3.5 投资问题

- 设有投资公司，在3月份预计总投资额为m万元，共有n个项目，$G_i(x)$为向第i项工程投资费用为x万元时的预计收益，如何分配资源才能获得**最大利润**？

x	0	1	2	3	4	5	6	7	8
$G_1(x)$	0	5	15	40	80	90	95	98	100
$G_2(x)$	0	5	15	40	60	70	73	74	75
$G_3(x)$	0	4	26	40	45	50	51	53	53

3.5投资问题

动态规划策略解题步骤：

1、分析投资问题的最优解结构特征

2、用递归式描述最优值结构

3、用算法求最优值

4、构造最优解

初步分析：建模

- 设总投资额为m万元，共有n个项目, $F_n(m)$为向n个项目投资m万元所获**最大收益**， $G_i(x_i)$为向第i项工程投资x_i时的收益，则有:

$$F_n(m)=\max\{G_1(x_1)+G_2(x_2)+\ldots\ldots+G_{n-1}(x_{n-1})+G_n(x_n)\}$$

- **问题的解向量（ x_1、 x_2、 ···..、 x_{n-1}、 x_n ）**
  - xi 是投给项目i的投资额，i=1,2,3，······n
- **约束条件： $x_1+x_2+\cdots..+x_{n-1}+x_n=m$**
  - 即 $\sum_{i=1}^{n} x_i=m$ ， $0<=x_i<=m$

1、分析投资问题的最优解结构特征

- **首先，向第n项工程投资，投资额为x_n，所获收益为$G_n(x_n)$**
- **则向剩余n-1项投资：**
  - 投资额为$m-x_n$，所获最大收益为$F_{n-1}(m-x_n)$

- 该投资问题的最大收益为

$$F_n(m)=\max_{x<=m}\{F_{n-1}(m-x)+G_n(x)\}$$

- 更一般化的形式：

$$F_i(j)=\max\{F_{i-1}(j-k)+G_i(k)\},0<=k<=j,1<=i<=n$$

1、分析投资问题的最优解结构特征

- $F_i(j) = \max\{ F_{i-1}(j-k)+G_i(k)\}$,
- $0<=k<=j, 1<=i<=n$

- **子问题的界定：** 由i和j界定
  - i：考虑对前i项项目投资，即对项目1，2,……，i进行投资；
  - j：前i项项目总投资额不超过j。

2、用递归式描述最优值结构

- 最优值f[i][j]：向前i项工程投资，投资额为j时获得的最大收益。

- f[i][j]=
  - g[1][j]　　　　i=1，0<=j<=m
  - max{f[i-1][j-k] + g[i][k] }　　0<=k<=j,1<i<=n, 0<=j<=m

- 原问题的最优值在f[n][m]。

- **标记d[i][j]：** 前i项工程投资额为j时，获得最大收益时，向**当前第i项工程的投资额k**。

向前i项工程投资额为
时f[i][j]获得最大收

向第i项工程
为j时g[i][
益

d[i][j]：前i项工程投资额为j时，
向第i项工程的投资额

- Invest(m,n,f[n][m],g[n][m],d[n][m])
- { for(j=0;j<=m;j++)
- { f[1][j]=g[1][j];d[1][j]=j; }
- for(i=2;i<=n;i++)
- for(j=0;j<=m;j++)
- { f[i][j]=0;
- for(k=0;k<=j;k++)
- { s=f[i-1][j-k]+g[i][k];
- if(s>f[i][j]) { f[i][j]=s; d[i][j]=k; }
- } }

只投资第1项工程，投资额为j时，获得最大收益

- 时间复杂度分析：
  - $O(nm^2)$
- 空间复杂度分析
  - $O(nm)$

当前的

x	0	1	2	3	4	5	6
$G_1(x)$	0	8	13	15	25	36	40
$G_2(x)$	0	4	16	20	28	34	45
$G_3(x)$	0	6	13	20	24	32	40
$G_4(x)$	0	7	20	16	22	34	38

```
Invest(m,n,f[n][m],g[n][m] ,d[n][m])
{ for(j=0;j<=m;j++)
     {  f[1][j]=g[1][j];d[1][j]=j;  }
   for(i=2;i<=n;i++)
       for(j=0;j<=m;j++)
         {  f[i][j]=0;
             for(k=0;k<=j;k++)
               { s=f[i-1][j-k]+g[i][k];
                 if(s>f[i][j])  { f[i][j]=s;d[i][j]=k; }
   }}}
```

- 课堂练习1:
- 已知:
  - n=4, m=6, g数组如上图所示
- 画出f和d数组,求出最优值

4、构造最优解

- d[i][j]：向前i项工程投资为j，获得最大收益时，向第i项工程的投资额存放在d[i][j]
- s为当前剩余投资额, 初始s=m;

$$d[n][m]=k_n\ ,\ s=s-k_n$$

$$d[n-1][s]=k_{n-1}\ ,\ s=s-k_{n-1}$$

$$d[n-2][s]=k_{n-2}\ ,\ s=s-k_{n-2}$$

……..

$$d[1][s]=k_1$$

x	0	1	2	3	4	5	6
$G_1(x)$	0	8	13	15	25	36	40
$G_2(x)$	0	4	16	20	28	34	45
$G_3(x)$	0	6	13	20	24	32	40
$G_4(x)$	0	7	20	16	22	34	38

投资额

```
{
    s=m;  k[n]=d[n][m];
    for(i=n-1;  i>0;  i--)
     {    s = s-k[i+1];
          k[i] = d[i][s];
      }
output  k[i];
}
```

k[n]:第n项工程的投资额
s:当前剩余投资额

- 课堂练习1:
- 已知:
  - n=4, m=6, g数组如上图所示
- 画出f和d数组，求出最优值和最优解

课堂练习2：投资问题

- 设总投资额为m，共有n个项目，下图$G_i(x)$为向第i项工程投资费用为x时的收益，如何分配资源才能获得最大利润？
- 要求画出f和d数组，求出最优值和最优解。

x	0	1	2	3	4	5	6	7	8
$G_1(x)$	0	5	15	40	80	90	95	98	100
$G_2(x)$	0	5	15	40	60	70	73	74	75
$G_3(x)$	0	4	26	40	45	50	51	53	53

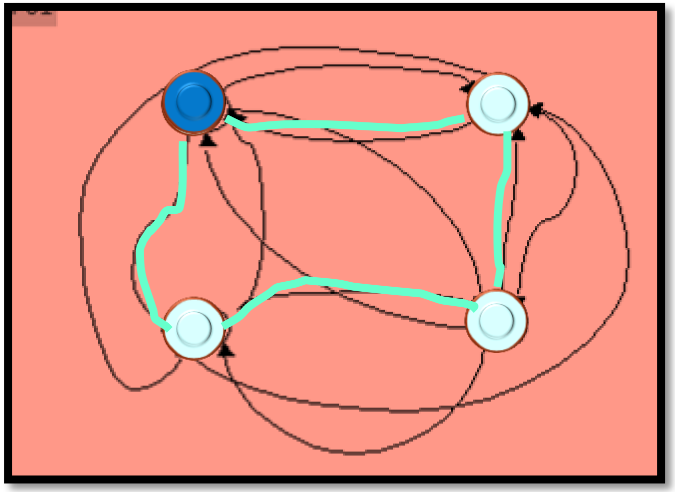
课后练习3：投资问题

- 设总投资额为m，共有n个项目，下图$G_i(x)$为向第i项工程投资费用为x时的收益，如何分配资源才能获得最大利润？
- 要求画出f和d数组，求出最优值和最优解。

x	0	1	2	3	4	5	6
$G_1(x)$	0	8	12	15	25	36	40
$G_2(x)$	0	4	16	20	28	34	42
$G_3(x)$	0	6	13	14	24	32	40
$G_4(x)$	0	7	14	16	22	34	38

3.6旅行商问题(Travelling salesman problem)

- 旅行商问题又称货郎担问题，是指某售货员要到n个城市去推销商品，已知各城市之间的路程（或旅费）。
- 售货员要选定一条从驻地出发经过每个城市一次，最后回到驻地的路线，使总的路程（总旅费）最短（最小）。



Min

3.6旅行商问题(Travelling salesman problem)

□TSP问题的应用领域

- 物流配送
- 交通规划
- 网络节点设置

□TSP问题的解决方法

- 穷举搜索
- 动态规划

3.6旅行售货员问题
(Traveling Saleman Problem)

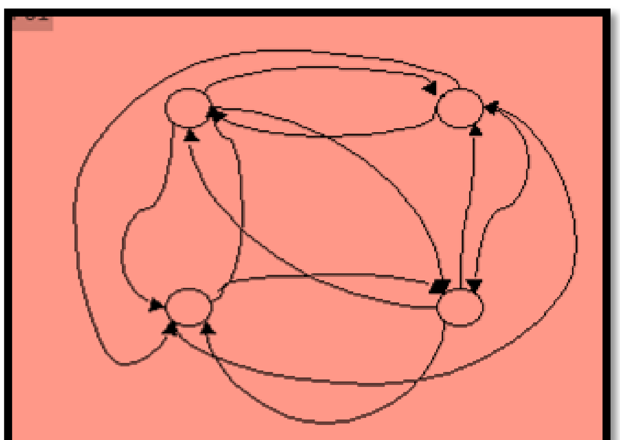
- 使用图论语言描述：
  - 设图G=（V，E，W）是
  - 要求：
    - |V|=n
    - 边（i,j）(i不等于j)的费用c[i][j]为正数。

$$C = \begin{bmatrix} \infty & 2 & 1 & 3 \\ 1 & \infty & 6 & 1 \\ 3 & 4 & \infty & 4 \\ 7 & 1 & 5 & \infty \end{bmatrix}$$

- 图G的一条周游路线是经过V中的每个顶点恰好一次的一条回路。
- 周游路线的费用是这条回路上的所有边的费用之和。
- TSP问题就是要在图G中找出费用最小的周游路线。

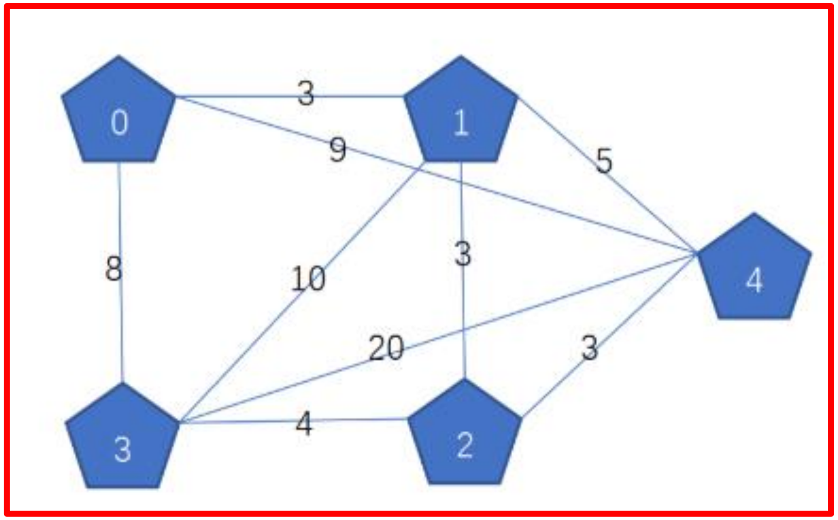
3.6旅行售货员问题
(Traveling Saleman Problem)

- 方法一：穷举法
  - 旅行售货员问题是一个排列问题。
  - 若是一个连通完全图，则由起始点出发的周游路线一共有(n-1)!条，即等于除始结点外的n-1个结点的排列数。
  - 穷举法的时间复杂度：O(n!)



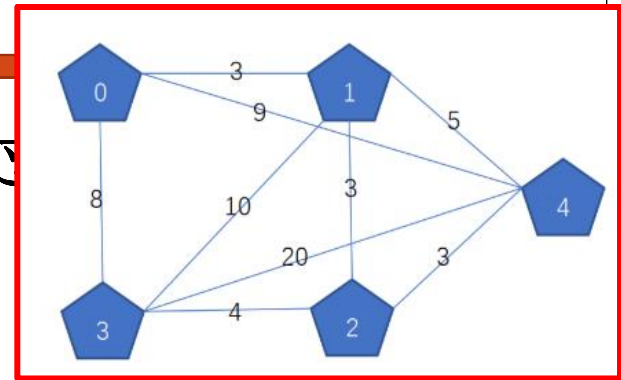
动态规划求解：1. 刻画解的结构特征

- 设从顶点i出发，经过图G中各顶点最终返回顶点i的回路的**最优解结构**为：
  - （$i, x_2, x_3, \ldots .x_n$），$x_k \in (1, 2, \ldots, i-1, i+1, \ldots, n)$
  - 其中：
    - $x_i \neq x_k$
    - $x_2, x_3, \ldots .x_n$ 是n-1个顶点的一个**排列**。



- 反证法

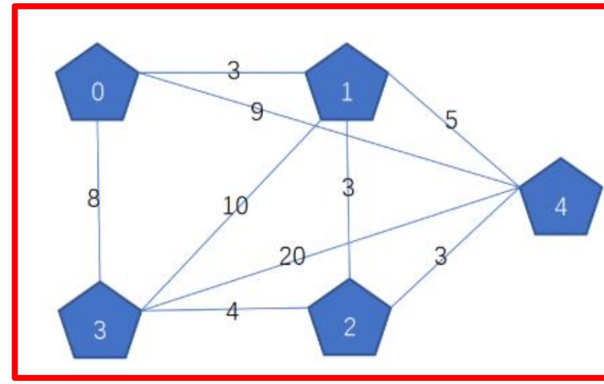
2、建立递归关系

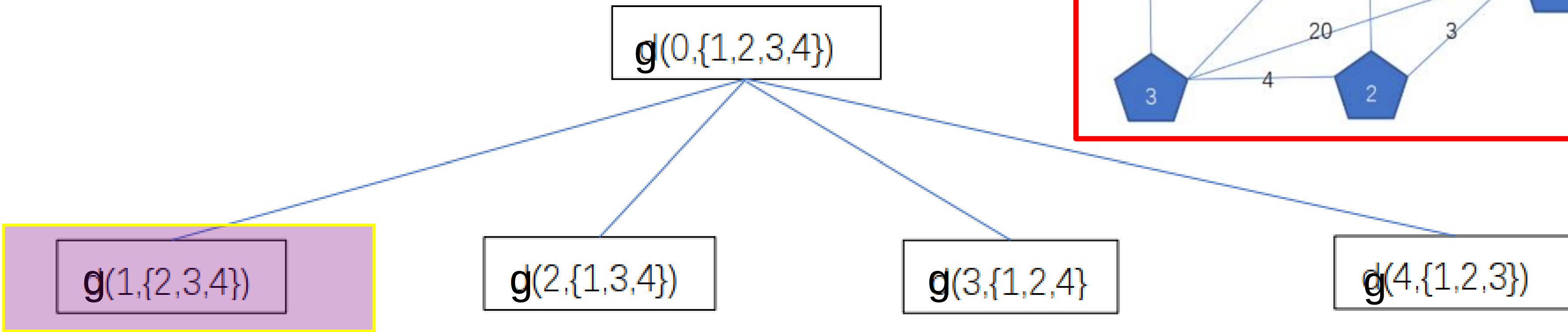


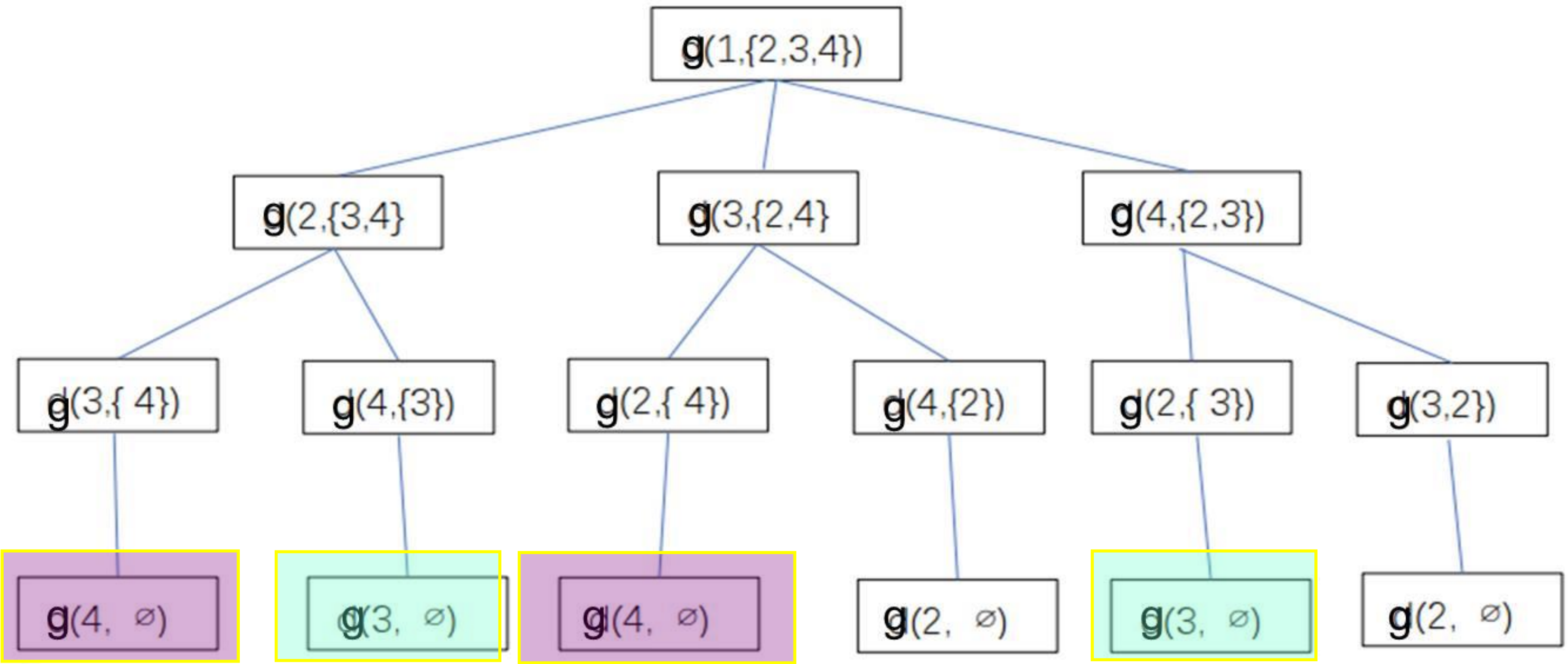
- 定义函数$g(k, V_1)$为从顶点k出发，经过
  并最终返回顶点i的**最短路径长度**
- 则TSP问题的**最优值为$g(i, V-\{i\})$**。
- 由于TSP问题满足最优子结构， $g(i, V-\{i\})$具有如下递归关系：

> ● $g(i,V-\{i\})=\min\{c[i][k]+g(k,V-\{i,k\})\}, k \in V-\{i\}$
>
> ● **边界条件：** $g(k, \Phi)$
>
> ● $=c[k][i]$

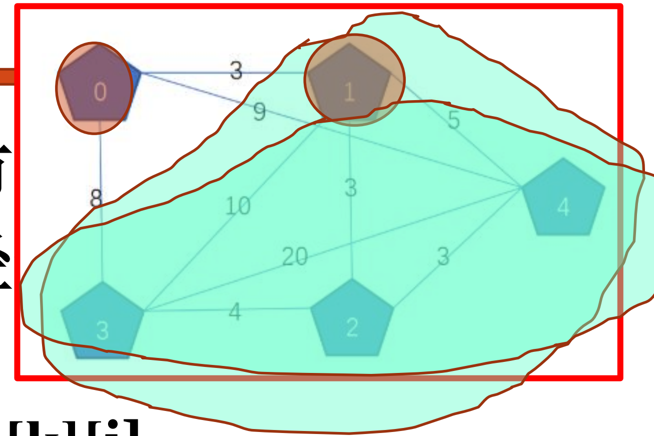
□举例说明


0
1
2
3
4
3
9
5
8
10
3
20
3
4


g(0,{1,2,3,4})
g(1,{2,3,4})
g(2,{1,3,4})
g(3,{1,2,4}
g(4,{1,2,3})


g(1,{2,3,4})
g(2,{3,4}
g(3,{2,4}
g(4,{2,3})
g(3,{ 4})
g(4,{3})
g(2,{ 4})
g(4,{2})
g(2,{ 3})
g(3,2})
g(4, ∅)
g(3, ∅)
g(4, ∅)
g(2, ∅)
g(3, ∅)
g(2, ∅)

3、计算最优值



- 输入：n个城市及费用矩阵c, 出发城市
- 输出：从城市i出发并返回的最终路径

- **Step1:初始化：对k=1,2,…,n, g(k, null)=c[k][i]**
- **Step2:对V-{i}的含有m（m=1,2,…,n-2）个元素的子集A依次计算：**
  - 对不属于A的所有顶点（i除外）：
    - g(j, A)=min{ c[j][t]+g(t, A-{t})} t ∈ A
    - **p(j, A)=t;** //使g(j, A)取最小值的t

- **Step3: 计算g(i, V-i)**
  - **g(i, V-i)=min{ c[i][t]+g(t,V-{i,t})}**
  - **Mindis= g(i, V-i); p(i,V-i)=使g(i, V-i)取最小值的t;**
  - **Return (Mindis);**

4、构造最优解

- TSP问题的**最优解**是path（i，$k_1, k_2, \cdots\cdots k_{n-1}$）
- **Step1:**初始化 **j= i; V_1={i}; path[0]=i;**
- **Step2: for (t=1;t<=n-1;t++)**
- **{ k=p(j, V-V1);**
- **path[t]=k;**
- **$V_1=V_1 \cup \{ k \}$;**
- **j=k;**
- **}**
- **Step3:** 输出**path[ ]**

5、算法复杂度

- **需要进行计算的次数与V的大小k(k=0,1,2,…,n-1)的子集的个数相关，并且每确定一个g(j,A)，需要k次加法和k-1比较运算。**
- 计算中所需的加法和比较的次数,可以算出**时间复杂度**为**O(n^2*2^n)**
- **算法改进：**通过改进集合操作降低比较次数，利用**二进制表示集合**。确定元素k是否在集合S中的比较次数为1，从而降低了时间复杂度到O(n2^n)

3.7 动态规划小结

- **利用动态规划算法求解问题的过程是一个多阶段最优决策的过程，一般可分为如下四个求解步骤。**
  - (1) 分析问题最优解的结构，并刻画其结构特征，证明其满足最优子结构性质。通过对问题的分析，将问题最优解的求解分为若干个阶段，各个阶段之间应具有明确的先后顺序关系；
  - (2) 建立最优值的递归关系，问题的最优值对应的递归关系是各阶段决策的依据；
  - (3) 计算最优值。根据最优值对应的递归关系，按照自底向上的顺序计算问题的最优值；
  - (4) 依据最优值求解过程中记录下来的最优解的信息，构造问题的一个最优解。