6-2

算法思路分析：

用一个优先队列（最小堆）维护当前可行的节点，每个节点维护着该节点情况下节点层数、目前的权值之和、当前最优解、边上的点等信息，节点的扩展遵循分支定界策略。总体思路是：

①将原图数据构造成一个解空间树的节点，利用定界策略判断是否有解，如果无解直接退出，如果有可能有解则插入到优先队列中；

②若优先队列不为空，那么便从优先队列中取出第一个可行的节点，进入步骤③，如果优先队列为空则退出；

③判断当前节点是否满足解的条件，如果满足便输出解退出，如果不满足便进入步骤④；

④检查当前节点是否可以扩展，不能扩展的话便进入②继续循环，如果能扩展的话则扩展，然后验证扩展到左右节点是否有解，将有解的扩展节点插入到优先队列中，然后进入②继续循环。

#include<iostream>

#include<algorithm>

#include<queue>//使用优先队列实现最小堆

#include<cstring>

using namespace std;

int n,m;

int weight[100];

int G[100][100];

int bestx[100];

class node{

public:

int depth;//层数

int val;//所含顶点的权值之和

int \*x;

int \*c;//顶点关联点

node()

{

x= new int[n+1];

c= new int[n+1];

}

//重载 < ,使优先队列变成最小堆

bool operator < (const node &b)const {return b.val<val;}

};

priority\_queue<node> q;

bool iscover(node a)

{

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

if(!a.x[i] && !a.c[i])

return false;

return true;

}

void insert(node a,int i,int v,bool isleft)

{

node tmp;

tmp.depth=i;

tmp.val=v;

//copy(a.x,a.x+n+1,tmp.x);复制函数从x[0]->x[n+1];

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

tmp.x[i]=a.x[i];

//copy(a.c,a.c+n+1,tmp.c);

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

tmp.c[i]=a.c[i];

if(isleft)

{

tmp.val=v+weight[i];

tmp.x[i]=1;

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

if(G[j][i])

tmp.c[j]++;

}

else

{

tmp.x[i]=0;

}

q.push(tmp);

}

int search()

{

node t;

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

{

t.c[j]=0;

t.x[j]=0;

}

int best;

int v=0;

int i=1;

while(1)

{

if(i>n)

{

if(iscover(t))

{

best=v;

//copy(t.x,t.x+n+1,bestx);

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

bestx[i]=t.x[i];

break;

}

}

else

{

if(!iscover(t))

insert(t,i,v,true);

insert(t,i,v,false);

}

if(q.empty())

break;

else

{

t=q.top();

q.pop();

i=t.depth+1;

v=t.val;

}

}

return best;

}

int main()

{

cin>>n>>m;

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

cin>>weight[i];

memset(G,0,sizeof(G));

int a,b; // 边

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

{

cin>>a>>b;

G[a][b]=1;

G[b][a]=1; //无向图

}

cout<<search()<<endl;

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

cout<<bestx[i]<<" ";

return 0;

}

