4-1会场安排问题

算法分析：

struct time{

int s;

int f;

int no;//会场编号，如果活动后面接了活动，那么No重新变成0,在下一次循环中不再考虑该活动。

time()

{ no=0;

}

};  
 先按照开始时间最早，其次，结束时间最早的顺序排好数组，然后，由贪心算法，最先安排最早开始最快结束的活动，no记录活动的会场编号，其余活动的开始时间在该活动的结束时间后，就变为同一会场，如果循环一次后，没有发现可以接上的活动，那就会场数+=1,同时给此活动编号。

缺点;no最后不一定表示会场编号，如本实验

会场1：1-23（no=0），25-35；

会场2：12-28(no=0)，36-50；

会场3：27-80；

所以改成

struct time{

int s;

int f;

int no;//会场编号

bool anpai;//后面接了其他活动就是true,其他就是false;

time()

{ no=0;

anpai=false;

}

};

源码：

#include<iostream>

#include<vector>

#include<algorithm>

using namespace std;

struct time{

int s;

int f;

int no;//会场编号

bool anpai;//后面接了其他活动就是true,其他就是false;

time()

{ no=0;

anpai=false;

}

};

bool comp(time a,time b)

{

if(a.s<b.s)

return true;

else if(a.s==b.s)

return a.f<b.f;

else

return false;

}

int greedy(vector<time> p)

{

int len=p.size();

sort(p.begin(),p.end(),comp);

int sum=1;

p[0].no=1;

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

{

int j=0;

for(;j<i;j++)

{

if(p[j].anpai==false&&p[j].f<p[i].s)

{

p[i].no=p[j].no;

p[j].anpai=true;//安排好了后面的活动，就不再考虑下一次为此活动接后续活动。

break;

}

}

if(j==i)

{

sum+=1;

p[i].no=sum;

}

}

return sum;

}

int main()

{

int k;

vector<time> p;

cin>>k;

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

{

time h;

cin>>h.s>>h.f;

p.push\_back(h);

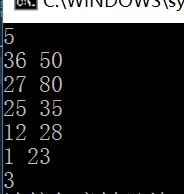
}

cout<<greedy(p)<<endl;

system("pause");

return 0;

}



4-9：

本题很简单，就是运用贪心思想，汽车能跑多少是多少，实在下一加油站不能到了就停下来加一次油。

#include<iostream>

#include<vector>

using namespace std;

void greedy\_jiayou(vector<int> d,int n,int k)

{

int num=0;

int s=0;

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

if(d[i]>n)

{

cout<<"No solution"<<endl;

break;

}

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

{

s+=d[i];

if(s<=n&&s+d[i+1]>n)

{

num++;

s=0;

}

}

cout<<num<<endl;

}

int main()

{

vector<int> d;

int n,k;

cin>>n>>k;

int x;

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

{

cin>>x;

d.push\_back(x);

}

greedy\_jiayou(d,n,k);

system("pause");

return 0;

}

