**活动安排问题**

贪心策略：根据给的活动开始时间和结束时间，活动安排至少有三种看似合理的贪心策略可供选择。

(1)每次从剩下未安排的活动中选择具有最早开始时间且不会与已安排的活动重叠的活动来安排，这样可以增大资源的利用率。

(2)每次从剩下未安排的活动中选择使用时间最短且不会与已安排的活动重叠的活动来安排。

(3)每次从剩下未安排的活动中选择最早结束时间且不会与已安排的活动重叠活动来安排。

通过比较上面三种策略，我们知道应该选用第三种，因为根据“活动结束时间=活动开始时间+使用资源时间”

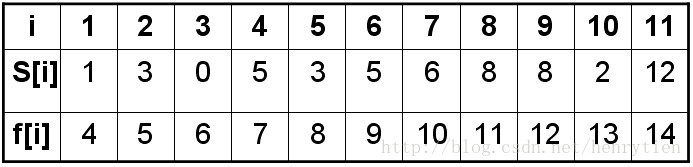
根据问题描述和所选用的贪心策略，对贪心策略求解活动安排的GreedySelector算法设计思路如下：

(1)初始化。将k个活动的开始时间存储在数组S中；将k个活动的结束时间存储在数组F中且按照结束时间的非减序：f1≤f2≤…≤fn,数组S需要做相应的调整；采用集合A来存储问题的解，即所选择的活动集合，活动i如果在集合A中，当且仅当A[i]=ture.

(2)根据贪心策略，算法GreedySelector首先选择活动1，即令A[1]=true.

(3)依次扫描每一个活动，如果活动i的开始时间不小于最后一个选入集合A中的活动结束时间，即活动i和A中活动相同，则将活动i加入集合A中；否则，放弃活动i,继续检查下一个活动与集合A中活动相容性。

例：设待安排的11个活动的开始时间和结束时间按结束时间的非减序排列如下：



源代码：

#include<stdio.h>

#include<stdlib.h>

#define M 11

#define MAX 2000

void select(int num, int s[], int f[]){

int preStart = 0;

int preFinal = MAX;//保证是无限大即可

int i;

int temp;

int OK = 1;

int sel[M];//用来储存相容的活动编号

int selNum = 0;

while (OK){

OK = 0;

for (i = 0; i<M; i++){

if (f[i]<preFinal&&s[i] >= preStart){//寻找开始时间合适地情况下结束时间最早者

preFinal = f[i];

temp = i;

OK = 1;

printf("%d-------------->%d\n", s[i], f[i]);

}

}

if (preFinal != MAX){ //变量的重新赋值

sel[selNum++] = temp;

preStart = f[temp];

preFinal = MAX;

}

}

printf("%d\n", selNum);//相容数量

}

int main(){

int s[] = { 1, 3, 0, 5, 3, 5, 6, 8, 8, 2, 12 };

int f[] = { 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14 };

select(M, s, f);

system("pause");

}

**资源分配问题**

设有资源n（n为整数），分配给m个项目，gi(x)为第i个项目分得资源x(x为整数)所得到的利润。求总利润最大的资源分配方案，也就是解下列问题：  
max z=g1(x1)+g2(x2)+……+gm(xm);  
x1+x2+x3+……+xm=n  
算法设计：项目的投资、利润之间都不是独立的，也找不到一种能由局部信息得到唯一结论的决策策略，应当采用动态规划的算法解决问题。

源代码：

#include<stdio.h>

void main(){

int i, j, k, m = 3, n = 7, rest, a[100][100], gain[100];

float q[100], f[100], temp[100];

printf("How many item?\n");

scanf("%d", &m);

printf("How many money?\n");

scanf("%d", &n);

printf("input one item gain table :\n");

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

scanf("%lf", &q[j]);

f[j] = q[j];

}

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

a[1][j] = j;

}

for (k = 2; k <= m; k++){

printf("Please input another item gain table :\n");

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

temp[j] = q[j];

scanf("%lf", &q[j]);

a[k][j] = 0;

}

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

for (i = 0; i <= j; i++){

if (f[j - i] + q[i]>temp[j]){

temp[j] = f[j - i] + q[i];

a[k][j] = i;

}

}

}

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

f[j] = temp[j];

}

}

rest = n;

for (i = m; i >= 1; i--){

gain[i] = a[i][rest];

rest = rest - gain[i];

}

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

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

}

printf("%.2lf", f[n]);

}