0]-=x[i][mini];

 x[i][mini]=0;

 }

 ales(mini);

 peste();

}

int main()

{

 cin>>n;

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

 {

 cin>>x[i][0];

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

 {

 cin>>y;

 x[i][y]=1;

 v[y]++;

 }

 }

 peste();

 return 0;

}