# 第 5 章　动态规划

# 第 5 章的主要内容

- 多阶段决策和最优化原理
- 离散变量的动态规划
  - 阶段数固定（定期）的动态规划
  - 阶段数不固定（不定期）的动态规划
- 连续变量的动态规划

# 5.1/5.2 多阶段决策和最优化原理

## 动态规划研究的问题

- 动态规划起源于 **1950** 年代，创始人为 **R. Bellman**。
- 动态规划所研究的对象是多阶段决策问题。
  - 这样的问题可以转化为一系列相互联系的单阶段优化问题。 在每个阶段都需要作出决策。
  - 每个阶段的决策确定以后，就得到一个决策序列，称为策略。
  - 多阶段决策问题就是求一个策略，使各阶段的总体

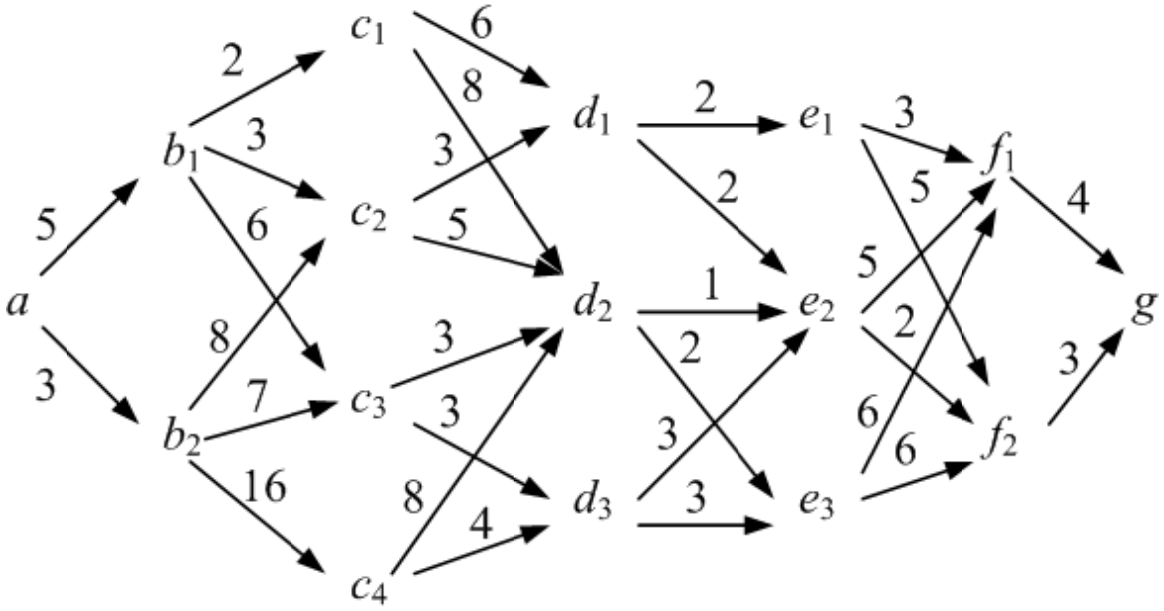
目标达到最优（如最小化费用，或最大化收益）。

- 相对于线性规划一次性地对一个问题求出整体最优解，多阶段决策问题的这种解决办法（一个阶段一个阶段地，而不是一次性地）称为动态规划（**Dynamic Programming**）。而原来的线性规划方法被称为静态规划。

## 问题举例一：最短路问题

- 如图，求从 ***a*** 到 ***g*** 的最短路。

- 由于这是一个多阶段图（分层图），该图上的最短路问题是一个多阶段决策（优化）问题。



## 如何求解？

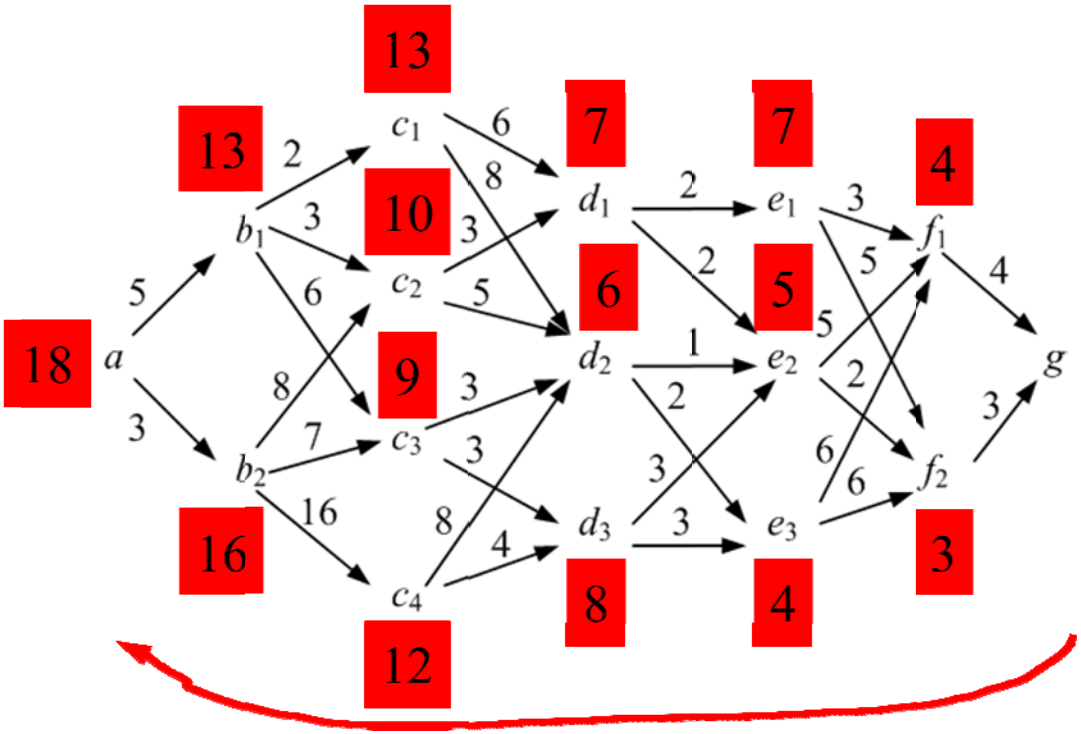
- 由于这是一个多阶段图，从 $a$ 到 $g$ 的任何一条路径的边数都是 6。
- 从 $a$ 到 $g$，必然要经过 $a$ 的下一个阶段中的顶点 $b_1$ 或 $b_2$。因此，从 $a$ 到 $g$ 的最短路就是从 $a$ 到 $b_1$，然后再从 $b_1$ 走到 $g$，以及从 $a$ 到 $b_2$，再从 $b_2$ 走到 $g$，两种走法中最短的一个。
- 于是，定义 $f_k(u, g)$为从当前顶点 $u$ 开始经过 $k$ 条边到

达 $g$ 的最短路长度。则有：

$$f_k(u,g)=\begin{cases}\min\limits_{v\in N^+(u)}\{l(u,v)+f_{k-1}(v,g)\}, & k\geq 2\\ d(u,g), & k=1\end{cases}$$。

- 原问题即是求 $f_n(a,g)$（$n=6$）。
- 上述过程实际上是动态规划中“后向优化”的方法。

# 逆向求解递推方程（标号法）



# 动态规划表格

- 行标 $i$：从目标顶点 $g$ 开始，倒数第 $i$ 层（即，计算 $f_i(u)$）。
- $p_1$, $p_2$, $p_3$, $p_4$：倒数第 $i$ 层的自上而下的各个顶点（最多 4 个）。

| | $p_1$ | $p_2$ | $p_3$ | $p_4$ |
|---|---|---|---|---|
| 1 | 4 | 3 | × | × |
| 2 | 7 | 5 | 9 | × |
| 3 | 7 | 6 | 8 | × |
| 4 | 13 | 10 | 9 | 12 |

| | $p_1$ | $p_2$ | $p_3$ | $p_4$ |
|---|---|---|---|---|
| 1 | 1 | 1 | × | × |
| 2 | 1 | 2 | 2 | × |
| 3 | 2 | 2 | 2 | × |
| 4 | 1 | 1 | 2 | 3 |

| | | | | |
|---|---|---|---|---|
| 5 | 13 | 16 | × | × |
| 6 | 18 | × | × | × |

| | | | | |
|---|---|---|---|---|
| 5 | 2 | 3 | × | × |
| 6 | 1 | × | × | × |

- 最短路为：$a \to b_1 \to c_2 \to d_1 \to e_2 \to f_2 \to g$。

递归如何？循环如何？

- 递归程序

**$f(i, u, g)$**

**1　if $i = 1$ then return $d(u, g)$，**

**2　else return $\min_{v \in N(u)} \{d(u, v) + f(i-1, v, g)\}$。**

- 循环程序

$f(u, g)$

1 $p \leftarrow (a, b_1, c_1, d_1, e_1, d_1, f_1, g)$。
2 for $b \leftarrow b_1$ 到 $b_2$ do
3 for $c \leftarrow c_1$ 到 $c_4$ do
4 for $d \leftarrow d_1$ 到 $d_3$ do
5 for $e \leftarrow e_1$ 到 $e_3$ do
6 for $f \leftarrow f_1$ 到 $f_2$ do
7 $q \leftarrow (a, b, c, d, e, f, g)$。

```
 8                                    if q 的长度 < p 的长度 then
                                        p ← q。
 9                                endfor    /* f */
10                            endfor    /* e */
11                        endfor    /* d */
12                    endfor    /* c */
13            endfor    /* b */
14            return p。
```

# 问题举例二：资源分配问题

- 设有数量为 $x$ 的某种资源，将它投入两种生产方式 $A$ 和 $B$ 中（或称为投给部门 $A$ 和部门 $B$）。
- 若投给部门 $A$ 的数量为 $z$，则可获收益 $g(z)$，回收 $az$，其中 $a$（$0 \le a \le 1$）为部门 $A$ 的回收率。类似地，若投给部门 $B$ 的数量为 $z$，则可获收益 $h(z)$，回收 $bz$，其中 $b$（$0 \le b \le 1$）为部门 $B$ 的回收率。
- 连续投放 $n$ 个阶段，问每个阶段如何分配资源才能

使总收入最大？

## 再描述一遍

- 设第 $k$ 个阶段的资源总数为 $x_k$，投给部门 $A$ 的资源数量为 $y_k$。则投给部门 $B$ 的数量为 $x_k - y_k$。于是可得到收入 $g(y_k) + h(x_k - y_k)$，回收 $ax_k + b \cdot (x_k - y_k)$。
- 因此，问题就成为：求 $y_1, y_2, \cdots, y_n$，
  最大化 $\sum_{1 \le k \le n} g(y_k) + h(x_k - y_k)$ ，且满足条件

$$x_1 = x$$

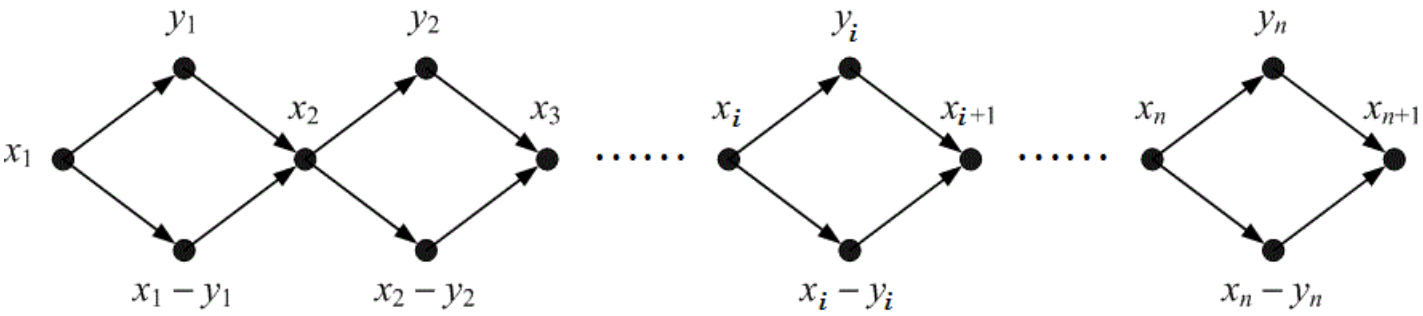
$$x_2 = ay_1 + b(x_1 - y_1)$$

……

$$x_n = ay_{n-1} + b(x_{n-1} - y_{n-1})$$

$$y_k \geq 0,\quad x_k \geq 0,\qquad k = 1..n-1$$

如何求解？



- 令 $f_k(x)$表示当前资源数量为 $x$，再经过 $k$ 个阶段投放完成系统目标，所得到的最大总收入。
- 则有：

$$f_k(x)=\begin{cases}\max\limits_{0\le y\le x}\{g(y)+h(x-y)+f_{k-1}(ay+b(x-y))\}, & k\ge 2\\ \max\limits_{0\le y\le x}\{g(y)+h(x-y)\}, & k=1\end{cases}$$。

- 原问题即是求 $f_n(x)$。
- 上述方法实际上是动态规划中的“后向优化”方法。

## 例 5.1.2：离散变量的资源分配问题

- 今有 **1000** 台机床（$x = 1000$），投放到 ***A***、***B*** 两个部门。
- 若给部门 ***A*** 投放 ***z*** 台机床，则产生效益 $g(z) = z^2$，回收 $0.8z$ 台机床（$a = 0.8$）。
- 若给部门 ***B*** 投放 ***z*** 台机床，则产生效益 $h(z) = 2z^2$，回收 $0.4z$ 台机床（$b = 0.4$）。
- 问连续投放 **5** 年（$n = 5$），每年如何投放，可使 **5** 年

的总收益最大？

## 如何求解？

- 行标 $k$：到达目标，还需要多少个阶段，即，计算 $f_k(x)$。
- 列标 $x$：可能的资源数。

| $k \backslash x$ | 0 | 1 | 2 | 3 | … | | | 1000 |
|---|---|---|---|---|---|---|---|---|
| 1 | $f_1(0)$ | $f_1(1)$ | $f_1(2)$ | $f_1(3)$ | … | | | $f_1(1000)$ |
| 2 | $f_2(0)$ | $f_2(1)$ | $f_2(2)$ | | … | | | $f_2(1000)$ |
| 3 | $f_3(0)$ | | | | … | | | $f_3(1000)$ |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 4 | $f_4(0)$ | | | | … | | | $f_4(1000)$ |
| 5 | $f_5(0)$ | | | | … | | | $f_5(1000)$ |

- 问题的目标就是计算 $f_5(1000)$。
- 从左上到右下，计算整个表，可求得问题的解。

## 计算一个单元格

***f*(*k*, *x*)**

**1** $v \leftarrow -\infty$。

**2** **for** $y \leftarrow 0$ 到 $x$ **do**

3 $t \leftarrow g(y) + h(x-y) + \text{table}(k-1, \lfloor ay + b(x-y) \rfloor)$。

/* 令 table(0, $x$) = 0 。 */

4 if $t > v$ then $v \leftarrow t$。

5 endfor

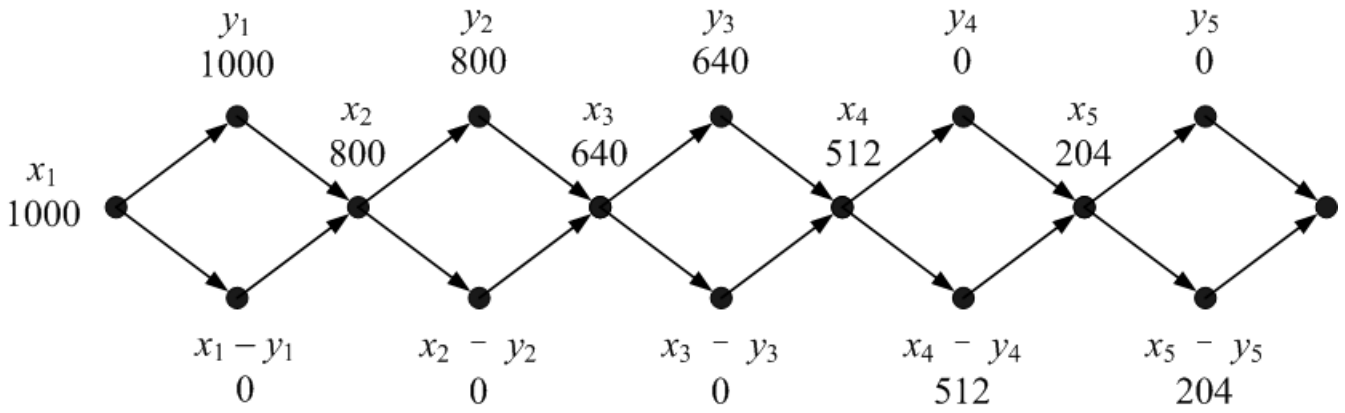
6 return $v$。

- 每计算一个单元格的 $f_k(x)$，都需要计算一个 $\max_{0 \le y \le x}\{\ldots\}$函数。因此，尽管使用表格暂存了计算结果，为计算出最后的 $f_n(x)$ 仍需要大量的计算。
- 小技巧：不用每行都从 0 计算到 1000。每年无论

如何投放，回收的机床最多是 $0.8x$ 台（$\max\{a, b\} = 0.8$）。例如表格第 5 行表示最后一个阶段，其前面有 4 个阶段。因此对于第 5 行，只需要从 0 计算到 $0.8^4 \times 1000 \cong 4096$。

- 但动态规划法已经比直接用递归的方法解递推方程减少了大量的计算。

# 计算结果




$f_5(1000) = 2657120$ $f_3(640) = 950272$ $f_1(204) = 83232$

$f_4(800) = 1657120$ $f_2(512) = 607520$

# 递归的方法

```
f(k, x)
  1    v ← 0。
  2    if k = 1 then
  3          for y ← 0 到 x do
  4                t ← g(y) + h(x − y)。
  5                if t > v then v ← t。
  6          endfor
```

```
 7    else
 8        for y ← 0 到 x do
 9            t ← g(y) + h(x − y) + f(k − 1, ay + b(x − y))。
10            if t > v then v ← t。
11        endfor
12    endif
13    return v。
```

# 多阶段决策问题

- 有一个系统，可以分成若干个阶段。
- 任意一个阶段 $k$，系统的状态可以用 $x_k$ 表示（可以是数量、向量、集合等）。
- 每一状态 $x_k$ 都有一个决策集合 $Q_k(x_k)$，在 $Q_k(x_k)$ 中选定一个决策 $q_k \in Q_k(x_k)$，状态 $x_k$ 就转移到新的状态 $x_{k+1} = T_k(x_k, q_k)$，并且得到效益（或费用）$R_k(x_k, q_k)$。

- 系统的目标就是在每一个阶段都在它的决策集合中选择一个决策，使所有阶段的总效益达到最大（或总费用达到最小）。
- 这样的多阶段决策问题通常使用动态规划方法来求解。

## 动态规划的最优子结构性质

动态规划的最优化原理：需要问题的最优解具有如下所述的最优子结构性质：

- 一个多阶段决策问题，假设其最优策略的第一阶段的决策为 $q_1$，系统转移到的新状态为 $x_2$。则该最优策略以后诸决策对以 $x_2$ 为初始状态的子问题而言，必须构成其最优策略。
- 该子问题与原问题是同一类问题，只是问题规模下降了。
- 当观察到问题解的最优子结构性质时，就意味着问题可能用动态规划法求解。

## 动态规划的子问题重叠性质

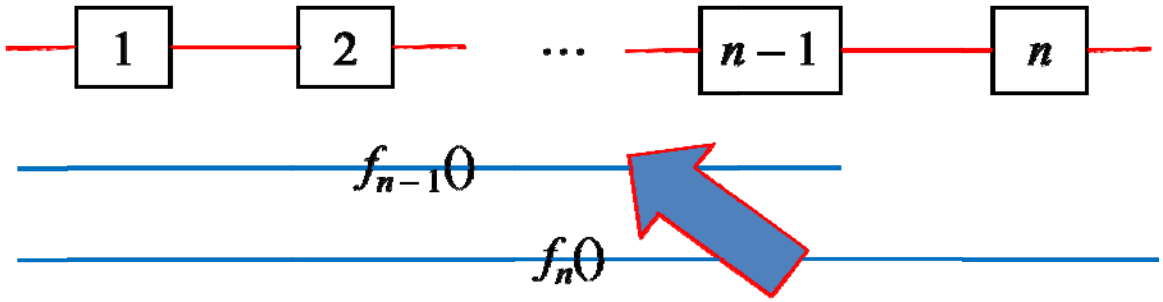
- 从算法角度而言，（离散变量的）动态规划所依赖的另一个要素是子问题重叠性质：一个问题的求解可以划分成若干子问题的求解，而处理这些子问题的计算是部分重叠的。
- 动态规划法利用问题的子问题重叠性质设计算法，能够节省大量的计算。对一些看起来不太可能快速求解的问题，往往能设计出多项式时间算法。

- 在算法理论中，多项式时间算法通常被认为是“有效的算法”（**efficient algorithm**）。

## 前向优化

- 写动态规划递推方程时，一般有两种写法：前向优化和后向优化。假设问题有 $n$ 个阶段。
- 定义 $f_k()$ 为问题的前 $k$ 个阶段（从第 1 阶段到第 $k$ 阶段）的最优解值，然后将 $f_k()$递推至 $f_{k-1}()$。
- 最后写出递推的终止条件 $f_1()$的表达式。

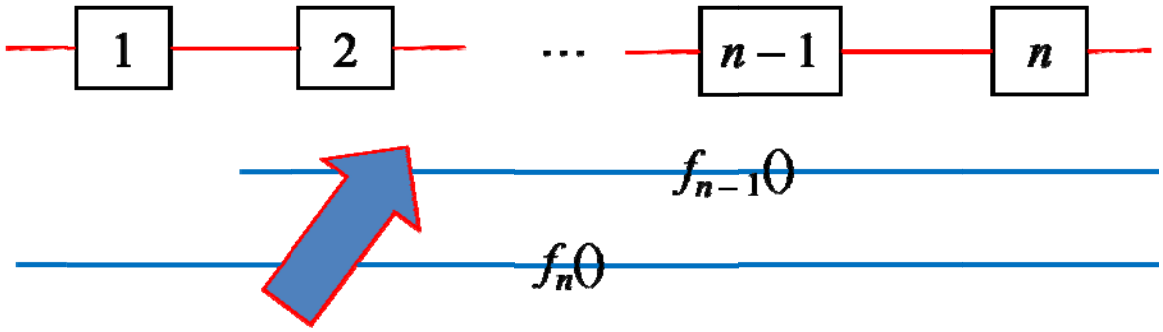
- 原问题就是计算 $f_n()$。这称为前向优化。




## 后向优化

- 假设问题有 $n$ 个阶段。
- 定义 $f_k()$ 为问题的后 $k$ 个阶段（从第 $n-k+1$ 阶段到第 $n$ 阶段）的最优解值，然后将 $f_k()$递推至 $f_{k-1}()$。

- 最后写出递推的终止条件 $f_1()$的表达式。
- 原问题就是计算 $f_n()$。这称为后向优化。




## 说明

- 前面给出的两个例子，最短路问题和资源分配问题，

都是采用后向优化技术解决的。

- 原则上，多阶段决策问题既可以使用前向优化技术解决，也可以使用后向优化技术解决。
- 依据问题不同，前向优化和后向优化其中的一种或二者是“自然”的解法。

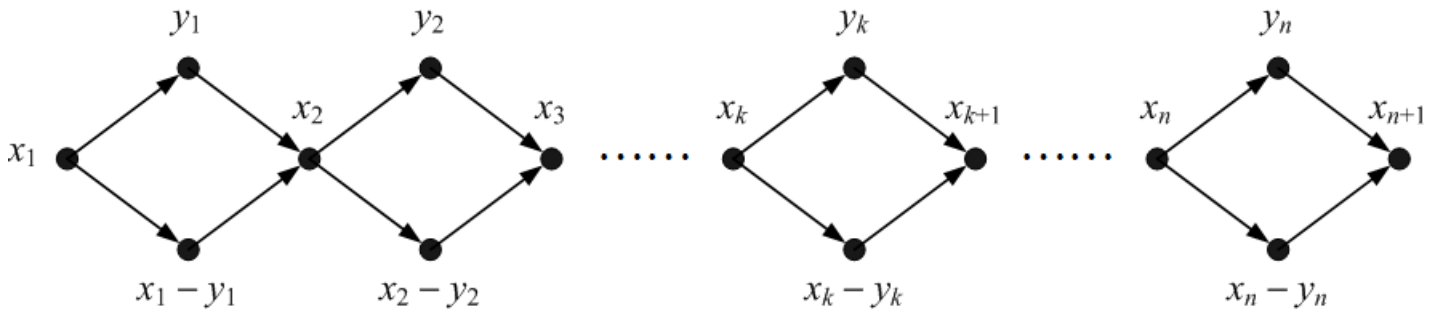
## 最短路问题，前向优化

- 定义 $f_k(a, u)$为从顶点 $a$ 经过 $k$ 条边到达当前顶点 $u$ 的最短路长度。则有：

$$f_k(a,u)=\begin{cases}\min\limits_{v\in N^-(u)}\{f_{k-1}(a,w)+l(w,u)\}, & k\geq 2\\ l(a,u), & k=1\end{cases}$$。

- 原问题即是求 $f_n(a, g)$（$n = 6$）。

资源分配问题，前向优化



- 令 $f_k(x_{k+1})$表示从第 1 阶段连续生产到第 $k$ 阶段，还余下恰好 $x_{k+1}$ 份资源，所产生的最大效益。
- 则有如下递推方程：当 $k = 2..n$ 时，

$$f_k(x_{k+1}) = \begin{cases} \max\limits_{0 \le y_k \le \frac{1}{a} x_{k+1}} \{ f_{k-1}(x_k) + g(y_k) + h(x_k - y_k) \}, & \min\{a^{k-1}x_1, b^{k-1}x_1\} \le x_k \le \max\{a^{k-1}x_1, b^{k-1}x_1\} \\ -\infty, & \text{o.w.} \end{cases}$$

给定 $x_{k+1}$、$y_k$，由于 $x_k$、$x_{k+1}$、$y_k$ 需满足 $ay^k + b(x_k - y_k) = x_{k+1}$，

可直接计算出 $x_k = \frac{1}{b}\left(x_{k+1} + (b-a)y_k\right)$。

- 当 $k = 1$ 时，

$$f_1(x_2)=\begin{cases} g(y_1)+h(x_1-y_1), & 0\le y_1=\dfrac{bx_1-x_2}{b-a}\le x_1 \\ -\infty, & \text{o.w.} \end{cases}$$。

由已知 $x_1$、$x_2$，因为 $x_1$、$x_2$、$y_1$ 需要满足 $ay_1+b(x_1-y_1)=x_2$，可直接计算出 $y_1=\dfrac{bx_1-x_2}{b-a}$。

- 最后，对 $\min\{a^n x_1, b^n x_1\}\le x_{n+1}\le \max\{a^n x_1, b^n x_1\}$ 计算所有可能的 $f_n(x_{n+1})$，然后选取一个最大值，即为原问题的最优解。

## 动态规划所研究的问题

- 阶段数固定还是不固定？
  - 阶段数固定（也称为“定期”），指阶段数有限且固定。
  - 阶段数不固定（也称为“不定期”）：指阶段数有限但不固定，以及阶段数无限大两种情况。
- 离散变量还是连续变量？
- 很多阶段数不明显的优化问题，也可以使用动态规

划方法求解。

# 5.3 阶段数固定的离散变量的优化问题

## 本节内容

- 背包问题
- 最长公共子序列问题
- 旅行售货员问题

## 背包问题（The Knapsack Problem）

- 实例：有一个背包，总承重为整数 $W$。

  有 $n$ 个物品，每个物品重量为 $w_i$，价值为 $v_i$，$i = 1..n$。

$w_i$和 $v_i$ 均为整数。

- 目标：装入背包若干物品，使其总重量不超过 $W$，总价值最大。

## 例子

- $n = 4, W = 5$。

| $i$ | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| $v_i$ | 6 | 10 | 12 | 13 |
| $w_i$ | 1 | 2 | 3 | 4 |

- 贪心策略 1：每次装当前价值最大的物品。

  找到的解：**13 + 6 = 19**。
- 贪心策略 2：每次装当前重量最小的物品（以留出尽可能多的空间给将来的物品使用）。

  找到的解：**6 + 10 = 16**。
- 贪心策略 3：每次装当前“性价比”最高的物品。

  找到的解：**6 + 10 = 16**。
- 最优解：**10 + 12 = 22**！

## 第一种解法

- 定义 $f(i, j)$表示将物品 $1..i$ 中的若干装入总容量为 $j$ 的背包，所获得的最大价值。则原问题是求 $f(n, W)$。
- 若 $j < w_i$，则第 $i$ 个物品必不能装入背包。
- 若 $j \geq w_i$，则第 $i$ 个物品可以装入背包。到底是否装入背包，取决于装入第 $i$ 个物品，再装入获得价值 $f(i-1, j-w_i)$的那些物品，所获得的总价值，以及将 $1..i-1$ 物品中的若干装入容量为 $j$ 的背包所获得的总价值 $f(i$

– 1, *j*)哪个更大。

- 则有：

$$f(i,j)=\begin{cases} f(i-1,j), & i\geq 1,\ j\geq 1,\ j<w_i \\ \max\{f(i-1,j-w_i)+v_i,\ f(i-1,j)\}, & i\geq 1,\ j\geq w_i \\ 0, & i=0\text{或}j=0 \end{cases}$$

## 动态规划表

| *f* | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|

| *h* | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 6 | 6 | 6 | 6 | 6 |
| 2 | 0 | 6 | 10 | 16 | 16 | 16 |
| 3 | 0 | 6 | 10 | 16 | 18 | 22 |
| 4 | 0 | 6 | 10 | 16 | 18 | 22 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | × | × | × | × | × | × |
| 1 | × | (0,0) +6 | (0,1) +6 | (0,2) +6 | (0,3) +6 | (0,4) +6 |
| 2 | × | (1,1) | (1,0) +10 | (1,1) +10 | (1,2) +10 | (1,3) +10 |
| 3 | × | (2,1) | (2,2) | (2,3) | (2,1) +12 | (2,2) +12 |
| 4 | × | (3,1) | (3,2) | (3,3) | (3,4) | (3,5) |

- $f$ 表记录 $f(i, j)$函数的值。
- $h$ 表记录对应的 $f(i, j)$是如何计算出来的。即，$f(i, j) = f(i - 1, j)$，还是 $f(i, j) = f(i - 1, j - w_i) + v_i$。后一种情况表明装了物品 $i$。

## 时间复杂度

- 以上动态规划法的时间复杂度为 $O(nW)$。
- 这个时间复杂度不是多项式的，原因在于 $W$ 是问题输入中的一个整数。

## 第二种解法

- 定义 $f(i, j)$表示在物品 $1..i$ 中选择若干装入背包，使其总价值恰好为 $j$，总重量最小，这样的若干物品的总重

量。若这样的集合不存在，则 $f(i, j) = \infty$。

- 则有：

$$f(i,j)=\begin{cases}0, & i=0,\ j=0\\ \infty & i=0,\ j\geq 1\\ 0, & i\geq 1,\ j=0\\ f(i-1,j), & i\geq 1, v_i > j\geq 1\\ \min\{f(i-1,j),\ f(i-1,j-v_i)+w_i\}, & i\geq 1,\ j\geq v_i\end{cases}$$

- 则原问题是寻找满足 $f(i, j) \leq W$ 的最大的 $j$。

# 例子

- 背包问题实例：$n = 4$，$W = 5$。

| $i$ | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| $v_i$ | 6 | 10 | 12 | 13 |
| $w_i$ | 1 | 2 | 3 | 4 |

- 动态规划表：

| | $j$ | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $i$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 |
| 0 | 0 | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ |
| 1 | 0 | ∞ | ∞ | ∞ | ∞ | ∞ | 1 | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ |
| 2 | 0 | ∞ | ∞ | ∞ | ∞ | ∞ | 1 | ∞ | ∞ | ∞ | 2 | ∞ | ∞ | ∞ | ∞ | ∞ | 3 | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ |
| 3 | 0 | ∞ | ∞ | ∞ | ∞ | ∞ | 1 | ∞ | ∞ | ∞ | 2 | ∞ | 3 | ∞ | ∞ | ∞ | 3 | ∞ | 4 | ∞ | ∞ | ∞ | 5 | ∞ | ∞ | ∞ | ∞ | ∞ | 6 | ∞ | ∞ |
| 4 | 0 | ∞ | ∞ | ∞ | ∞ | ∞ | 1 | ∞ | ∞ | ∞ | 2 | ∞ | 3 | 4 | ∞ | ∞ | 3 | ∞ | 4 | 4 | ∞ | ∞ | 5 | 6 | ∞ | 7 | ∞ | ∞ | 6 | 7 | ∞ |

表格中 $j$ 的上界 $= \min\{ \max\{v_i/w_i\}*W, \sum v_i\} = 30$。

## 最长公共子序列（LCS）问题

**LCS（Longest Common Subsequence）问题**

- 实例：序列 $X = x_1x_2\cdots x_m$，$Y = y_1y_2\cdots y_n$。
- 目标：找 $X$ 和 $Y$ 的一个最长公共子序列。
- 子序列：若有 $l$ 个数满足 $1 \le s_1 < s_2 < \cdots < s_l \le n$，则称 $x_{s1}x_{s2}\cdots x_{sl}$ 为 $X$ 的一个子序列。

## 递推公式

- 从 $X$ 和 $Y$ 的尾部向前，考虑它们的最长公共子序列。假设现在扫描的字符是 $x_i$ 和 $y_j$。
- 若 $x_i = y_j$，则 $x_1..x_i$ 与 $y_1..y_j$ 的最长公共子序列就是 $x_1..x_{i-1}$ 与 $y_1..y_{j-1}$ 的最长公共子序列，再加上字符 $x_i$。
- 否则，$x_1..x_i$ 与 $y_1..y_j$ 的最长公共子序列，就是 $x_1..x_{i-1}$ 与 $y_1..y_j$ 的最长公共子序列，以及 $x_1..x_i$ 与 $y_1..y_{j-1}$ 的最

长公共子序列中，较长的那一个。

- 定义 ***v*(*i*, *j*)**为 $x_1..x_i$ 与 $y_1..y_j$ 的最长公共子序列的长度。

  则有：

$$v(i,j)=\begin{cases} v(i-1,j-1)+1, & \text{若} i>0, j>0, x_i=y_j \\ \max\{v(i-1,j), v(i,j-1)\}, & \text{若} i>0, j>0, x_i\neq y_j \\ 0, & \text{若} i=0 \text{或} j=0 \end{cases}$$

。

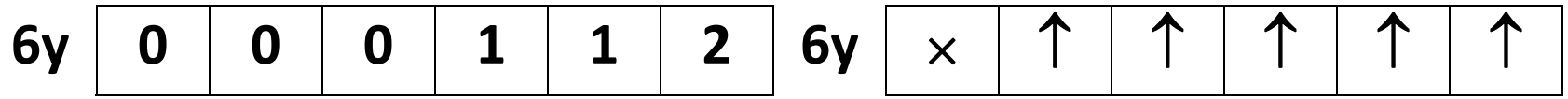
- 原问题就是计算 ***v*(*m*, *n*)**。

# 例子

***X*** **= monkey，** ***Y*** **= human。**

| *v* | 0 | 1h | 2u | 3m | 4a | 5n |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1m | 0 | 0 | 0 | 1 | 1 | 1 |
| 2o | 0 | 0 | 0 | 1 | 1 | 1 |
| 3n | 0 | 0 | 0 | 1 | 1 | 2 |
| 4k | 0 | 0 | 0 | 1 | 1 | 2 |
| 5e | 0 | 0 | 0 | 1 | 1 | 2 |

| *f* | 0 | 1h | 2u | 3m | 4a | 5n |
|---|---|---|---|---|---|---|
| 0 | × | × | × | × | × | × |
| 1m | × | ↑ | ↑ | \ | ← | ← |
| 2o | × | ↑ | ↑ | ↑ | ↑ | ↑ |
| 3n | × | ↑ | ↑ | ↑ | ↑ | \ |
| 4k | × | ↑ | ↑ | ↑ | ↑ | ↑ |
| 5e | × | ↑ | ↑ | ↑ | ↑ | ↑ |




# LCS 之动态规划算法

**Algorithm LCS(*X*, *Y*)**

计算 $X = x_1x_2...x_m$ 和 $Y = y_1y_2...y_n$ 的最长公共子序列。

**1 for *i* ← 1 到 *m* do**

**2 *c*[*i*, 0] ← 0。**

**3 endfor**

**4 for *j* ← 0 到 *n* do**

```
 5          c[0, j] ← 0。
 6 endfor
 7 for i ← 1  到  m do
 8          for j ← 1  到  n do
 9                    if x[i] = y[j] then
10                              c[i, j] ← c[i – 1, j – 1] + 1。
11                              f[i, j] ← “ \”。
12                    else
13                              if c[i – 1, j] ≥ c[i, j – 1] then
```

```
14                          c[i, j] ← c[i − 1, j]。
15                          f[i, j] ← “↑”。
16                      else
17                          c[i, j] ← c[i, j − 1]。
18                          f[i, j] ← “←”。
19                      endif
20                  endif
21              endfor
22 endfor
```

**23 return *c*, *f*。**

## 时间复杂度

- 以上动态规划法（**LCS** 问题）的时间复杂度为 **O(*mn*)**。
- 这是一个多项式的时间复杂度，表明 **LCS** 问题是多项式时间可解的。

## 旅行售货员（TSP）问题

- 旅行售货员问题是图论中一个著名问题。

- 实例：图 $G = (V, E)$，每条边$(v_i, v_j)$上有长度 $d(v_i, v_j)$。
- 目标：找一个最短的圈，走过所有的顶点。
- 等价描述：从 $v_1$ 点出发，经过其余顶点（$v_2$, …, $v_n$）各一次，最后返回 $v_1$ 的最短的圈。

## 递推方程

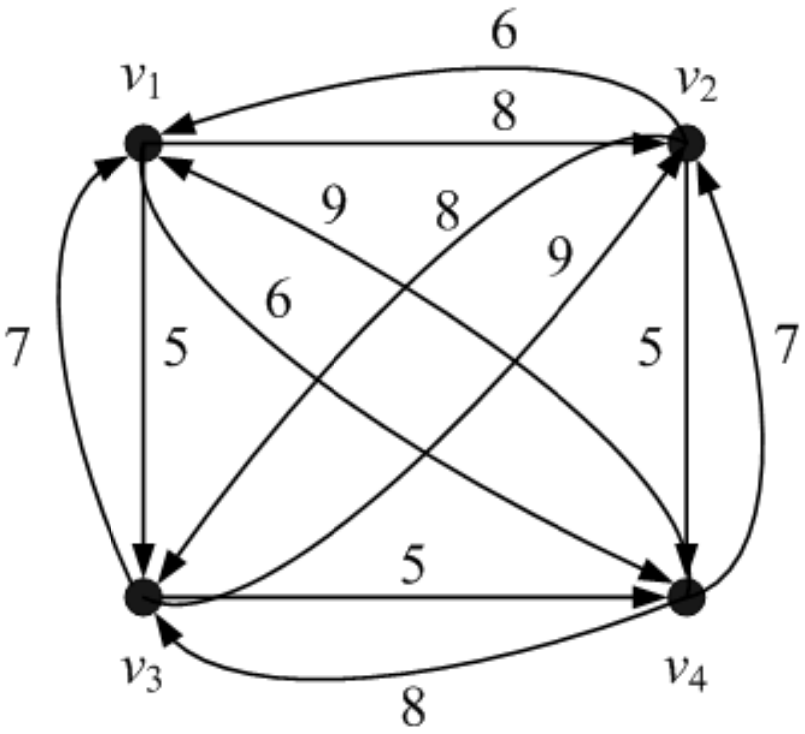
- 如何对问题（按最优化原理）进行分解？
- 由于原问题是找一个圈，从任何一个点开始走这个圈都是可以的。因此不妨假设从 $v_1$ 开始。

- 于是问题就是，从 $v_1$ 开始，经过 $V \setminus \{v_1\}$中的顶点各一次，最后回到 $v_1$，这样的最短路的长度是多少？
- 用 $f(v_i, U, v_1)$表示从顶点 $v_i$ 出发，经过 $U$ 中的顶点各一次，最后到达 $v_1$ 的最短路的长度，其中 $U \subset V$ 是一个顶点子集，满足 $v_1 \notin U$，$v_i \notin U$。则有

$$f(v_i, U, v_1) = \begin{cases} \min\limits_{v_j \in U} \{d(v_i, v_j) + f(v_j, U \setminus \{v_j\}, v_1)\}, & U \neq \phi \\ d(v_i, v_1), & U = \phi \end{cases}。$$

- 原问题就是计算 $f(v_1, V \setminus \{v_1\}, v_1)$。

# 例 5.3.1



| | $v_2$ | $v_3$ | $v_4$ |
|---|---|---|---|
| $\varnothing$ | 6 | 7 | 9 |
| $\{v_2\}$ | × | 15 | 13 |
| $\{v_3\}$ | 15 | × | 15 |
| $\{v_4\}$ | 14 | 14 | × |
| $\{v_2, v_3\}$ | × | × | 22 |
| $\{v_2, v_4\}$ | × | 18 | × |
| $\{v_3, v_4\}$ | 20 | × | × |

| | $v_2$ | $v_3$ | $v_4$ |
|---|---|---|---|
| $\varnothing$ | $v_1$ | $v_1$ | $v_1$ |
| $\{v_2\}$ | × | $v_2, \varnothing$ | $v_2, \varnothing$ |
| $\{v_3\}$ | $v_3, \varnothing$ | × | $v_3, \varnothing$ |
| $\{v_4\}$ | $v_4, \varnothing$ | $v_4, \varnothing$ | × |
| $\{v_2, v_3\}$ | × | × | $v_2, \{v_3\}$ |
| $\{v_2, v_4\}$ | × | $v_4, \{v_2\}$ | × |
| $\{v_3, v_4\}$ | $v_4, \{v_3\}$ | × | × |

- 使用表格计算诸 $f_k(v_i, U, v_1)$: $k = 1..2$; $\forall v_i \neq v_1$; $\forall U \subset V$, 满足 $v_1 \notin U$， $v_i \notin U$。

- 最后单独计算 $f_3(v_1, U = V \setminus \{v_1\}, v_1)$（因为 $v_1 \in U$，故 $f_3(v_1, U, v_1)$没有列入动态规划表格中）。

## 计算一个单元格

$f_k(v_i, U, v_1)$

1 $d \leftarrow \infty$。

2 for each $v_j \in U$ do

3 $t \leftarrow d(v_i, v_j) + \text{table}(v_j, U \setminus \{v_j\})$。

4 if $t < d$ then $d \leftarrow t$。

5 endfor

6 return $d$。

## 计算过程

- $f_1(v_2, \{v_3\}, v_1) = \min\{d(v_2, v_3) + f_0(v_3, \varnothing, v_1)\} = 8 + 7 = 15$。
- $f_1(v_2, \{v_4\}, v_1) = \min\{d(v_2, v_4) + f_0(v_4, \varnothing, v_1)\} = 5 + 9 = 14$。

- $f_1(v_3, \{v_2\}, v_1) = \min\{d(v_3, v_2) + f_0(v_2, \varnothing, v_1)\} = 9 + 6 = 15$。
- $f_1(v_3, \{v_4\}, v_1) = \min\{d(v_3, v_4) + f_0(v_4, \varnothing, v_1)\} = 5 + 9 = 14$。

- $f_1(v_4, \{v_2\}, v_1) = \min\{d(v_4, v_2) + f_0(v_2, \varnothing, v_1)\} = 7 + 6 = 13$。
- $f_1(v_4, \{v_3\}, v_1) = \min\{d(v_4, v_3) + f_0(v_3, \varnothing, v_1)\} = 8 + 7 = 15$。

- $f_2(v_2, \{v_3, v_4\}, v_1)$

  $= \min\{d(v_2, v_3) + f_1(v_3, \{v_4\}, v_1), d(v_2, v_4) + f_1(v_4, \{v_3\}, v_1)\}$

  $= \min\{8 + 14, 5 + 15\} = 20$。
- $f_2(v_3, \{v_2, v_4\}, v_1)$

  $= \min\{d(v_3, v_2) + f_1(v_2, \{v_4\}, v_1), d(v_3, v_4) + f_1(v_4, \{v_2\}, v_1)\}$

  $= \min\{9 + 14, 5 + 13\} = 18$。

- $f_2(v_4, \{v_2, v_3\}, v_1)$

  $= \min\{d(v_4, v_2) + f_1(v_2, \{v_3\}, v_1), d(v_4, v_3) + f_1(v_3, \{v_2\}, v_1)\}$

  $= \min\{7 + 15, 8 + 14\} = 22$。

- 最后一个：$f_3(v_1, \{v_2, v_3, v_4\}, v_1)$

  $= \min\{d(v_1, v_2) + f_2(v_2, \{v_3, v_4\}, v_1),$

  $d(v_1, v_3) + f_2(v_3, \{v_2, v_4\}, v_1),$

  $d(v_1, v_4) + f_2(v_4, \{v_2, v_3\}, v_1)\}$

  $= \min\{8 + 18, 5 + 18, 6 + 22\} = \min\{26, 23, 28\} = 23$。

- 最优 TSP 旅游为：$v_1 \to v_3 \to v_4 \to v_2 \to v_1$。

# 一般情況

| | $v_2$ | $v_3$ | $v_4$ | $\cdots$ | $\cdots$ | $\cdots$ | $v_n$ |
|---|---|---|---|---|---|---|---|
| $\varnothing$ | | | | | | | |
| $\{v_2\}$ | | | | | | | |
| $\{v_3\}$ | | | | | | | |
| $\vdots$ | | | | | | | |
| $\{v_4\}$ | | | | | | | |
| $\{v_2, v_3\}$ | | | | | | | |
| $\{v_2, v_4\}$ | | | | | | | |
| $\vdots$ | | | | | | | |
| $\{v_3, v_4\}$ | | | | | | | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| ⋮ | | | | | | | |
| $\{v_2, v_3, \cdots, v_{n-1}\}$ | | | | | | | |
| $\{v_2, v_3, \cdots, v_n\}$ | | | | | | | |
| ⋮ | | | | | | | |
| $\{v_3, v_4, \cdots, v_n\}$ | | | | | | | |

## 时间复杂度

- 计算过程中的基本运算为加法和比较。考虑加法运算的次数。
- 需要计算的 $f$ 函数：

**1.** $\forall v_i \neq v_1$，$\forall U \subset V$，满足$|U|=1$且$v_1, v_i \notin U$，$f_1(v_i, U, v_1)$。

**2.** $\forall v_i \neq v_1$，$\forall U \subset V$，满足$|U|=2$且$v_1, v_i \notin U$，$f_2(v_i, U, v_1)$。

**3.** …

**4.** $\forall v_i \neq v_1$，$\forall U \subset V$，满足$|U|=n-2$且$v_1, v_i \notin U$，$f_{n-2}(v_i, U, v_1)$。

● 因此，需要计算 $(n-1)\binom{n-2}{1}$ 个 $f_1(v_i, U, v_1)$，

$(n-1)\binom{n-2}{2}$ 个 $f_2(v_i,U,v_1)$，…，$(n-1)\binom{n-2}{n-2}$ 个 $f_{n-2}(v_i,U,v_1)$。

- 计算每个 $f_k(v_i,U,v_1)$需要计算 **k** 次加法，**k = 1 .. n – 1**。
- 因此，计算 $f_1(v_i,U,v_1)$，…，$f_{n-2}(v_i,U,v_1)$，以及最后一个 $f_{n-1}(v_1,U,v_1)$，所需要的加法运算的次数为：

$$T=(n-1)+(n-1)\sum_{k=1}^{n-2}\binom{n-2}{k}k$$ 。

- 当 **x = 1** 时，

$$\sum_{k=1}^{n-2}\binom{n-2}{k}k=\sum_{k=1}^{n-2}\binom{n-2}{k}kx^{k-1}=\left(\sum_{k=1}^{n-2}\binom{n-2}{k}x^{k}\right)'$$

$$=\left(\sum_{k=0}^{n-2}\binom{n-2}{k}x^{k}\right)'=\left((1+x)^{n-2}\right)'=(n-2)(1+x)^{n-3}=(n-2)2^{n-3}$$

。

- 因此，$T=(n-1)+(n-1)(n-2)2^{n-3}$。
- 比较运算的次数与加法运算的次数是同量级的。因此，上述 **TSP** 的动态规划算法的时间复杂度为 $O(n^2 2^{n-3})$。

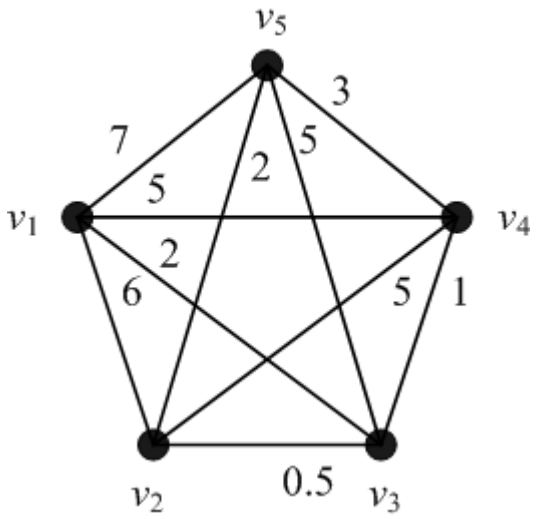
这已经比枚举法的时间复杂度 **O(*n*!)**好了很多。

# 5.4 阶段数不固定的离散变量的优化问题

## 一般图上的最短路问题

已经考察过多阶段图上从 $s$ 到 $t$ 的最短路问题。现在将其推广到一般图上。

- 实例：无向图 $G = (V, E)$，边 $e \in E$ 上定义有长度 $c(e)$。源顶点 $s$，目标顶点 $t$。
- 目标：求从 $s$ 到 $t$ 的最短路。




例 **5.4.1**，$s = v_1$，$t = v_5$

## 第一种解法

- 因为图上一共有 $n$ 个顶点，从 $s$ 到 $t$ 的最短路上边的

数目最多不超过 $n-1$。在允许使用的边数上递推，可用动态规划法解最短 *s-t* 路问题。

- 定义 $f_k(u, t)$为从 $u$ 到 $t$ 经过 $\le k$ 条边的最短路长度。

则有：

$$f_k(u,t)=\begin{cases}0, & u=t\\ \min\limits_{v\in N(u)}\{c(u,v)+f_{k-1}(v,t)\}, & u\neq t,k\geq 1\\ \infty, & u\neq t,k=0\end{cases}$$

。

- 原问题即是求 $f_{n-1}(s, t)$。

# 动态规划表

| | $v_1$ | $v_2$ | $v_3$ | $v_4$ | $v_5$ |
|---|---|---|---|---|---|
| 0 | ∞ | ∞ | ∞ | ∞ | 0 |
| 1 | 7 | 2 | 5 | 3 | 0 |
| 2 | 7 | 2 | 2.5 | 3 | 0 |
| 3 | 4.5 | 2 | 2.5 | 3 | 0 |
| 4 | 4.5 | 2 | 2.5 | 3 | 0 |

| | $v_1$ | $v_2$ | $v_3$ | $v_4$ | $v_5$ |
|---|---|---|---|---|---|
| 0 | × | × | × | × | × |
| 1 | $v_5$ | $v_5$ | $v_5$ | $v_5$ | × |
| 2 | $v_5$ | $v_5$ | $v_2$ | $v_5$ | × |
| 3 | $v_3$ | $v_5$ | $v_2$ | $v_5$ | × |
| 4 | $v_3$ | $v_5$ | $v_2$ | $v_5$ | × |

- 左表：从 $v_i$ 到 $v_5$ 边数不超过 $k$ 的最短路的长度。

  右表：从 $v_i$ 到 $v_5$ 边数不超过 $k$ 的最短路上 $v_i$ 的下一个

顶点。

## 计算过程

- $f_1(v_1, v_5) = c_{15} = 7$，$f_1(v_2, v_5) = c_{25} = 2$，$f_1(v_3, v_5) = c_{35} = 5$，$f_1(v_4, v_5) = c_{45} = 3$。
- $f_2(v_1, v_5) = \min\{7+0,5+3,2+5,6+2\} = 7$，

  $f_2(v_2, v_5) = \min\{6+7,2+0,5+3,0.5+5\} = 2$，

  $f_2(v_3, v_5) = \min\{0.5+2,6+7,5+0,1+3\} = 2.5$，

$f_2(v_4, v_5) = \min\{1+5, 5+2, 5+7, 3+0\} = 3$ 。

- $f_3(v_1, v_5) = \min\{7+0, 5+3, 2+2.5, 6+2\} = 4.5$ ,

  $f_3(v_2, v_5) = \min\{6+7, 2+0, 5+3, 0.5+2.5\} = 2$ ,

  $f_3(v_3, v_5) = \min\{0.5+2, 6+7, 5+0, 1+3\} = 2.5$ ,

  $f_3(v_4, v_5) = \min\{1+2.5, 5+2, 5+7, 3+0\} = 3$ 。
- $f_4(v_1, v_5) = \min\{7+0, 5+3, 2+2.5, 6+2\} = 4.5$ ,

  $f_4(v_2, v_5) = \min\{6+4.5, 2+0, 5+3, 0.5+2.5\} = 2$ ,

  $f_4(v_3, v_5) = \min\{0.5+2, 2+4.5, 5+0, 1+3\} = 2.5$ ,

$f_4(v_4, v_5) = \min\{1+2.5, 5+2, 5+4.5, 3+0\} = 3$ 。

（$f_4(u, v_5)$与 $f_3(u, v_5)$相同，说明在 ***k* = 3** 时已经求到了从诸 $v_i$ 到 $v_5$ 的最短路。）

- 计算完 $f_4(v_1, v_5)$后，只需要继续计算 $f_4(2), \ldots, f_4(n-1)$，就能求到 $v_1, \ldots, v_4$ 各点到 $v_5$ 的最短路。因此导出“单汇点最短路问题”。
- 最后求到的从 $v_1$ 到 $v_5$ 的最短路为 $v_1 - v_3 - v_2 - v_5$，长度为 **4.5**。

# 第二种解法

- 下面要介绍的最短路动态规划算法由[Floyd 1962]和[Warshall 1962]独自提出，也称为 Flody-Warshall 算法。该算法计算出了所有顶点对之间的最短路。
- 将所有顶点从 1 到 $n$ 编号。（假设 $s = v_1$， $t = v_n$。）
- 将费用函数 $c$ 的定义域从 $E$ 扩展到 $V \times V$：

  对于$(v_i, v_j) \in V \times V$，若 $i = j$，则定义 $c(i, j) = 0$；

  否则，若$(v_i, v_j) = e \in E$，则定义 $c(i, j) = c(j, i) = c(e)$；

否则（$v_i$、$v_j$之间没有边），定义 $c(i, j) = c(j, i) = +\infty$。

- $c(i, j)$简记为 $c_{ij}$。

## 递推方程

- 观测：设 $P$ 为从 $v_i$ 到 $v_j$ 的一条最短路，其上最大的顶点编号（不包括 $i$ 和 $j$）为 $k$。则 $P$ 也是从 $v_i$ 到 $v_j$ 中间顶点编号不超过 $k$ 的一条最短路。并且，$P$ 上从 $v_i$ 到 $v_k$ 的路 $P_1$ 也是从 $v_i$ 到 $v_k$ 的中间顶点编号不超过 $k-1$ 的一条最短路；类似地，$P$ 上从 $v_k$ 到 $v_j$ 的路 $P_2$ 也是从

$v_i$到$v_k$的中间顶点编号不超过$k-1$的一条最短路。

- 定义$d_{ij}^{(k)}=d^{(k)}(i,j)$为从$v_i$到$v_j$中间顶点编号不超过$k$的最短路长度。则有：

$$d_{ij}^{(k)}=\begin{cases}c_{ij}, & k=0\\ \min\left\{d_{ij}^{(k-1)}, d_{ik}^{(k-1)}+d_{kj}^{(k-1)}\right\}, & k\geq 1\end{cases}。$$

- 由于最大的顶点编号为 $n$，因此任何一条最短路都是中间顶点编号不超过$n$的一条最短路。原问题即是求$\left\{d_{ij}^{(n)}\right\}$。

# 动态规划表

$$D^{(0)}=\begin{bmatrix}0 & 6 & 2 & 5 & 7\\ 6 & 0 & 0.5 & 5 & 2\\ 2 & 0.5 & 0 & 1 & 5\\ 5 & 5 & 1 & 0 & 3\\ 7 & 2 & 5 & 3 & 0\end{bmatrix},\quad D^{(1)}=\begin{bmatrix}0 & 6 & 2 & 5 & 7\\ 6 & 0 & 0.5 & 5 & 2\\ 2 & 0.5 & 0 & 1 & 5\\ 5 & 5 & 1 & 0 & 3\\ 7 & 2 & 5 & 3 & 0\end{bmatrix}$$

$$D^{(2)} = \begin{bmatrix} 0 & 6 & 2 & 5 & 7 \\ 6 & 0 & 0.5 & 5 & 2 \\ 2 & 0.5 & 0 & 1 & 2.5 \\ 5 & 5 & 1 & 0 & 3 \\ 7 & 2 & 2.5 & 3 & 0 \end{bmatrix} ,$$

$$D^{(3)} = \begin{bmatrix} 0 & 2.5 & 2 & 3 & 4.5 \\ 2.5 & 0 & 0.5 & 1.5 & 2 \\ 2 & 0.5 & 0 & 1 & 2.5 \\ 3 & 1.5 & 1 & 0 & 3 \\ 4.5 & 2 & 2.5 & 3 & 0 \end{bmatrix}$$

$$D^{(4)} = \begin{bmatrix} 0 & 2.5 & 2 & 3 & 4.5 \\ 2.5 & 0 & 0.5 & 1.5 & 2 \\ 2 & 0.5 & 0 & 1 & 2.5 \\ 3 & 1.5 & 1 & 0 & 3 \\ 4.5 & 2 & 2.5 & 3 & 0 \end{bmatrix} ,$$

$$D^{(5)} = \begin{bmatrix} 0 & 2.5 & 2 & 3 & 4.5 \\ 2.5 & 0 & 0.5 & 1.5 & 2 \\ 2 & 0.5 & 0 & 1 & 2.5 \\ 3 & 1.5 & 1 & 0 & 3 \\ 4.5 & 2 & 2.5 & 3 & 0 \end{bmatrix}$$

（$D^{(3)} = D^{(4)} = D^{(5)}$。）

## 单汇点最短路问题

- 实例：无向图 $G = (V, E)$，顶点集合 $V = \{v_1, v_2, \ldots, v_n\}$。

边$(v_i, v_j) \in E$ 上定义有长度 $c_{ij}$。

- 目标：从 $v_1, v_2, \ldots, v_{n-1}$ 各个顶点到顶点 $v_n$ 的最短路？

- 定义 $f(i)$为从顶点 $v_i$ 到顶点 $v_n$ 的距离（最短路的长度）。

则有：$$\begin{cases} f(i) = \min\limits_{1 \le j \le n} \{c_{ij} + f(j)\}, & i = 1..n-1 \\ f(n) = 0, & i = n \end{cases}$$ 。 **(5.4.1)**

- 这是一个函数方程，而不是递推方程。

- 两种解法：函数空间迭代法和策略空间迭代法

## 第一种解法：函数空间迭代法

例 **5.4.1** 之前定义的$\{f_k(i)\}$就是函数方程**(5.4.1)**的一个解。

定理 **5.4.1** 由方程**(5.4.2)**确定的函数列$\{f_k(i)\}$单调下降收敛于函数方程**(5.4.1)**的解$f(i)$。

- 证：$\forall 1 \le i \le n-1$，$\forall k \ge 2$，

  $f_k(i) = \min\limits_{1 \le j \le n}\{c_{ij} + f_{k-1}(j)\} \le c_{ii} + f_{k-1}(i) = f_{k-1}(i)$，因此$\{f_k(i)\}$是单调下降序列。

- 因为诸 $c_{ij} \geq 0$，$f_k(i) \geq \min\{c_{ij}\} = 0$，有下界。所以 $\{f_k(i)\}$ 有极限，设极限为 $f(i)$。
- 由于 $f(i)$ 的定义域仅有有限个 ***i***，因此对任意 $\varepsilon > 0$，存在正整数 $k_0$，当 $k \geq k_0$ 时，对所有的 ***i***，都有 $|f_k(i) - f(i)| < \varepsilon$。
- 因此有 $-\varepsilon + f_k(i) < f(i) < \varepsilon + f_k(i)$，以及 $-\varepsilon + f_{k+1}(i) < f(i) < \varepsilon + f_{k+1}(i)$，其中 $k \geq k_0$。
- 因此，$f(i) < \varepsilon + f_{k+1}(i) = \varepsilon + \min_j \{c_{ij} + f_k(j)\}$

$$< \varepsilon + \min_j \{c_{ij} + f(j) + \varepsilon\} = \min_j \{c_{ij} + f(j)\} + 2\varepsilon \text{，} \quad \textbf{(1)}$$

以及，$f(i) > f_{k+1}(i) - \varepsilon = \min_j \{c_{ij} + f_k(j)\} - \varepsilon$

$$> \min_j \{c_{ij} + f(j) - \varepsilon\} - \varepsilon = \min_j \{c_{ij} + f(j)\} - 2\varepsilon \text{。} \quad \textbf{(2)}$$

- 当 $\varepsilon \to 0$ 时，**(1)**、**(2)**都 $\to \min_j \{c_{ij} + f(j)\}$。因此，$f(i) = \min_j \{c_{ij} + f(j)\}$。□

# 第二种解法：策略空间迭代法

- 称 $s:V\setminus\{v_n\}\mapsto V$ 为一个策略，其中 $s(i)$ 表示对于单汇点（$v_n$）最短路问题，从 $v_i$ 到 $v_n$ 的路上顶点 $v_i$ 的下一个顶点（即后继），$i=1..n-1$。
- $\{s(i)\}$ 需要是一个无回路策略，即，从任一个顶点 $v_i$ 出发，经过任意次 $s(i)$ 后继操作，不能再回到 $v_i$。特别地，$s(i)\neq i$。
- 由于 $s$ 是 $V\setminus\{v_n\}$ 到 $V$ 的函数，且 $s$ 是无回路的，以 $V$

为顶点集，以$\{(i,s(i))\,|\,i=1..n-1\}$为边集，必构成一棵以 $v_n$ 为根的树。

- 给定一个策略$\{s(i)\}$，解方程组

$$\begin{cases} f_k(i)=c(i,s_k(i))+f_k(s_k(i)), & i=1..n-1 \\ f_k(n)=0, & i=n \end{cases},$$

得到 $f_k(1), f_k(2),\cdots, f_k(n)$，其中 $f_k(i)$ 表示在策略 $s_k$ 之下，从顶点 $v_i$ 到顶点 $v_n$ 的路的长度。

- 令 $s_{k+1}(i)=\arg\min_{1\le j\le n,\, j\ne i}\{c(i,j)+f_k(j)\}$，$i=1..n-1$，构造

下一个策略 $s_{k+1}$。

- 从一个初始策略 $s_0$ 开始，重复上述过程，直到相邻的两个策略 $s_{k-1}$ 和 $s_k$ 完全相同，则 $s_k$ 就是最优解，此时 $f_k$ 是函数方程 **(5.4.1)** 的解。

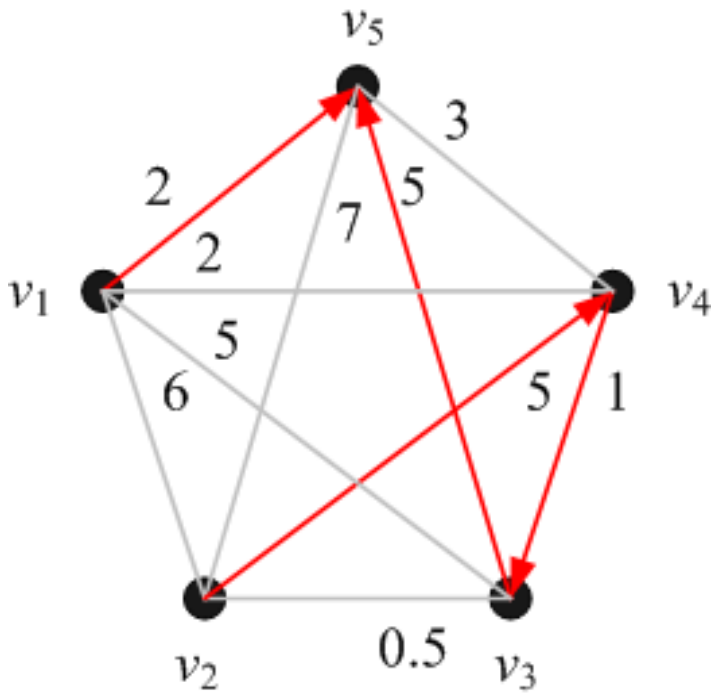
## 例 5.4.1（2）

**例 5.4.1(2)** 用策略空间迭代法重解最短路问题。

- 初始策略 $s_0$ 为：

| $s_0(1)$ | $s_0(2)$ | $s_0(3)$ | $s_0(4)$ |
|---|---|---|---|

| | | | |
|---|---|---|---|
| 5 | 4 | 5 | 3 |




- 解方程组

$$\begin{cases} f_0(1) = c(1,5) + f_0(5) = 2 + f_0(5) \\ f_0(2) = c(2,4) + f_0(4) = 5 + f_0(4) \\ f_0(3) = c(3,5) + f_0(5) = 5 + f_0(5) \\ f_0(4) = c(4,3) + f_0(3) = 1 + f_0(3) \\ f_0(5) = 0 \end{cases}$$ ，得：

| $f_0(1)$ | $f_0(2)$ | $f_0(3)$ | $f_0(4)$ | $f_0(5)$ |
|---|---|---|---|---|
| 2 | 11 | 5 | 6 | 0 |

- 构造策略 $s_1$：

  $s_1(1) = \text{argmin}\{---, c_{12} + f_0(2), c_{13} + f_0(3), c_{14} + f_0(4), c_{15} + f_0(5)\}$

= argmin{---, 6 + 11, 5 + 5, 2 + 6, 2 + 0} = 5。

$s_1(2)$ = argmin{ $c_{21} + f_0(1)$, ---, $c_{23} + f_0(3)$, $c_{24} + f_0(4)$, $c_{25} + f_0(5)$}

= argmin{6 + 2, ---, 0.5 + 5, 5 + 6, 7 + 0} = 3。

$s_1(3)$ = argmin{ $c_{31} + f_0(1)$, $c_{32} + f_0(2)$, ---, $c_{34} + f_0(4)$, $c_{35} + f_0(5)$}

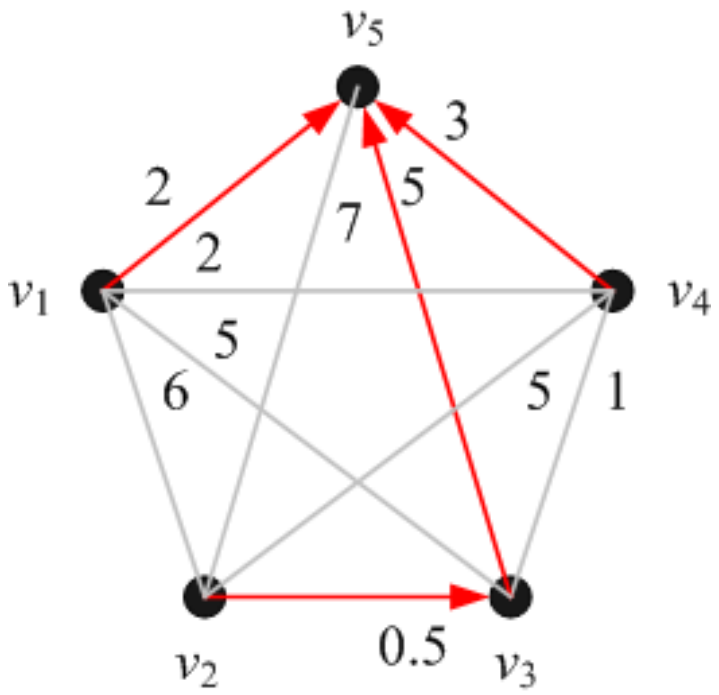
= argmin{5 + 2, 0.5 + 11, ---, 1 + 6, 5 + 0} = 5。

$s_1(4)$ = argmin{ $c_{41} + f_0(1)$, $c_{42} + f_0(2)$, $c_{43} + f_0(3)$, ---, $c_{45} + f_0(5)$}

= argmin{2 + 2, 5 + 11, 1 + 5, ---, 3 + 0} = 5。

- 策略 $s_1$ 为：

| $s_1(1)$ | $s_1(2)$ | $s_1(3)$ | $s_1(4)$ |
|---|---|---|---|
| 5 | 3 | 5 | 5 |




- 解方程组

$$\begin{cases} f_1(1) = c(1,5) + f_1(5) = 2 + f_1(5) \\ f_1(2) = c(2,3) + f_1(3) = 0.5 + f_1(3) \\ f_1(3) = c(3,5) + f_1(5) = 5 + f_1(5) \\ f_1(4) = c(4,5) + f_1(5) = 3 + f_1(5) \\ f_1(5) = 0 \end{cases}$$ ，得：

| $f_1(1)$ | $f_1(2)$ | $f_1(3)$ | $f_1(4)$ | $f_1(5)$ |
|---|---|---|---|---|
| 2 | 5.5 | 5 | 3 | 0 |

- 构造策略 $s_2$：

  $s_2(1) = \text{argmin}\{---, c_{12} + f_1(2), c_{13} + f_1(3), c_{14} + f_1(4), c_{15} +$

$f_1(5)\}$

$= \text{argmin}\{---, 6 + 5.5, 5 + 5, 2 + 3, 2 + 0\} = 5$。

$s_2(2) = \text{argmin}\{ c_{21} + f_1(1), ---, c_{23} + f_1(3), c_{24} + f_1(4), c_{25} + f_1(5)\}$

$= \text{argmin}\{6 + 2, ---, 0.5 + 5, 5 + 3, 7 + 0\} = 3$。

$s_2(3) = \text{argmin}\{ c_{31} + f_1(1), c_{32} + f_1(2), ---, c_{34} + f_1(4), c_{35} + f_1(5)\}$

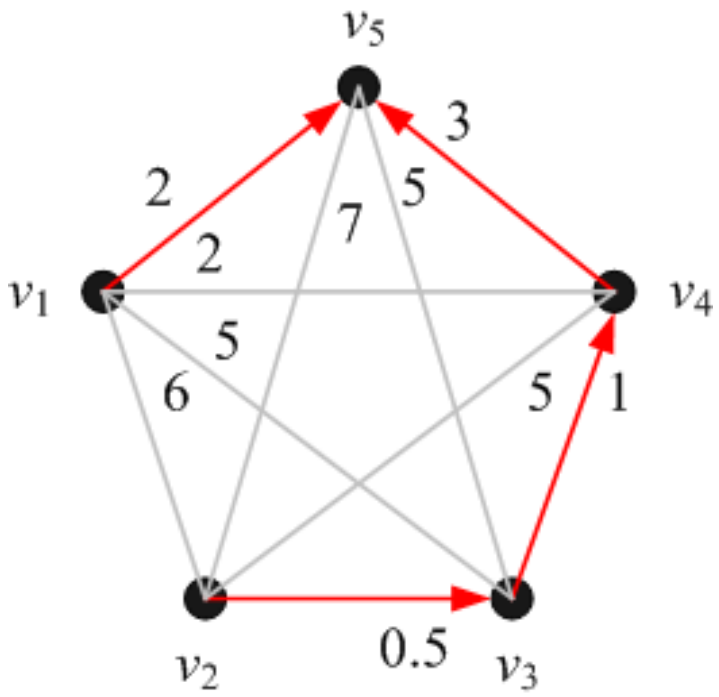
$= \text{argmin}\{5 + 2, 0.5 + 5.5, ---, 1 + 3, 5 + 0\} = 4$。

$s_2(4) = \text{argmin}\{ c_{41} + f_1(1), c_{42} + f_1(2), c_{43} + f_1(3), ---, c_{45} + f_1(5)\}$

**= argmin{2 + 2, 5 + 5.5, 1 + 5, ---, 3 + 0} = 5。**

- 策略 $s_2$ 为：

| $s_1(1)$ | $s_1(2)$ | $s_1(3)$ | $s_1(4)$ |
|---|---|---|---|
| 5 | 3 | 4 | 5 |




- 解方程组

$$\begin{cases} f_2(1)=c(1,5)+f_2(5)=2+f_2(5) \\ f_2(2)=c(2,3)+f_2(3)=0.5+f_2(3) \\ f_2(3)=c(3,4)+f_2(4)=1+f_2(4) \\ f_2(4)=c(4,5)+f_2(5)=3+f_2(5) \\ f_2(5)=0 \end{cases}$$ ，得：

| $f_2(1)$ | $f_2(2)$ | $f_2(3)$ | $f_2(4)$ | $f_2(5)$ |
|---|---|---|---|---|
| 2 | 4.5 | 4 | 3 | 0 |

- 构造策略 $s_3$：

  $s_3(1) = \text{argmin}\{---, c_{12} + f_2(2), c_{13} + f_2(3), c_{14} + f_2(4), c_{15} +$

$f_2(5)\}$

$= \text{argmin}\{\text{---}, 6 + 4.5, 5 + 4, 2 + 3, 2 + 0\} = 5$。

$s_3(2) = \text{argmin}\{ c_{21} + f_2(1), \text{---}, c_{23} + f_2(3), c_{24} + f_2(4), c_{25} + f_2(5)\}$

$= \text{argmin}\{6 + 2, \text{---}, 0.5 + 4, 5 + 3, 7 + 0\} = 3$。

$s_3(3) = \text{argmin}\{ c_{31} + f_2(1), c_{32} + f_2(2), \text{---}, c_{34} + f_2(4), c_{35} + f_2(5)\}$

$= \text{argmin}\{5 + 2, 0.5 + 4.5, \text{---}, 1 + 3, 5 + 0\} = 4$。

$s_3(4) = \text{argmin}\{ c_{41} + f_2(1), c_{42} + f_2(2), c_{43} + f_2(3), \text{---}, c_{45} + f_2(5)\}$

= argmin{2 + 2, 5 + 4.5, 1 + 4, ---, 3 + 0} = 5。

- 策略 $s_3$ 与 $s_2$ 相同，计算结束。□
- 动态规划之策略空间迭代法计算过程：

| $s$ | $v_1$ | $v_2$ | $v_3$ | $v_4$ |
|---|---|---|---|---|
| 0 | 5 | 4 | 5 | 3 |
| 1 | 5 | 3 | 5 | 5 |
| 2 | 5 | 3 | 4 | 5 |
| 3 | 5 | 3 | 4 | 5 |

| $f$ | $v_1$ | $v_2$ | $v_3$ | $v_4$ | $v_5$ |
|---|---|---|---|---|---|
| 0 | 2 | 11 | 5 | 6 | 0 |
| 1 | 2 | 5.5 | 5 | 3 | 0 |
| 2 | 2 | 4.5 | 4 | 3 | 0 |
| 3 | 2 | 4.5 | 4 | 3 | 0 |

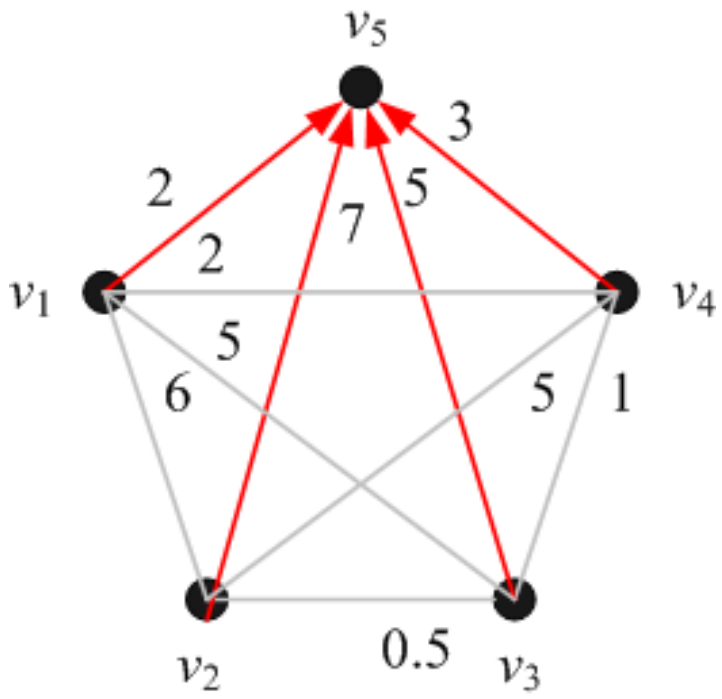
# 例 5.4.1（3）

**例 5.4.1(3)** 用策略空间迭代法重解最短路问题。

- 初始策略 $s_0$ 为：

| $s_0(1)$ | $s_0(2)$ | $s_0(3)$ | $s_0(4)$ |
|---|---|---|---|
| 5 | 5 | 5 | 5 |

| $f_0(1)$ | $f_0(2)$ | $f_0(3)$ | $f_0(4)$ | $f_0(5)$ |
|---|---|---|---|---|
| 2 | 7 | 5 | 3 | 0 |




- 构造策略 $s_1$：

  $s_1(1) = \text{argmin}\{---, c_{12} + f_0(2), c_{13} + f_0(3), c_{14} + f_0(4), c_{15} +$

$f_0(5)\}$

$= \text{argmin}\{\text{---}, 6 + 7, 5 + 5, 2 + 3, 2 + 0\} = 5$。

$s_1(2) = \text{argmin}\{ c_{21} + f_0(1), \text{---}, c_{23} + f_0(3), c_{24} + f_0(4), c_{25} + f_0(5)\}$

$= \text{argmin}\{6 + 2, \text{---}, 0.5 + 5, 5 + 3, 7 + 0\} = 3$。

$s_1(3) = \text{argmin}\{ c_{31} + f_0(1), c_{32} + f_0(2), \text{---}, c_{34} + f_0(4), c_{35} + f_0(5)\}$

$= \text{argmin}\{5 + 2, 0.5 + 7, \text{---}, 1 + 3, 5 + 0\} = 4$。

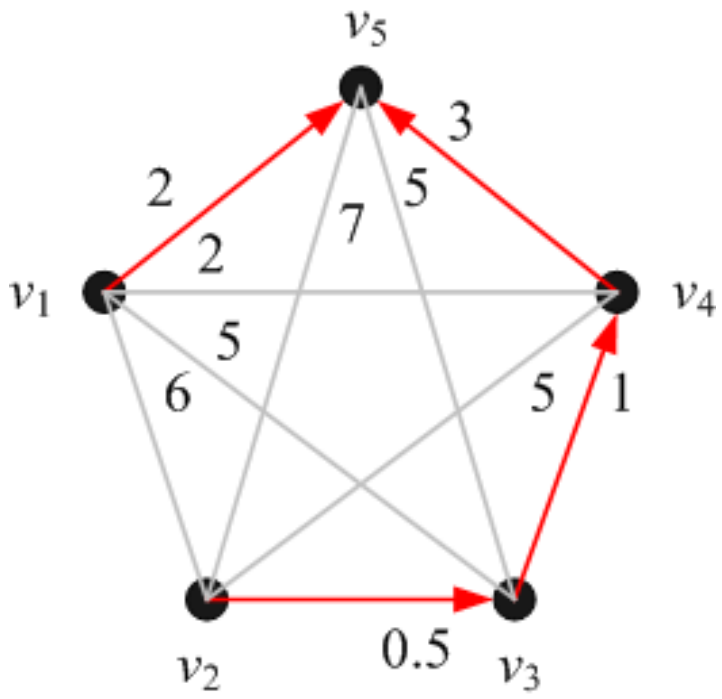
$s_1(4) = \text{argmin}\{ c_{41} + f_0(1), c_{42} + f_0(2), c_{43} + f_0(3), \text{---}, c_{45} + f_0(5)\}$

**= argmin{2 + 2, 5 + 7, 1 + 5, ---, 3 + 0} = 5。**

- 策略 $s_1$ 为：

| $s_1(1)$ | $s_1(2)$ | $s_1(3)$ | $s_1(4)$ |
|---|---|---|---|
| 5 | 3 | 4 | 5 |

| $f_0(1)$ | $f_0(2)$ | $f_0(3)$ | $f_0(4)$ | $f_0(5)$ |
|---|---|---|---|---|
| 2 | 4.5 | 4 | 3 | 0 |




- 再计算策略 $s_2$，将与 $s_1$ 相同，求到最优解。
- 动态规划之策略空间迭代法计算过程：

| $s$ | $v_1$ | $v_2$ | $v_3$ | $v_4$ |
|---|---|---|---|---|
| 0 | 5 | 5 | 5 | 5 |
| 1 | 5 | 3 | 4 | 5 |
| 2 | 5 | 3 | 4 | 5 |

| $f$ | $v_1$ | $v_2$ | $v_3$ | $v_4$ | $v_5$ |
|---|---|---|---|---|---|
| 0 | 2 | 7 | 5 | 3 | 0 |
| 1 | 2 | 4.5 | 4 | 3 | 0 |
| 2 | 2 | 4.5 | 4 | 3 | 0 |

# 5.5 阶段数固定的连续变量的优化问题

## 连续变量的资源分配问题

- 下面讨论有限资源分配问题，它的递推公式是：

$$f_k(x)=\begin{cases}\max\limits_{0\le y\le x}\{g(y)+h(x-y)+f_{k-1}(ay+b(x-y))\}, & k\ge 2\\ \max\limits_{0\le y\le x}\{g(y)+h(x-y)\}, & k=1\end{cases}$$

- 当 ***g(y)***、***h(y)***是一般函数时，这个问题的解不容易找。（但对于离散变量的该问题，有动态规划的办法求到算法解……☺）

- 下面证明，当 $g(y)$、$h(y)$为凸函数，且 $h(0) = g(0) = 0$ 时，在每个阶段上 $y$ 的最优决策总是取其端点的值。即上述递推公式可以化简为：

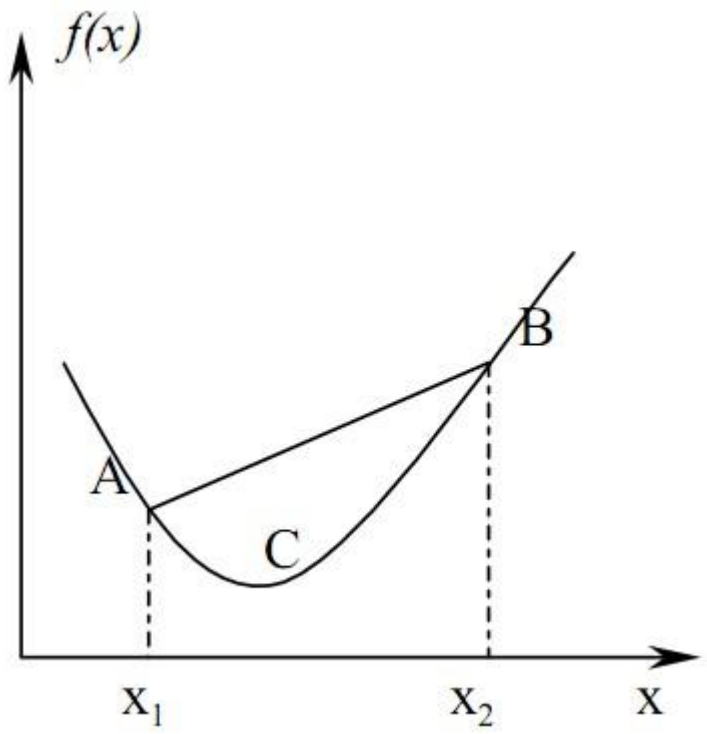
$$\begin{cases} f_k(x) = \max\{h(x) + f_{k-1}(bx), g(x) + f_{k-1}(ax)\}, & k \geq 2 \\ f_1(x) = \max\{h(x), g(x)\}, & k = 1 \end{cases}$$

## 凸函数（第 4.2 节）

定义 **4.2.1** 设有函数 $f: R \to R$。若对任意的$\alpha \in (0,1)$，$x_1 \in R$，$x_2 \in R$，都有

$$\alpha f(x_1) + (1-\alpha) f(x_2) \geq f(\alpha x_1 + (1-\alpha) x_2)$$

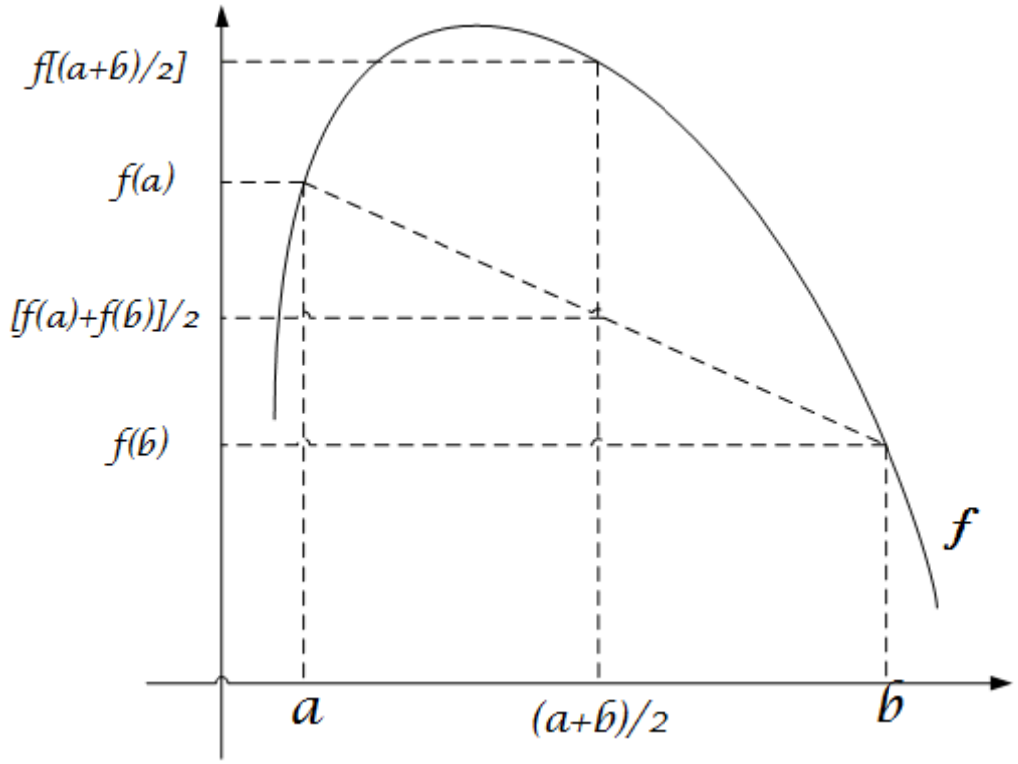
则称 $f$ 是凸函数。并且，若上述不等式以严格不等式成立，则称 $f$ 是严格凸函数。




若对任意的$\alpha \in (0,1)$，$x_1 \in \mathbf{R}$，$x_2 \in \mathbf{R}$，都有

$$\alpha f(x_1) + (1-\alpha) f(x_2) \leq f(\alpha x_1 + (1-\alpha) x_2)$$

则称 $f$ 是凹函数。并且，若上述不等式以严格不等式成立，则称 $f$ 是严格凹函数。


f[(a+b)/2]
f(a)
[f(a)+f(b)]/2
f(b)
f
a
(a+b)/2
b

## 两个引理

引理 5.3.1 设 $g(x)$、$h(y)$是凸函数，则对任何固定的 $x$，$f(y) = g(y) + h(x - y)$是 $y$ 的凸函数。

- 证明：由于两个凸函数的和仍然是凸函数（自证），只需证 $h(x - y)$是 $y$ 的凸函数。
- 令 $h'(y) = h(x - y)$。按照定义，需证：$\forall 0 \le \alpha \le 1$，$\forall y_1$，$\forall y_2$，

$$\alpha h'(y_1) + (1 - \alpha)h'(y_2) \ge h'(\alpha y_1 + (1 - \alpha)y_2)。$$

即，$\forall 0 \le \alpha \le 1$，$\forall y_1$，$\forall y_2$，

$\alpha h(x - y_1) + (1 - \alpha)h(x - y_2) \ge h(x - (\alpha y_1 + (1 - \alpha)y_2))$。

- 而这是成立的，因为 $h(x - (\alpha y_1 + (1 - \alpha)y_2)) = h(\alpha(x - y_1) + (1 - \alpha)(x - y_2)) \le \alpha h(x - y_1) + (1 - \alpha)h(x - y_2)$，其中最后一步是因为 $h(y)$ 是 $y$ 的凸函数。□

**引理 5.3.2**　设 $F_1(x)$、$F_2(x)$ 是 $x$ 的凸函数，则 $F(x) = \max\{F_1(x), F_2(x)\}$ 也是 $x$ 的凸函数。□

## 主要定理

**定理 5.3.1** 设 $g(x)$、$h(y)$是凸函数，$g(0) = h(0) = 0$。则 $n$ 阶段资源分配问题每个阶段的最优策略 $y$ 总是在 $0 \le y \le x$ 的端点处取得。

- 证明：先证 $k = 1$ 的情形。
- $f_1(x) = \max_{0 \le y \le x} \{g(y) + h(x - y)\}$。由引理 5.3.1，$g(y) + h(x - y)$是 $y$ 的凸函数，因此$g(y) + h(x - y)$在区间$[0, x]$上的最大值必定在 $y = 0$ 或 $y = x$ 处取得。即，

$f_1(x)=\max\{g(x),h(x)\}$。

- 再用归纳法证 ***k*** **≥ 2** 的情形。
- （基本步）

  $f_2(x)=\max_{0\le y\le x}\{g(y)+h(x-y)+f_1(ay+b(x-y))\}$。由引理 **5.3.2**，$f_1(x)$是 ***x*** 的凸函数。由于$ay+b(x-y)$是 ***y*** 的线性函数，可证 $f_1(ay+b(x-y))$是 ***y*** 的凸函数（类似于引理 **5.3.1** 中对 ***h()***函数的证明）。
- 由于$g(y)$、$h(x-y)$、$f_1(ay+b(x-y))$均是 ***y*** 的凸函数，

它们的和也是 $y$ 的凸函数。因此

$g(y)+h(x-y)+f_1(ay+b(x-y))$ 在区间**[0, $x$]**上的最大值必定在其中一个端点处取得。即，

$f_2(x)=\max\{g(x)+f_1(ax), h(x)+f_1(bx)\}$。由引理 **5.3.2**，$f_2(x)$是 $x$ 的凸函数。

- （假设步）假设

  $f_{k-1}(x)=\max\{g(x)+f_{k-2}(ax), h(x)+f_{k-2}(bx)\}$

  （$k \geq 3$），并且是凸函数。
- （归纳步）下面证

$f_k(x)=\max\{g(x)+f_{k-1}(ax),h(x)+f_{k-1}(bx)\}$，并且是凸函数。

- 由定义，

  $f_k(x)=\max\limits_{0\le y\le x}\{g(y)+h(x-y)+f_{k-1}(ay+b(x-y))\}$。由于 $g(y)$、$h(x-y)$、$f_{k-1}(ay+b(x-y))$均是 ***y*** 的凸函数，因此 $g(y)+h(x-y)+f_{k-1}(ay+b(x-y))$在区间**[0, *x*]**上的最大值必在端点处取得。即，

  $f_k(x)=\max\{g(x)+f_{k-1}(ax),h(x)+f_{k-1}(bx)\}$。

- 由于 $h(x)+f_{k-1}(bx)$、$g(x)+f_{k-1}(ax)$ 是 ***x*** 的凸函数，由引理 **5.3.2**，$f_k(x)$ 也是 ***x*** 的凸函数。□

## 定理 5.3.1 的应用

- 当 ***g*(*x*)**、***h*(*x*)**满足给定条件时，应用定理 **5.3.1**，对于离散变量的有限阶段资源分配问题，动态规划表的每个单元格的计算可由计算 ***x* + 1** 个值（$f_k(x)=\max\limits_{0\le y\le x}\{h(x)+f_{k-1}(bx),g(x)+f_{k-1}(ax)\}$）简化至只

计算 **2** 个值

（ $f_k(x)=\max\{g(x)+f_{k-1}(ax),h(x)+f_{k-1}(bx)\}$ ）。

- 当 ***g*(*x*)**、***h*(*x*)**是给定的满足定理 **5.3.1** 的显函数时，可利用函数本身的性质对计算做进一步简化。例如，对连续变量的有限阶段资源分配问题可给出解析解。

## 例 5.1.2

例 **5.1.2**：连续变量的资源分配问题。

- 今有 **1000** 吨油（***x* = 1000**），投放到 ***A***、***B*** 两个部

门。

- 若给部门 $A$ 投放 $z$ 吨油，则产生效益 $g(z) = z^2$，回收 $0.8z$ 吨油（$a = 0.8$）。
- 若给部门 $B$ 投放 $z$ 吨油，则产生效益 $h(z) = 2z^2$，回收 $0.4z$ 吨油（$b = 0.4$）。
- 问连续投放 5 年（$n = 5$），每年如何投放，可使 5 年的总收益最大？

解：

- $g(x) = x^2$，$h(x) = 2x^2$，显然 $g(x)$ 和 $h(x)$ 都是凸函数

且 $g(x)=h(x)=0$，满足定理 **5.3.1** 的条件。因此，

- $f_1(x)=\max\{g(x),h(x)\}=\max\{x^2,2x^2\}=2x^2$，

$\Rightarrow y_5=0$。

- $f_2(x)=\max\{g(x)+f_1(ax),h(x)+f_1(bx)\}$

$$=\max\{x^2+(2a^2x^2),2x^2+(2b^2x^2)\}$$

$$=\max\{(1+2a^2)x^2,(2+2b^2)x^2\}$$

**2.28** **2.32**

$$=(2+2b^2)x^2,$$

$\Rightarrow y_4 = 0$ 。

- $f_3(x) = \max\{g(x) + f_2(ax), h(x) + f_2(bx)\}$

$$= \max\{x^2 + (2+2b^2)a^2x^2, 2x^2 + (2+2b^2)b^2x^2\}$$

$$= \max\{(1+2a^2+2a^2b^2)x^2, (2+2b^2+2b^4)x^2\}$$

**2.4848** **2.3712**

$$= (1+2a^2+2a^2b^2)x^2,$$

$\Rightarrow y_3 = x_3$ 。

- $f_4(x) = \max\{g(x) + f_3(ax), h(x) + f_3(bx)\}$

$$= \max\begin{Bmatrix} x^2 + (1 + 2a^2 + 2a^2b^2)a^2x^2, \\ 2x^2 + (1 + 2a^2 + 2a^2b^2)b^2x^2 \end{Bmatrix}$$

$$= \max\begin{Bmatrix} (1 + a^2 + 2a^4 + 2a^4b^2)x^2, \\ (2 + b^2 + 2a^2b^2 + 2a^2b^4)x^2 \end{Bmatrix}$$

**2.590272** **2.397568**

$$= (1 + a^2 + 2a^4 + 2a^4b^2)x^2 ,$$

$\Rightarrow \quad y_2 = x_2$ 。

- $f_5(x) = \max\{g(x) + f_4(ax), h(x) + f_4(bx)\}$

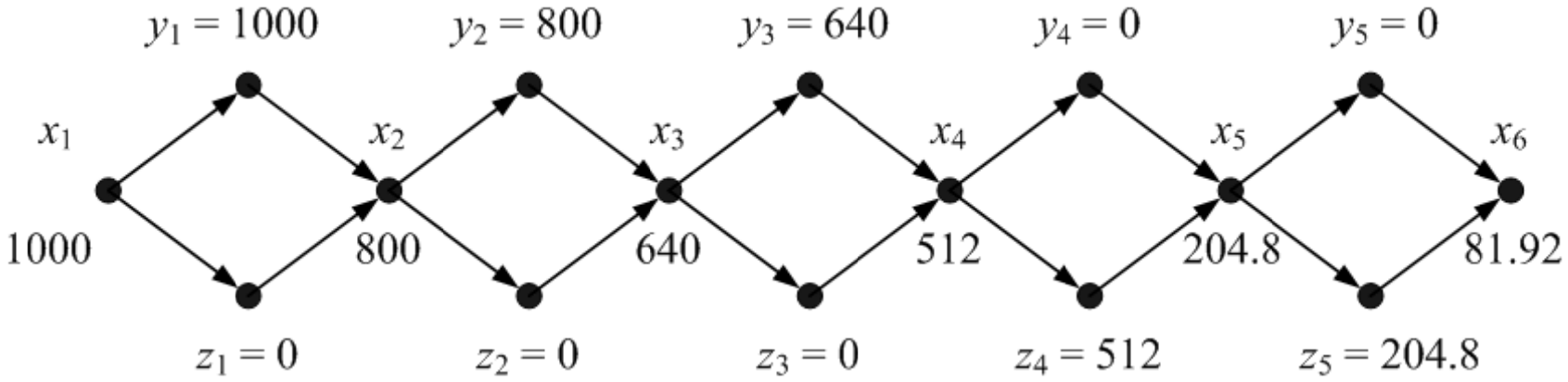
$$= \max\begin{Bmatrix} x^2 + \left(1 + a^2 + 2a^4 + 2a^4b^2\right)a^2x^2, \\ 2x^2 + \left(1 + a^2 + 2a^4 + 2a^4b^2\right)b^2x^2 \end{Bmatrix}$$

$$= \max\begin{Bmatrix} \left(1 + a^2 + a^4 + 2a^6 + 2a^6b^2\right)x^2, \\ \left(2 + b^2 + a^2b^2 + 2a^4b^2 + 2a^4b^4\right)x^2 \end{Bmatrix}$$

（**2.65777408，2.41444352**）

$$= \left(1 + a^2 + a^4 + 2a^6 + 2a^6b^2\right)x^2$$

$\Rightarrow \quad y_1 = x_1$。




- $i_1 = 1000^2 = 1000000$， $i_2 = 800^2 = 640000$，

  $i_3 = 640^2 = 409600$， $i_4 = 2 \times 512^2 = 524288$，

  $i_5 = 2 \times 204.8^2 = 83886.08$。

- 总收益 **= 2657774.08**，即 **$f_5$(1000)**的值。

## 例 5.3.2

**例 5.3.2** 多阶段有限资源分配问题中，$g(x)=-2cx+x^2$，$h(x)=-cx+x^2$，$0\le x\le c$，$0<a,b<1$且$0<b-a\le 1-b$。求 $f_k(x)$。

**解：**

- ***g*(*x*)、*h*(*x*)都是凸函数，且 *g*(0) = *h*(0) = 0，满足定理 5.3.1 的条件。**
- $f_1(x)=\max\{-2cx+x^2,-cx+x^2\}=-cx+x^2$。

- $f_2(x)=\max\{g(x)+f_1(ax),h(x)+f_1(bx)\}$

$$=\max\{-2cx+x^2-cax+a^2x^2,-cx+x^2-cbx+b^2x^2\}$$
$$=\max\{-c(2+a)x+(1+a^2)x^2,-c(1+b)x+(1+b^2)x^2\}$$
$$=-c(1+b)x+(1+b^2)x^2,$$

因为$1+b^2>1+a^2$，及$2+a\geq 1+b$（即，需证$b-a\leq 1$，由已知可得）。

- $f_3(x)=\max\{g(x)+f_2(ax),h(x)+f_2(bx)\}$

$$= \max\begin{Bmatrix} -2cx + x^2 - c(1+b)ax + (1+b^2)a^2x^2, \\ -cx + x^2 - c(1+b)bx + (1+b^2)b^2x^2 \end{Bmatrix}$$

$$= \max\begin{Bmatrix} -c(2+a+ab)x + (1+a^2+a^2b^2)x^2, \\ -c(1+b+b^2)x + (1+b^2+b^4)x^2 \end{Bmatrix}$$

$$= -c(1+b+b^2)x + (1+b^2+b^4)x^2,$$

因为$1+b^2+b^4 > 1+a^2+a^2b^2$，以及

$2+a+ab \geq 1+b+b^2$（即，需证$b+b^2-a-ab \leq 1$，需

证 $b(1+b)-a(1+b)\le 1$，需证 $b-a\le\dfrac{1}{1+b}$，由已知确实有 $b-a\le 1-b<\dfrac{1}{1+b}$）。

- 归纳可知，

$$f_k(x)=-c\left(1+b+\cdots+b^{k-1}\right)x+\left(1+b^2+\cdots+b^{2(k-1)}\right)x^2$$

$$=-\frac{1-b^k}{1-b}cx+\frac{1-b^{2k}}{1-b^2}x^2\ 。$$

（对一般项 $f_k(x)$，需证 $b-a\le\dfrac{1}{1+b+\cdots+b^{k-2}}$，而由

已知确实有$b-a\le 1-b<\frac{1}{1+b+\cdots+b^{k-2}}$）。□