运筹学

8. 动态规划

李 力
清华大学

Email: li-li@tsinghua.edu.cn

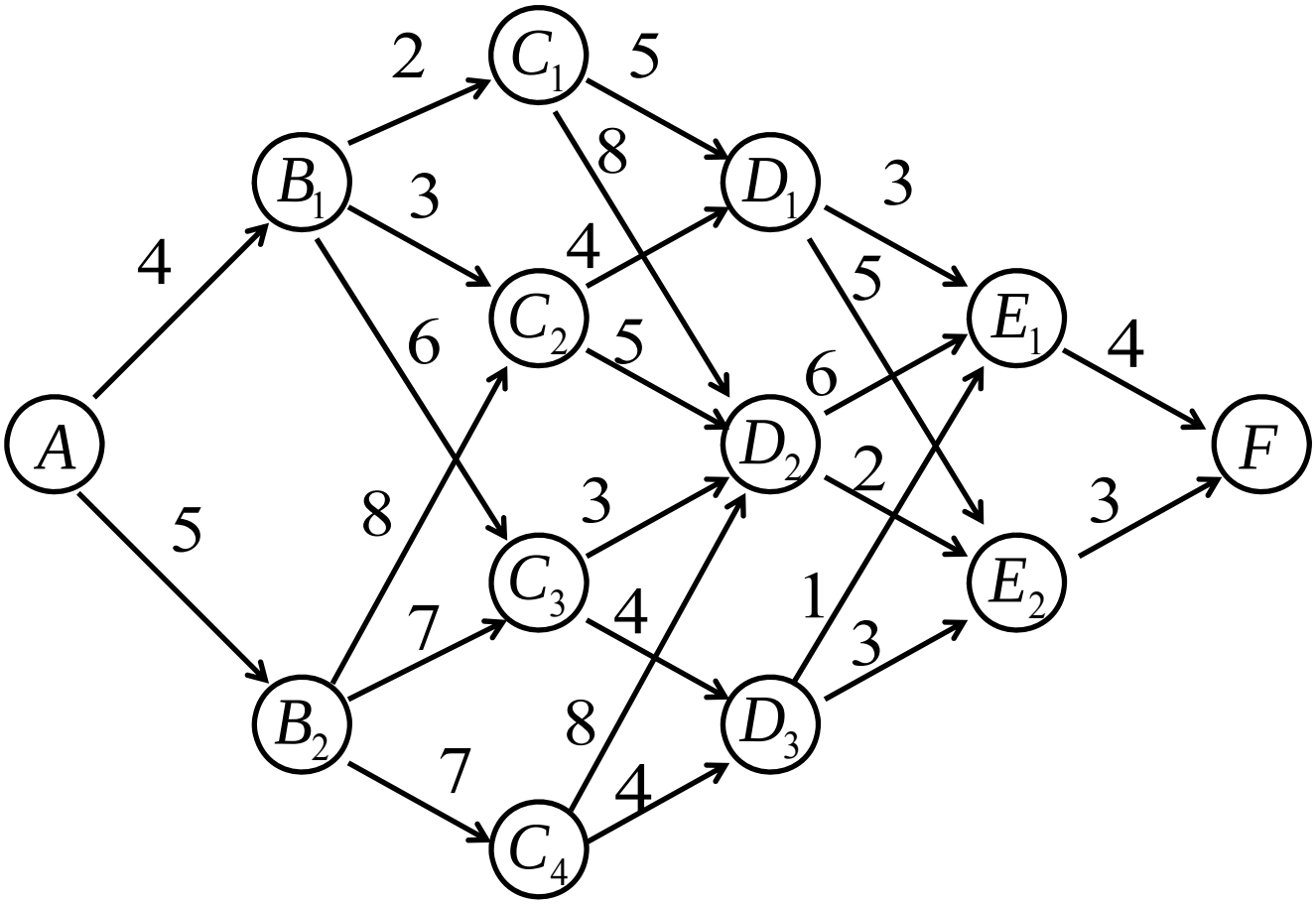
2023.11.

主要内容

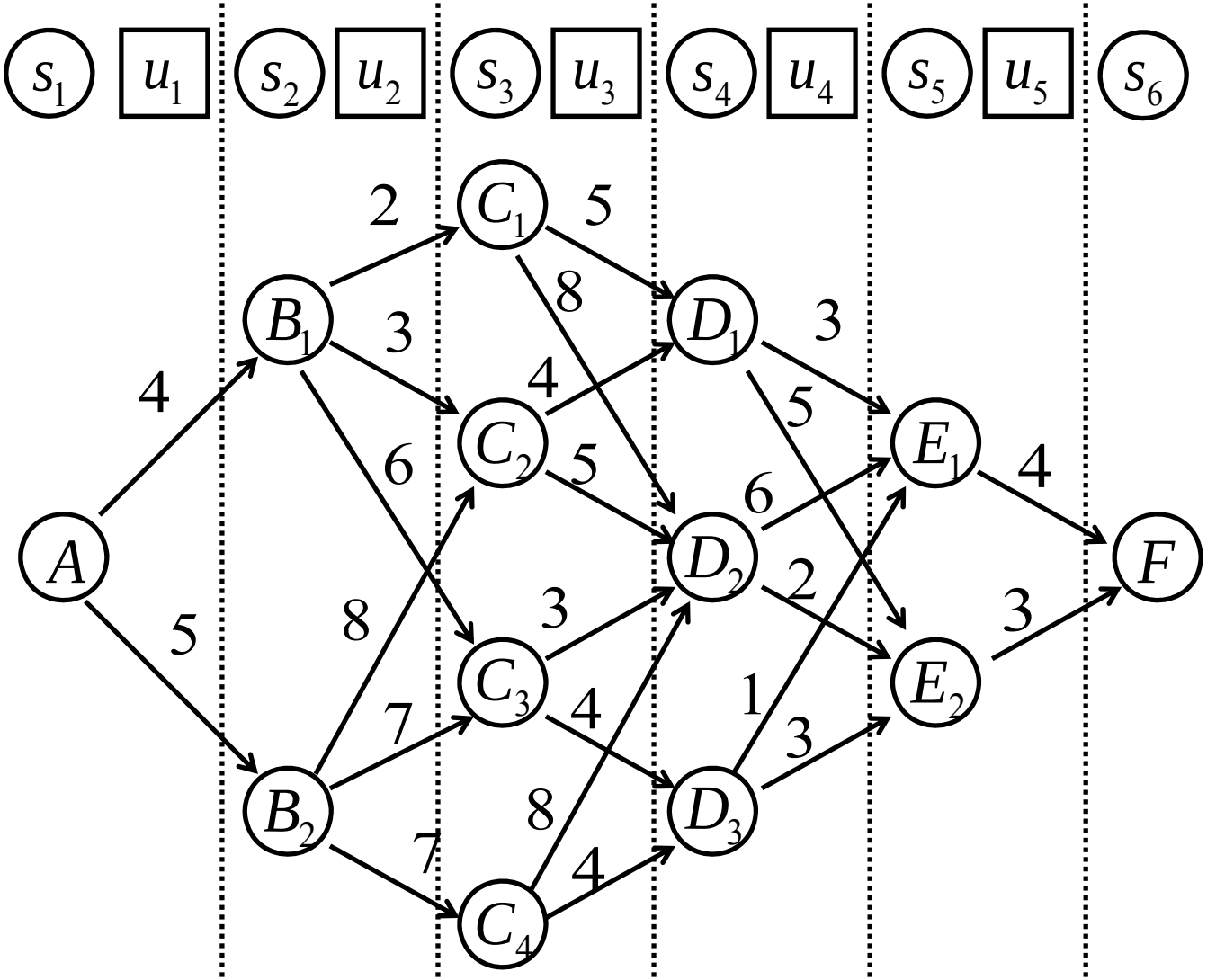
- 最短路问题
- 一般性问题
- 最优性原理
- 建模与求解
- 其它典型问题
- 迭代求解方法
- 迭代算法比较

最短路问题

（多阶段决策）最短路问题



选择从 (A) 至 (F) 的最短路铺设输油管道



状态 s_k

状态集 S_k

决策 u_k

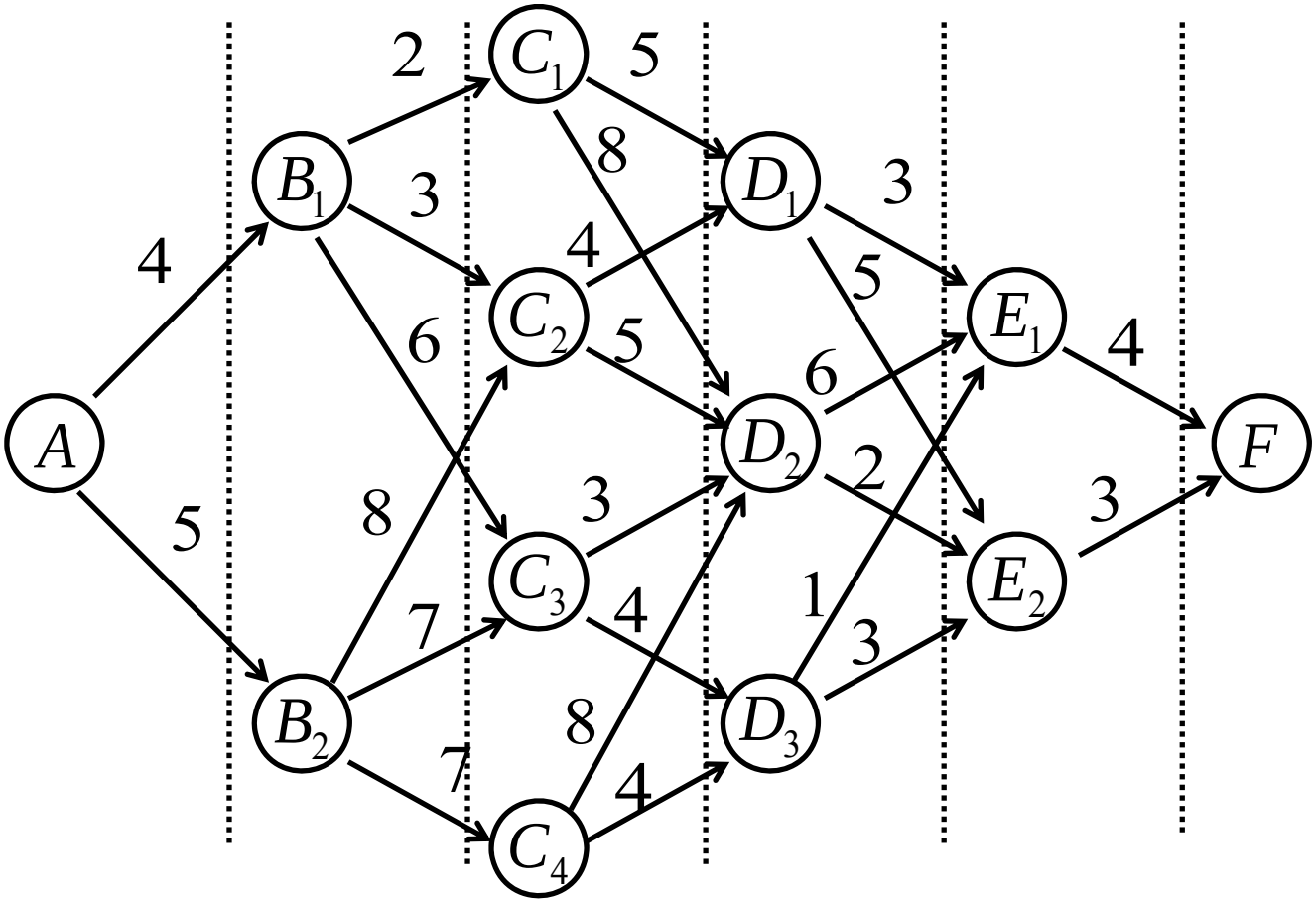
决策集 $U_k(s_k)$

策略

$p=\{u_1,\cdots,u_5\}$

策略集 P

解： $S_2=\{B_1,B_2\}$ $U_4(D_2)=\{E_1,E_2\}$ $p=\{B_1,C_3,D_2,E_2,F\}$



状态转移函数 $T_k(s_k,u_k)\in S_{k+1},\ \forall s_k\in S_k,u_k\in U_k(s_k)$

阶段指标函数 $d_k(s_k,u_k),\ \forall s_k\in S_k,u_k\in U_k(s_k)$

过程指标函数 $d_1(s_1,u_1)+\sum_{k=2}^{5}d_k\left(T_{k-1}\left(s_{k-1},u_{k-1}\right),u_k\right)$

用动态规划的术语描述最短路问题

已知　状态集　$S_k, k=1,2,\cdots,6$

决策集　$U_k(s_k), \forall s_k \in S_k, k=1,2,\cdots,5$

状态转移函数

$$T_k(s_k,u_k)=u_k, \forall s_k \in S_k, u_k \in U_k(s_k), k=1,2,\cdots,5$$

阶段指标函数

$$d_k(s_k,u_k), \forall s_k \in S_k, u_k \in U_k(s_k), k=1,2,\cdots,5$$

问题　求 $p=\{u_1,\cdots,u_5\}$ 使下述过程指标函数达到最小

$$d_1(s_1,u_1)+\sum_{i=2}^{5} d_k\left(T_{k-1}(s_{k-1},u_{k-1}),u_k\right)$$

最短路问题的动态规划模型

$$\min \quad \sum_{k=1}^{5} d_k\left(s_k, u_k\right)$$

$$\text{s.t.} \quad s_{k+1} = T_k\left(s_k, u_k\right),\ s_k \in S_k,\ u_k \in U_k(s_k),\ 1 \le k \le 5$$

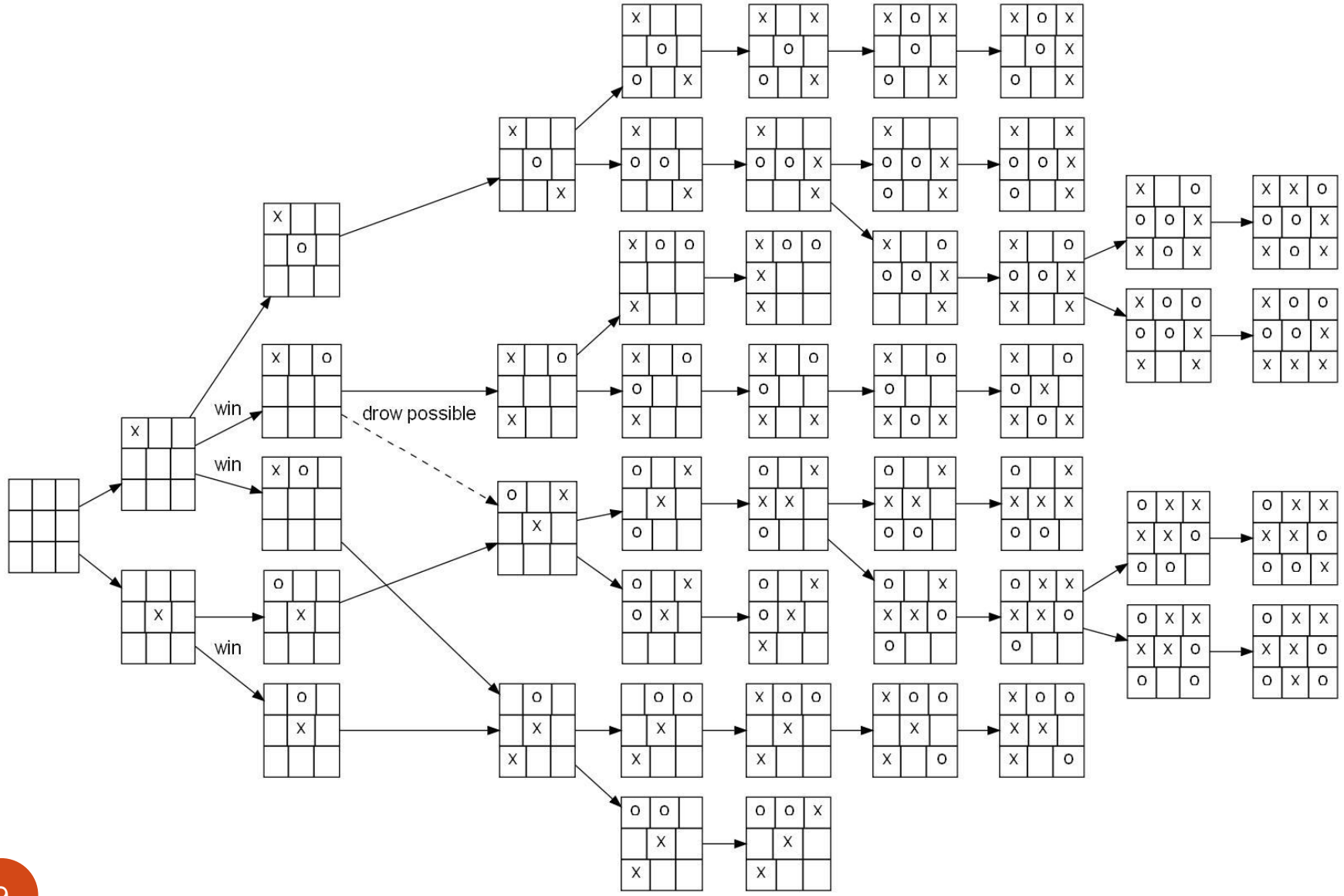
满足马尔可夫性或无后效性，用动态规划求解的多阶段模型必须具有无后效性！

给定 s_k ，系统在 k 阶段以后的状态和系统经由什么路径到达 s_k 无关，即和 $s_1, s_2, \ldots, s_{k-1}$ 的取值无关

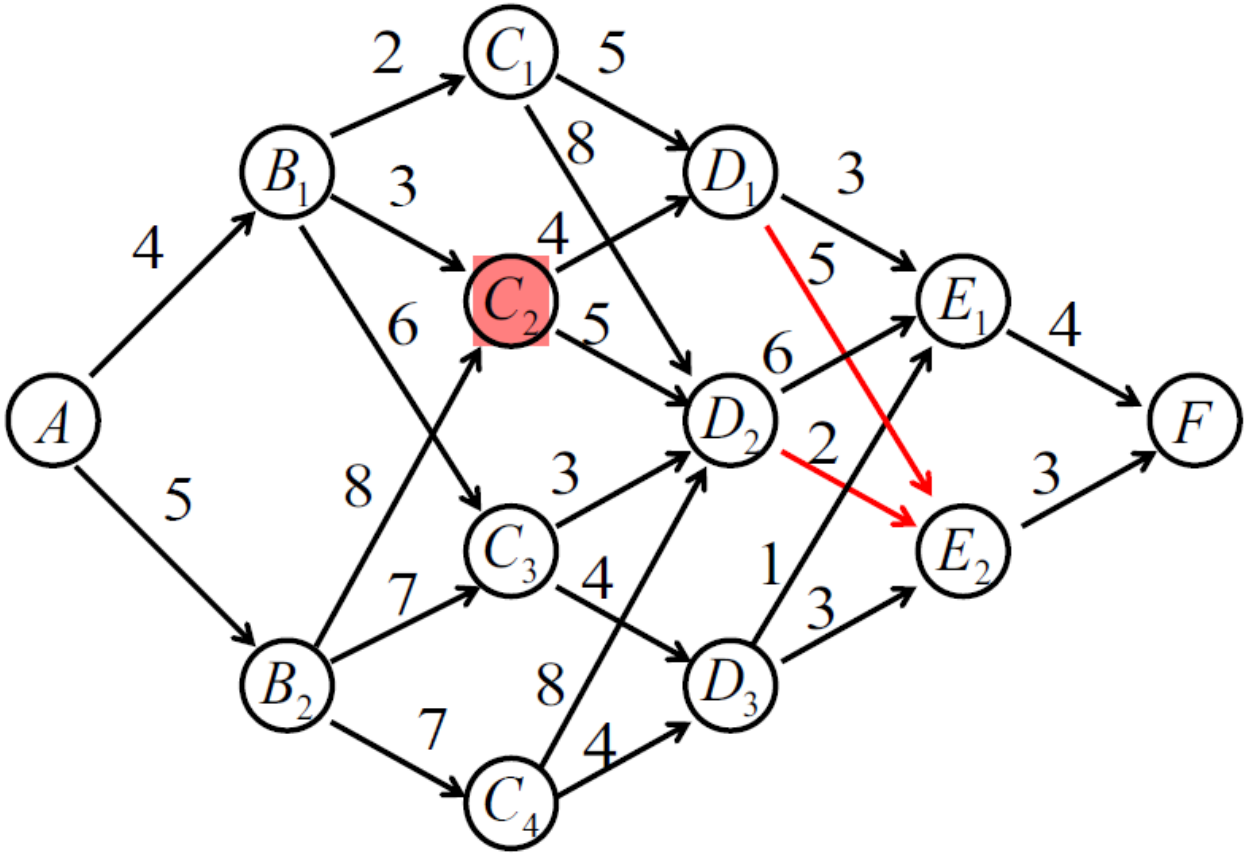
$\Rightarrow$ 最优值函数一定满足最优值方程

$$f(s) = \min_{u \in U(s)} \left\{ d(s,u) + f\left(T(s,u)\right) \right\},\ \forall s \in S$$

棋类游戏是典型满足马尔可夫性的



不满足马尔可夫性的情况



假定附加约束：如果管道经过 C_2，就必须经过 E_1

$$U_4(D_1, s_2 = C_2) = \{E_1\},\ U_4(D_1, s_2 \neq C_2) = \{E_1, E_2\},$$

$$U_4(D_2, s_2 = C_2) = \{E_1\},\ U_4(D_2, s_2 \neq C_2) = \{E_1, E_2\},$$

多阶段最短路问题的逆推求解

定义最优值函数 $f(s)$ 为从 s 到终点的最短路程，并根据多阶段结构将其表示为

$$f(s)=f_k(s),\forall s\in S_k,k=1,\cdots,6$$

终止条件： $f(s)=f_6(s)=0,\forall s\in S_6$

利用多阶段结构，可得到最优性方程的以下等价式

$$f(s)=\min_{u\in U(s)}\left\{d(s,u)+f\left(T(s,u)\right)\right\}$$

$$\Leftrightarrow\quad f_k(s)=\min_{u\in U_k(s)}\left\{d_k(s,u)+f_{k+1}\left(T_k(s,u)\right)\right\},\ \forall s\in S_k$$

结论：最优值函数 $f(s)$ 可以用以下公式逆推确定

$$f_6(s)=0,\ \forall s\in S_6$$

$$f_5(s)=\min_{u\in U_5(s)} d_5(s,u)+f_6\left(T_5(s,u)\right),\ \forall s\in S_5$$

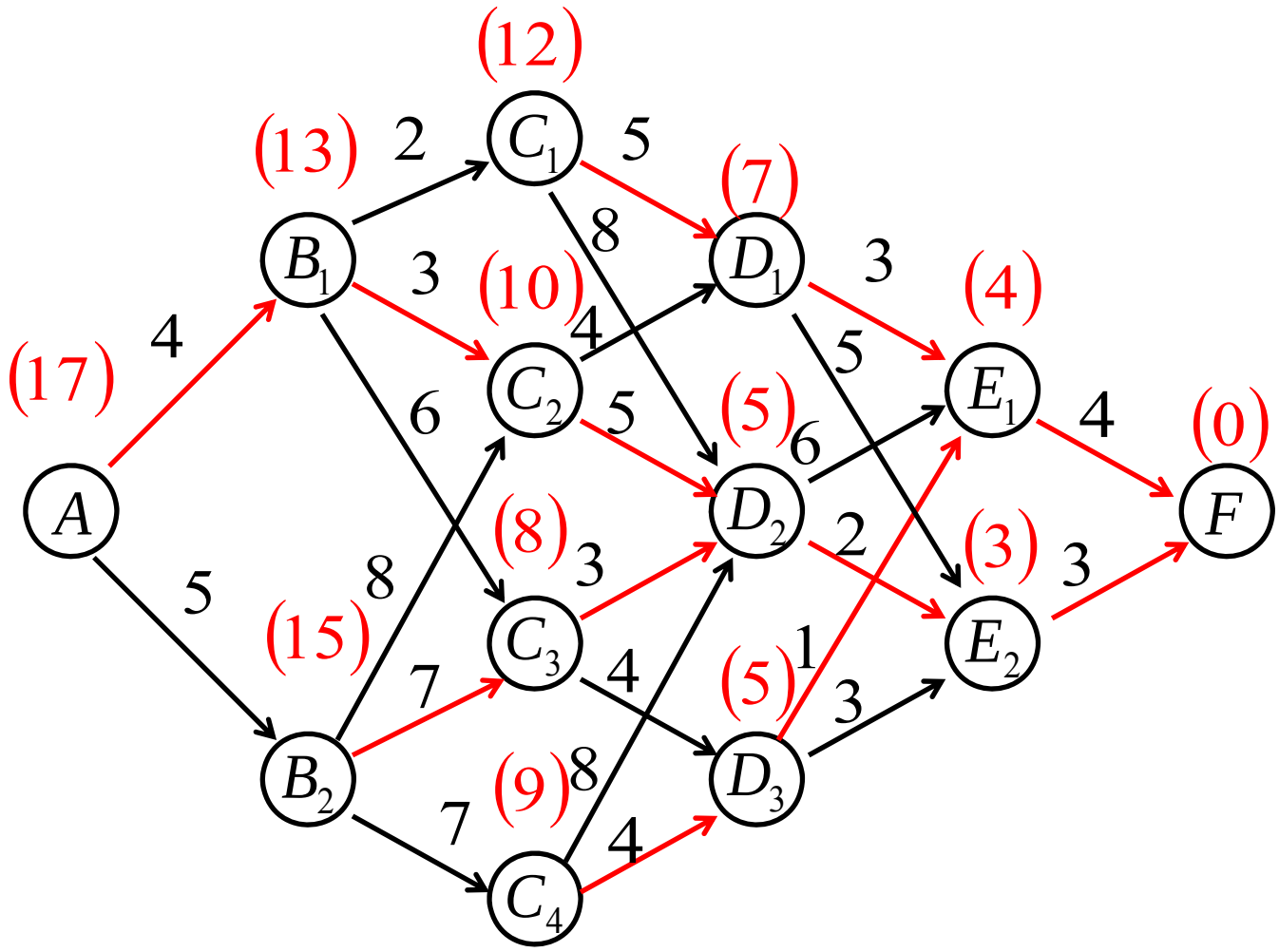
$$f_4(s)=\min_{u\in U_4(s)} d_4(s,u)+f_5\left(T_4(s,u)\right),\ \forall s\in S_4$$

$$f_3(s)=\min_{u\in U_3(s)} d_3(s,u)+f_4\left(T_3(s,u)\right),\ \forall s\in S_3$$

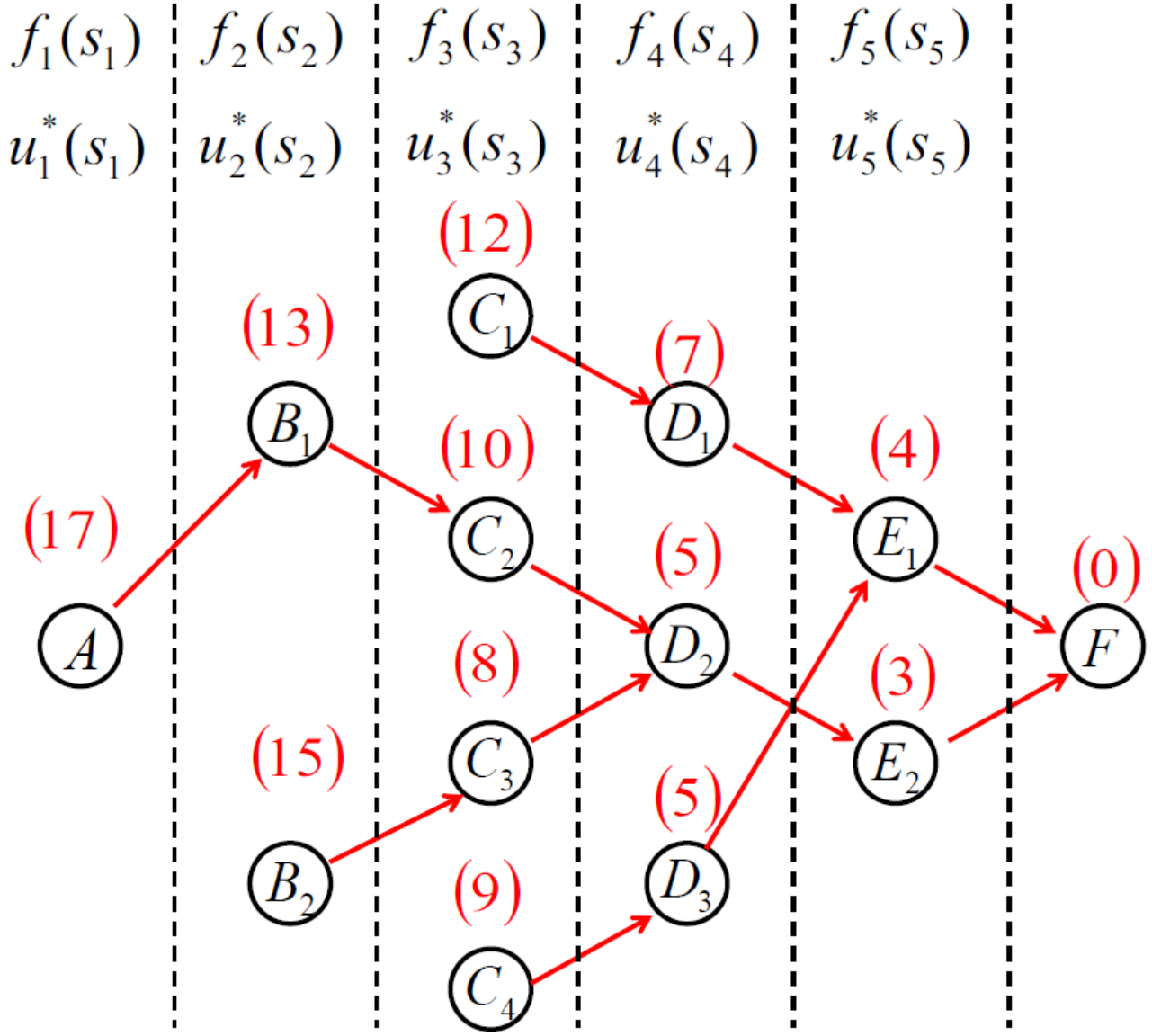
$$f_2(s)=\min_{u\in U_2(s)} d_2(s,u)+f_3\left(T_2(s,u)\right),\ \forall s\in S_2$$

$$f_1(s)=\min_{u\in U_1(s)} d_1(s,u)+f_2\left(T_1(s,u)\right),\ \forall s\in S_1$$

最短路问题的逆推结果（最优决策可由最优值得到）



最短路问题的逆推结果（最优决策可由最优值得到）



例如

$f_3(C_1)=12$

$f_3(C_2)=10$

$f_3(C_3)=8$

$f_3(C_4)=9$

$u_3^*(C_1)=D_1$

$u_3^*(C_2)=D_2$

$u_3^*(C_3)=D_2$

$u_3^*(C_4)=D_3$

多阶段最短路问题的顺推求解

定义最优值函数 $\overline{f}(s)$ 为从起点到 s 的最短路程，并根据多阶段结构将其表示为

$$\overline{f}(s)=\overline{f}_k(s),\ \forall s\in S_k,\ k=1,\cdots,6$$

初始条件：$\overline{f}(s)=\overline{f}_1(s)=0,\ \forall s\in S_1$

由于对任意 k 成立 $S_{k+1}=T_k\left(S_k,U_k(S_k)\right)$，所以

$$\overline{f}(s)=\min_{\substack{T(\overline{s},u)=s\\ \overline{s}\in S,u\in U(\overline{s})}}\left\{d(\overline{s},u)+\overline{f}(\overline{s})\right\}$$

$$\Leftrightarrow\quad \overline{f}_{k+1}(s)=\min_{\substack{T_k(\overline{s},u)=s\\ \overline{s}\in S_k,u\in U_k(\overline{s})}}\left\{d_k(\overline{s},u)+f_k(\overline{s})\right\},\ \forall s\in S_{k+1}$$

结论：最优值函数 $\overline{f}(s)$ 可以用以下公式顺推确定

$$\overline{f}_1(s)=0,\ \forall s\in S_1$$

$$\overline{f}_2(s)=\min_{\substack{T_1(\overline{s},u)=s\\ \overline{s}\in S_1,u\in U_1(\overline{s})}} d_1(\overline{s},u)+f_1(\overline{s}),\ \forall s\in S_2$$

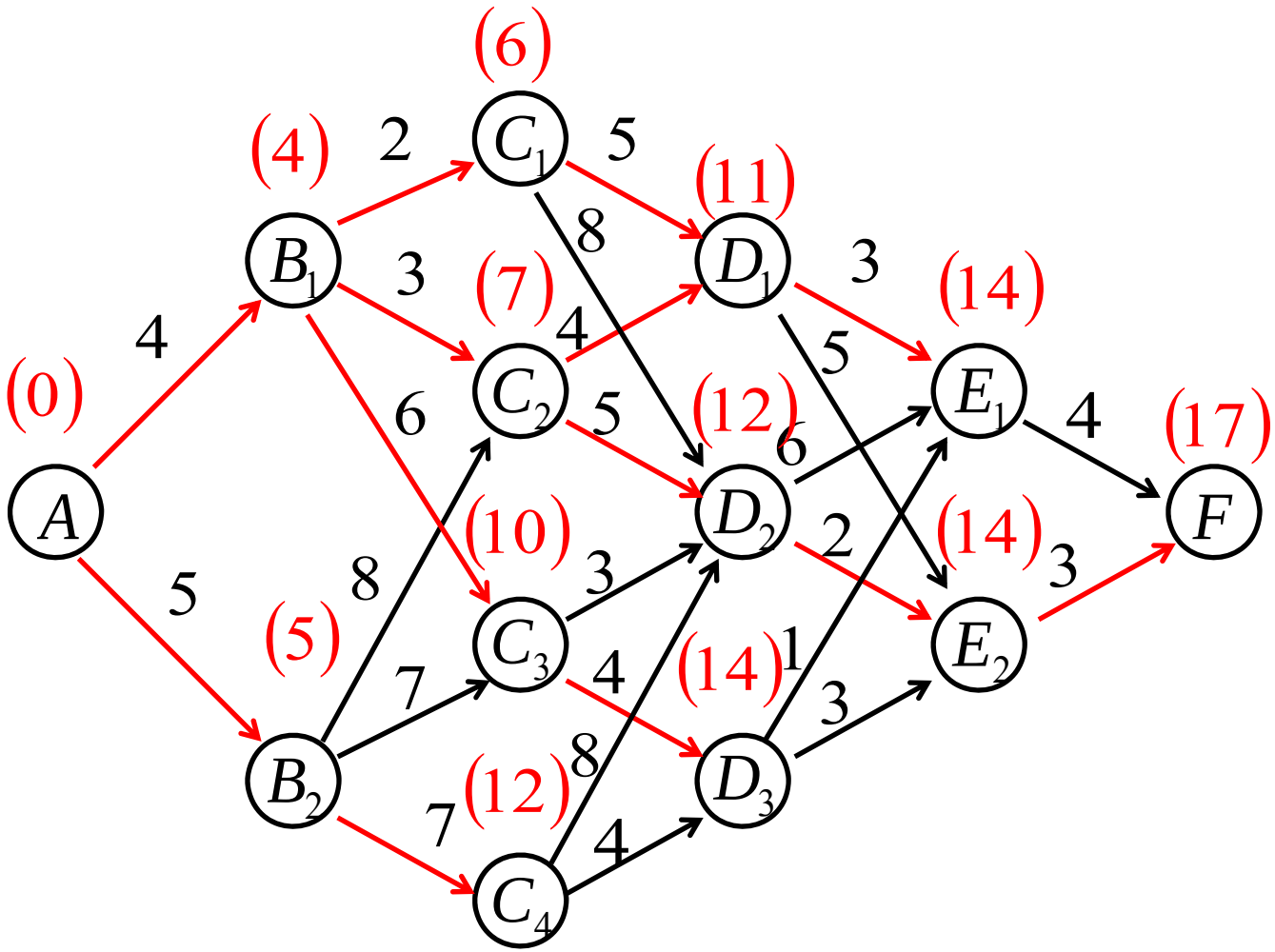
$$\overline{f}_3(s)=\min_{\substack{T_2(\overline{s},u)=s\\ \overline{s}\in S_2,u\in U_2(\overline{s})}} d_2(\overline{s},u)+f_2(\overline{s}),\ \forall s\in S_3$$

$$\overline{f}_4(s)=\min_{\substack{T_3(\overline{s},u)=s\\ \overline{s}\in S_3,u\in U_3(\overline{s})}} d_3(\overline{s},u)+f_3(\overline{s}),\ \forall s\in S_4$$

$$\overline{f}_5(s)=\min_{\substack{T_4(\overline{s},u)=s\\ \overline{s}\in S_4,u\in U_4(\overline{s})}} d_4(\overline{s},u)+f_4(\overline{s}),\ \forall s\in S_5$$

$$\overline{f}_6(s)=\min_{\substack{T_5(\overline{s},u)=s\\ \overline{s}\in S_5,u\in U_5(\overline{s})}} d_5(\overline{s},u)+f_5(\overline{s}),\ \forall s\in S_6$$

最短路问题的顺推结果（最优决策可由最优值得到）



一般性动态规划模型（其中 $\odot$ 表示某种运算，如加法）

$$\min\ (\text{or}\ \ \max)\ \ d_1(s_1,u_1)\odot d_2(s_2,u_2)\odot\cdots\odot d_n(s_n,u_n)$$

$$\text{s.t.}\quad s_{k+1}=T_k(s_k,u_k),\ \ s_k\in S_k,\ \ u_k\in U_k(s_k),\ \ 1\le k\le n$$

逆推求解

$$f_{n+1}(s)=0,\ \ \forall s\in S_{n+1}$$

（0 的含义根据问题定）

$$f_k(s)=\min_{u\in U_k(s)} d_k(s,u)\odot f_{k+1}\left(T_k(s,u)\right),\ \ \forall s\in S_k,k=n,\cdots,1$$

顺推求解

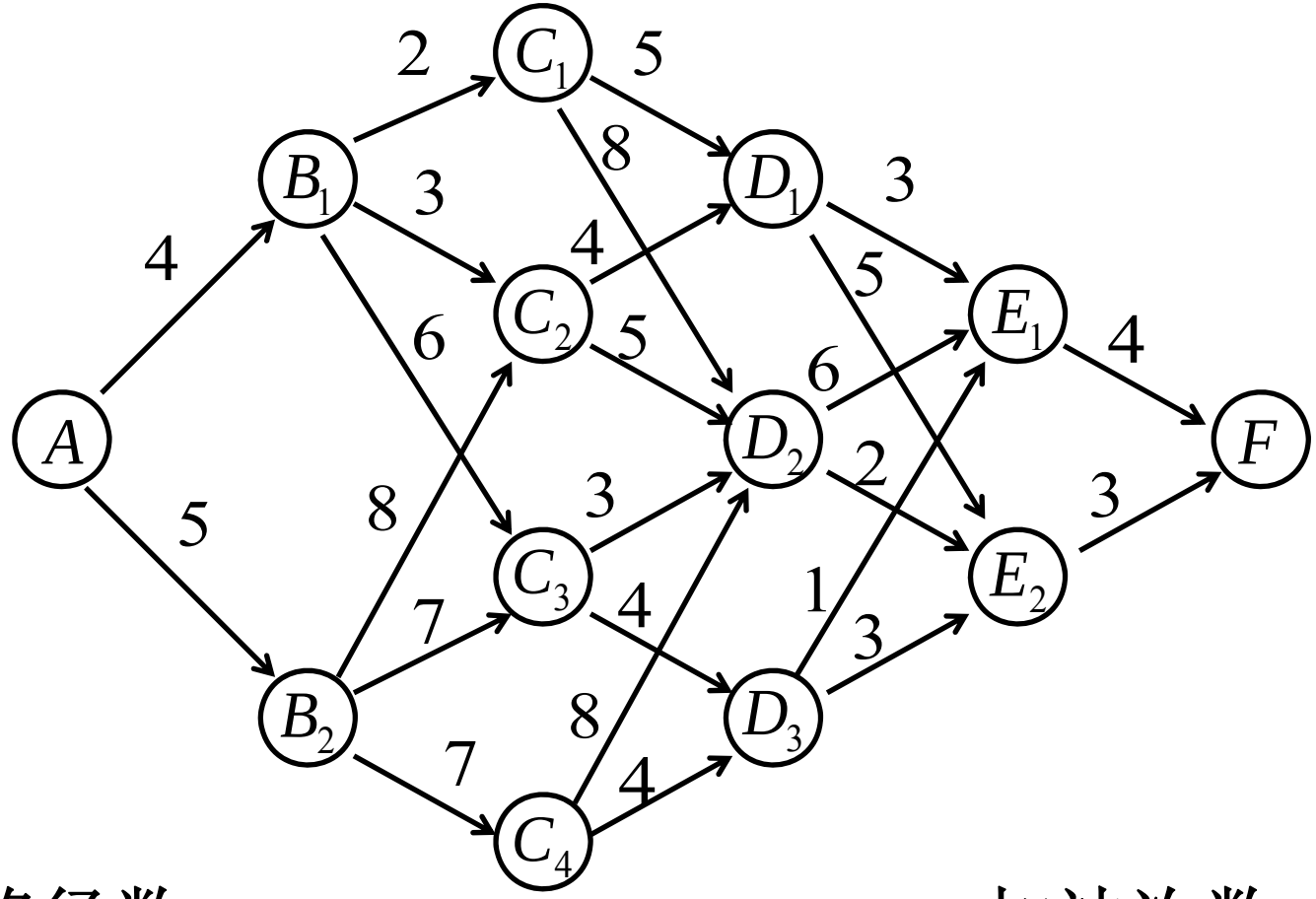
$$\overline{f}_1(s)=0,\ \ \forall s\in S_1$$

$$\overline{f}_{k+1}(s)=\min_{\substack{T_k(\overline{s},u)=s\\ \overline{s}\in S_k,u\in U_k(\overline{s})}} d_k(\overline{s},u)\odot f_k(\overline{s}),\ \ \forall s\in S_{k+1},k=1,\cdots,n$$

问题：动态规划方法能否保证最优解？

动态规划方法如何节省计算量？

穷举（可保证最优解）



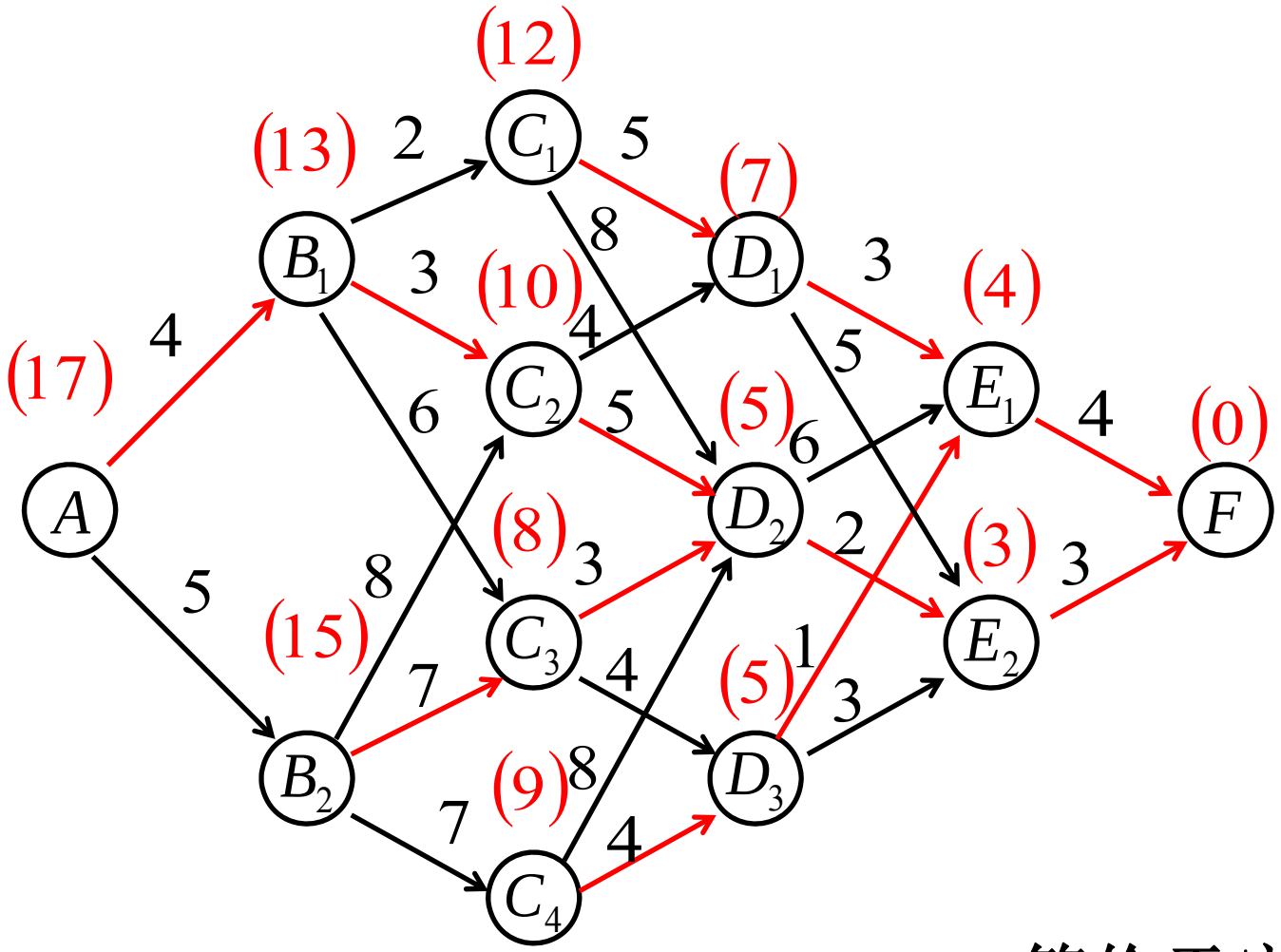
总路径数

$2\times3\times2\times2=24$

加法次数

$5\times24=120$

逆推求解



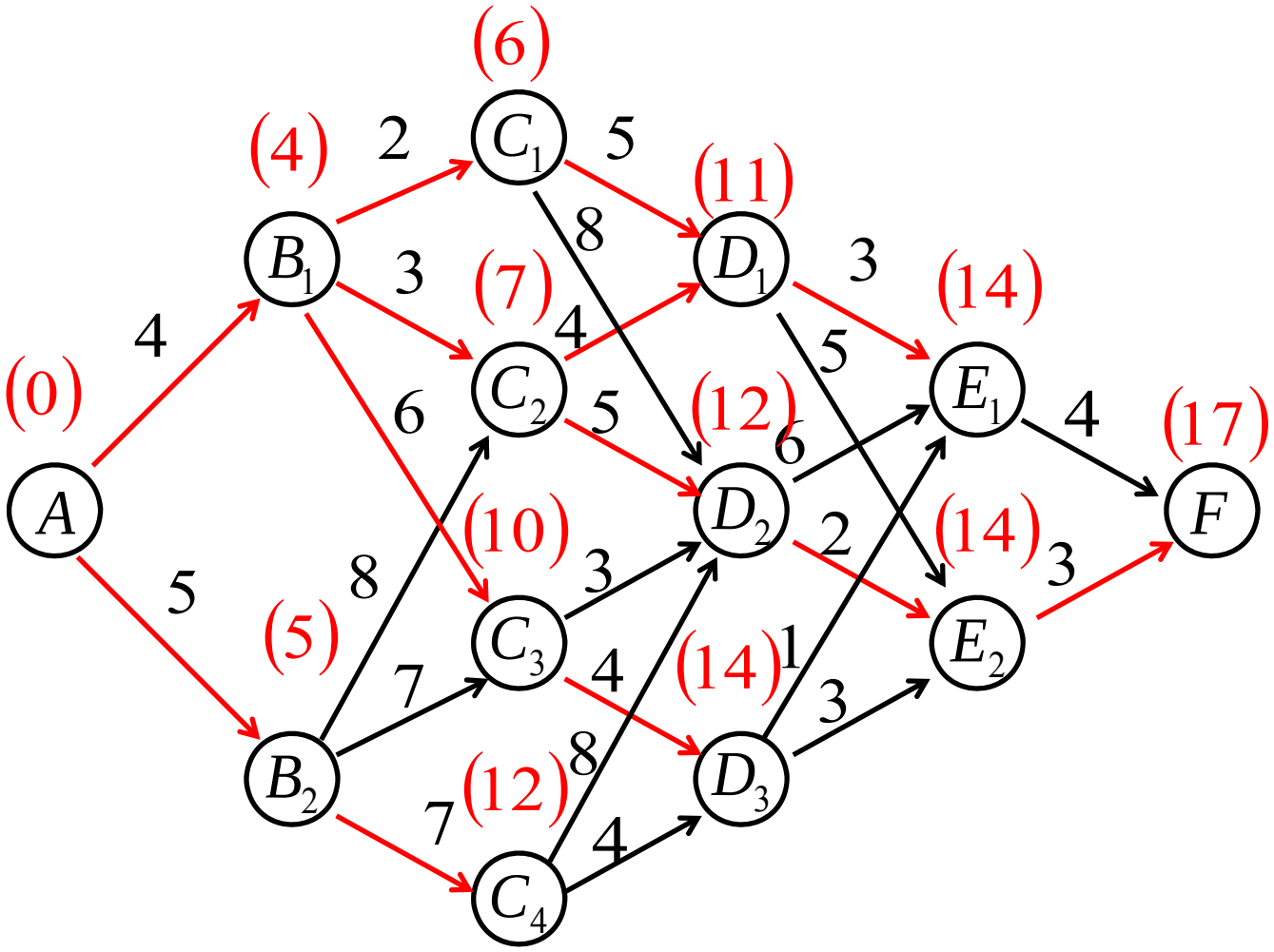
加法次数

$3\times3+2\times4+3\times2+2=22$

等价于穷举

无重复计算

顺推求解



加法次数

$$(2\times2+2\times2)+(2\times2+4)+3\times2+2=22$$

等价于穷举

无重复计算

一般性问题

"The term dynamic programming was originally used in the 1940s by Richard Bellman to describe the process of solving problems where one needs to find the best decisions one after another. By 1953, he refined this to the modern meaning, referring specifically to nesting smaller decision problems inside larger decisions, and the field was thereafter recognized by the IEEE as a systems analysis and engineering topic. Bellman's contribution is remembered in the name of the Bellman equation, a central result of dynamic programming which restates an optimization problem in recursive form."

https://en.wikipedia.org/wiki/Dynamic_programming#History

阶段（stage） 把所研究的决策问题，按先后顺序划分为若干相互联系的决策步骤，以便按一定的次序进行求解。描述阶段的变量称阶段变量，常用k表示。

状态（state） 表示每个阶段开始时所处的自然状况或客观条件，它描述了影响决策的因素随决策进程的变化情况，它既是前面阶段所作决策的结果又是本阶段作出决策的出发点和依据。描述状态的变量称为状态变量，第k阶段的状态变量常用s_k表示。通常，在第一阶段状态变量s_1是确定的，称初始状态。

决策（decision） 表示在某一阶段处于某种状态时，决策者在若干种方案中作出的选择决定。描述决策的变量称决策变量，第k阶段的决策变量常用u_k表示。决策变量的取值会受到状态变量的制约，被限制在某范围之内。

策略（policy） 把从第一阶段开始到最后阶段终止的整个决策过程，称为问题的全过程；而把从第k阶段开始到最后阶段终止的决策过程，或称为k子过程。在全过程上，各阶段的决策按顺序排列组成的决策序列

$$p_{1,n} = \{ u_1, u_2, \cdots\cdots, u_n \}$$

称为全过程策略，简称策略；而在k子过程上的决策序列

$$p_{k,n} = \{ u_k, u_{k+1}, \cdots\cdots, u_n \}$$

称为k子过程策略，也简称子策略。

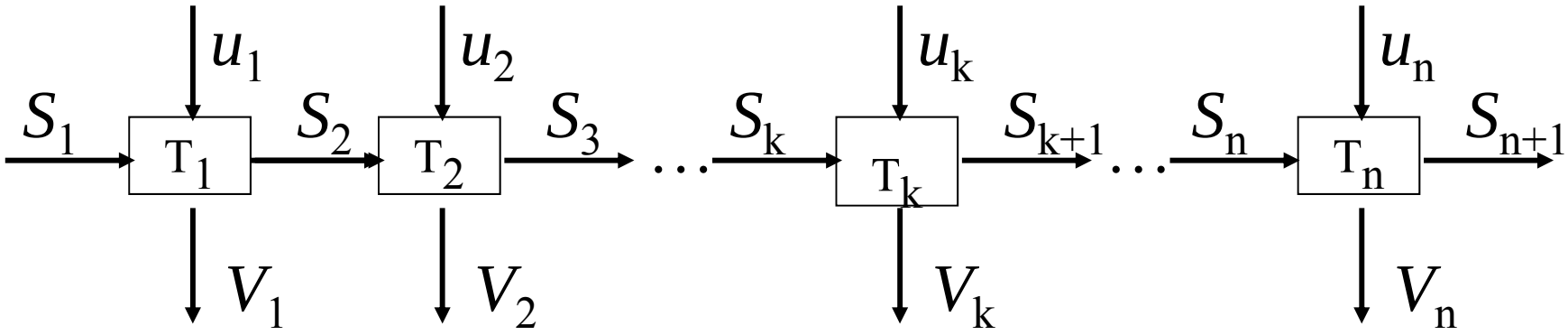
状态转移方程

若第k阶段的状态变量值为s_k，当决策变量u_k的取值决定后，下一阶段状态变量s_{k+1}的值也就完全确定。即s_{k+1}的值对应于s_k和u_k的值。这种对应关系记为：

$$s_{k+1}=T_k(s_k, u_k),$$

称为状态转移方程。状态转移方程描述了由一个阶段的状态到下一阶段的状态的演变规律。

序贯多阶段决策过程如下：



n个决策子问题

k称为阶段变量

S_k描述k阶段初的状态，一般把输入状态称为该阶段的阶段状态。

u_k的取值代表k阶段对第k子问题所进行的决策，称为k阶段的决策变量

V_k为k阶段从状况S_k出发，做出决策u_k之后的后果，

指标函数（准则函数），目标函数和最优值函数

指标函数分为阶段指标函数和过程指标函数。

阶段指标函数是对某一阶段的状态和决策产生的效益值的度量，用$V_k(s_k, u_k)$表示。

过程指标函数是指从第k阶段至第n阶段所包含的各阶段的状态和决策所产生的总的效益值，记为：

$$V_{k,n} = V_{k,n}(S_k, u_k, S_{k+1}, u_{k+1}, \cdots\cdots, S_n, u_n)$$

定义在全过程上的准则函数相当于目标函数，一般记为：

$V_{1,n} = V_{1,n}(S_1, u_1, \cdots S_k, u_k, \cdots, S_n, u_n)$，或简记为$V_{1,n}$

动态规划所要求的**过程指标函数应具有可分离性**，即可表达为它所包含的各阶段指标函数的函数形式。常见的两种过程指标函数形式是：

①各阶段指标函数的和：

②各阶段指标函数的积：

$$V_{k,n}=\sum_{j=k}^{n}V_j(S_j,u_j)$$

$$V_{k,n}=\prod_{j=k}^{n}V_j(S_j,u_j)$$

把过程指标函数$V_{k,n}$对k子过程策略$p_{k,n}$求最优，得到一个关于状态S_k的函数，称为**最优值函数**或**贝尔曼函数**，记为：

$$f_k(S_k)=\underset{u_k,\cdots,u_n}{opt}\ V_{k,n}(S_k,u_k,S_{k+1},u_{k+1}\cdots,S_n,u_n)$$

也就是说在阶段k从初始状态S_k出发，执行最优决策序列或策略，到达过程终点时，整个k-子过程中的最优目标函数取值。式中的“opt”可根据具体问题取min或max。

不同问题的要素不尽相同，根据要素的差异，多阶段决策问题可以分成不同类型：

1. 根据阶段数可分为：有限阶段决策问题，其阶段数为有限值；无限阶段决策问题，其阶段数为无穷大，决策过程可无限持续下去

2. 根据变量取值情况可分为：连续多阶段决策问题，决策变量和状态变量取连续变化的实数；离散多阶段决策问题，决策变量和状态变量取有限的数值

3. 根据阶段个数是否明确可分为：定期多阶段决策问题，其阶段数是明确的，不受决策的影响；不定期多阶段决策问题，其阶段数是不确定的，不同的决策下阶段数不同

4. 根据参数取值情况可分为：确定多阶段决策问题，其参数是给定的常数；不确定多阶段决策问题，其参数中包含不确定因素，如随机参数，区间取值参数等

最优性原理

从多阶段决策问题的数学模型可以看到，一个多阶段决策过程的极值函数可以看做是过程的初始状态与阶段数目的函数.任意给定一个决策序列(或称为策略)，如果是最优的，那么从任何最后 k 阶段开始，对由这个策略形成的后面 k 阶段的初始状态组成的 k 阶段问题而言，这个策略的后面 k 个决策一定是这个 k 阶段问题的最优策略，与这 k 阶段以前的决策无关.这个必要条件是很显然的.这样，动态规划的最优化原理可叙述如下：

动态规划最优化原理

一个过程的最优策略具有这样的性质，即无论其初始状态及其初始决策如何，其以后诸决策对以第一个决策所形成的状态作为初始状态而言，必须构成最优策略.

教材原话

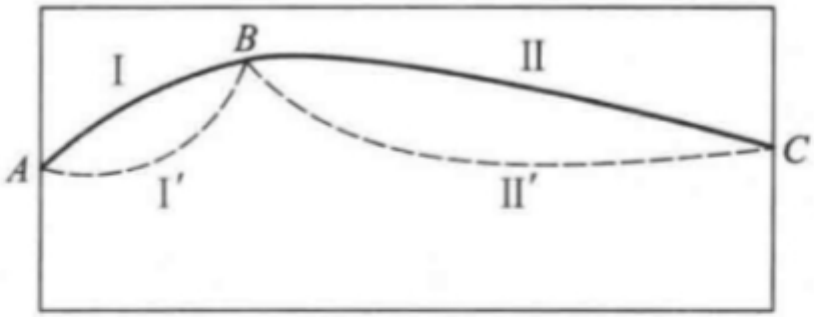


图 4.2.2

很容易想到，如果Ⅰ－Ⅱ是 A 到 C 的最优路线，那么Ⅰ就是 A 到 B 的最优路线.如果存在 A 到 B 的更好的路线Ⅰ′，显然Ⅰ′－Ⅱ将是比Ⅰ－Ⅱ更好的 A 到 C 的路线，这与假设矛盾.

所以最优化原理也应该包括下述性质：

对于多阶段决策问题的最优策略，如果用它的前 i 步策略产生的情况（加上原有的约束条件）来形成一个前 i 步问题，那么所给最优策略的前 i 阶段的策略构成这前 i 步问题的一个最优策略.

有的书上称上述性质为**前向最优化原理**，而称前面所述的性质为**后向最优化原理**，我们都统称为**最优化原理**.

建模与求解

动态规划建模有以下过程：

①确定阶段与阶段变量

②明确状态变量和状态可能集合。

③确定决策变量和决策允许集合。

④确定状态转移方程。

⑤明确阶段目标，写出基本方程和边界条件

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**）。

动态规划的三个特点：

1. 把原来的问题分解成了几个相似的子问题。（强调“相似子问题”）

2. 所有的子问题都只需要解决一次。（强调“只解决一次”）

3. 储存子问题的解。（强调“储存”）储存重叠子问题的解，并不只是一个“花费空间来节省时间的技巧”。如果不考虑这一点，则只能算是分治法，而不是动态规划。

动态规划是通过拆分问题，定义问题状态和状态之间的关系，使得问题能够以递推（或者说分治）的方式去解决。Dynamic programming is a method for solving a complex problem by breaking it down into simpler subproblems, solving each of those subproblems just once, and storing their solutions.

https://www.sanfoundry.com/dynamic-programming-problems-solutions/

并非所有的递推式求解都是动态规划。

动态规划是一种巧妙Smart的递推式求解

如何拆分问题是动态规划研究的核心之一。

斐波那契数列

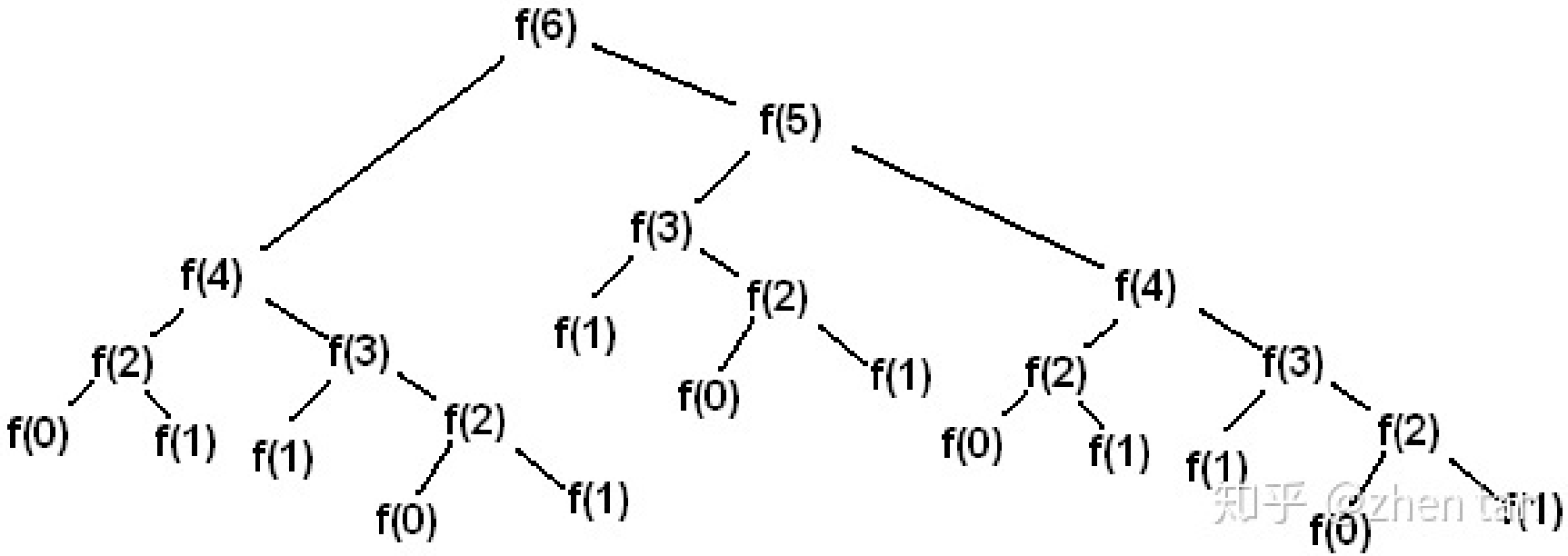
1. 简单递归(反例)

```
def fib(n):
    if n < 2:
        return n
    else:
        return fib(n-1) + fib(n-2)

if __name__ == '__main__':
    result = fib(100)
```

https://www.zhihu.com/question/39948290

斐波那契数列


f(6)
f(5)
f(4)
f(3)
f(4)
f(2)
f(3)
f(1)
f(2)
f(2)
f(3)
f(0)
f(1)
f(1)
f(2)
f(0)
f(1)
f(0)
f(1)
f(1)
f(2)
f(0)
f(1)
f(0)
f(1)

斐波那契数列

2. 动态规划

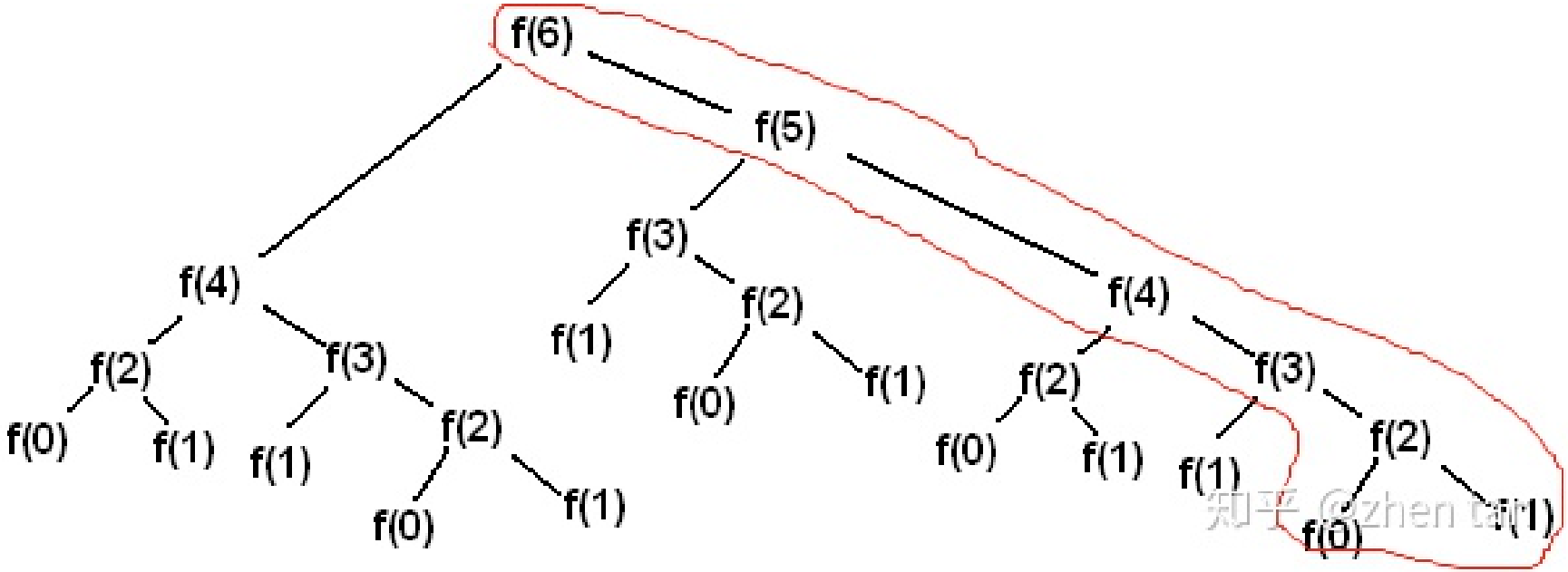
```
def fib(n):

    results = list(range(n+1)) # 用于缓存以往结果，以便复用（目标2）

    for i in range(n+1):  # 按顺序从小往大算（目标3）
        if i < 2:
            results[i] = i
        else:
            # 使用状态转移方程（目标1），同时复用以往结果（目标2）
            results[i] = results[i-1] + results[i-2]

    return results[-1]
```

斐波那契数列


f(6)
f(5)
f(4)
f(3)
f(4)
f(2)
f(3)
f(1)
f(2)
f(2)
f(3)
f(0)
f(1)
f(1)
f(2)
f(0)
f(1)
f(0)
f(1)
f(2)
f(0)
f(1)
f(1)
f(2)
f(0)
f(1)

枚举不同路径数目

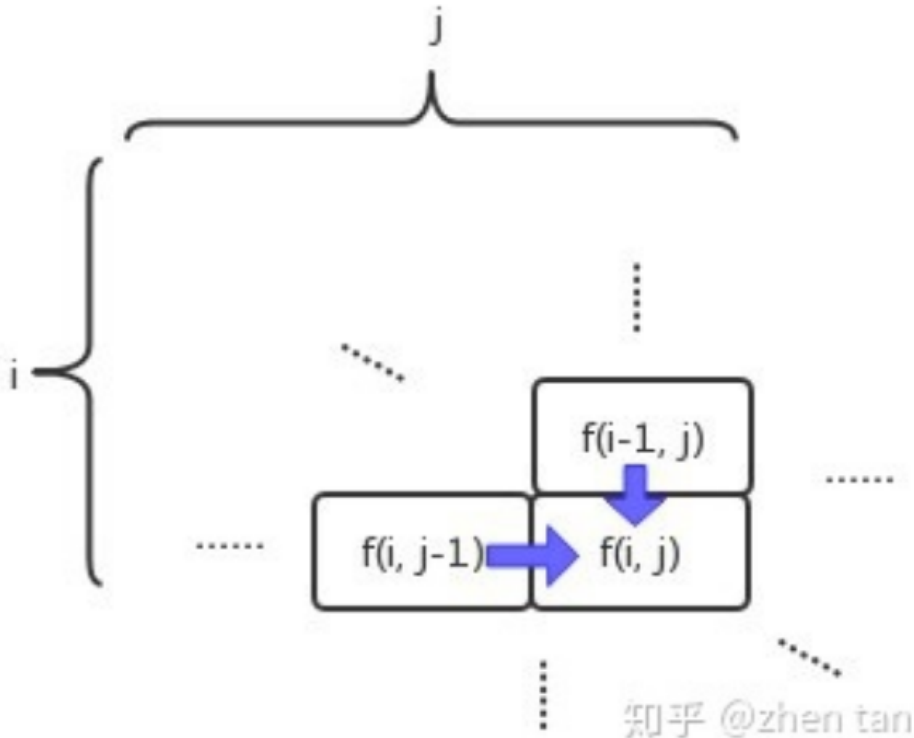
一个机器人位于一个 m x n 网格的左上角（起始点在下图中标记为“Start”）。机器人每次只能**向下**或者**向右**移动一步。机器人试图达到网格的右下角（在下图中标记为“Finish”）。问总共有多少条不同的路径？

https://www.zhihu.com/question/39948290

枚举不同路径数目

1. 建立状态转移方程。该题就难在这里，这一步搞不定基本上GG了。实际上，如图3所示，第 i 行第 j 列的格子的路径数，是等于它左边格子和上面格子的路径数之和：

$$f(i,j) = f(i-1,j) + f(i,j-1)$$ 。



枚举不同路径数目

2. 缓存并复用以往结果。与之前说的一维数列不同，这里的中间结果可以构成一个二维数列（如图3），所以需要用二维的数组或者列表来存储。

3. 按顺序从小往大算。这次有两个维度，所以需两个循环，分别逐行和逐列让问题从小规模到大规模计算。

```python
# m是行数，n是列数
def count_paths(m, n):

    results = [[1] * n] * m  # 将二维列表初始化为1，以便之后用于缓存（目标2）
    # 题外话：results的空间复杂度不是O(nm)，是O(n)

    # 第0行和第0列的格子路径数显然均取值为1，所以跳过
    for i in range(1, m):        # 外循环逐行计算（目标3）
        for j in range(1, n):    # 内循环逐列计算（目标3）
            # 状态方程（目标1），以及中间结果复用（目标2）
            results[i][j] = results[i-1][j] + results[i][j-1]

    return results[-1][-1]
```

宝石挑选问题

小 Q 是一个宝石爱好者。

这一天，小 Q 来到了宝石古董店，店家觉得小 Q 是个宝石行家，于是决定和小 Q 玩一个游戏。

游戏是这样的，一共有 n 块宝石，每块宝石在小 Q 心中都有其对应的价值。注意，由于某些宝石质量过于差劲，因此存在只有店家倒贴钱，小 Q 才愿意带走的宝石，即价值可以为负数。

小 Q 可以免费带走一个连续区间中的宝石，比如区间 $[1,3]$ 或区间 $[2,4]$ 中的宝石。

请问小 Q 能带走的最大价值是多少？

https://www.zhihu.com/question/39948290

宝石挑选问题

问题分析

首先思考最暴力的解法。

枚举所有区间，暴力累加区间中宝石的价值，最后选一个价值最大的区间。时间复杂度 $O(n^3)$。

$O(n^3)$ 显然有些无法接受，因此想想有没有办法优化，比如优化掉暴力累加的部分。

优化 1.0

仔细思考不难发现，我们可以枚举区间右端点，然后固定右端点，左端点不断向左移动，边移动边累加，就可以将时间复杂度优化到 $O(n^2)$ 。

例如我们固定右端点是 3，那么左端点就从 3 移动到 1，边移动边累加答案，就可以在移动过程中计算出区间 $[3,3]$ 、 $[2,3]$ 、 $[1,3]$ 的答案了。因此枚举所有区间右端点，即可在 $O(n^2)$ 时间复杂度内找到答案。

宝石挑选问题

优化 2.0

观察 $O(n^2)$ 的解法，不难发现我们用了 $O(n)$ 的时间复杂度才求出了固定某个点为区间右端点时，区间最大价值和。

例如固定了 n 为区间右端点后，我们通过从 n 到 1 枚举左端点，才求出了以 n 为区间右端点时的区间最大价值和，即 $O(n)$ 的时间复杂度。

那么继续思考，「以 n 为区间右端点的区间最大和」，与「以 $n-1$ 为区间右端点的区间最大和」，这两者是否有关联呢？

为了描述方便，接下来我们用 $f[i]$ 来代替「以 i 为区间右端点的区间最大和」，用 $a[i]$ 来代替第 i 块宝石的价值。

宝石挑选问题

不难发现，如果 $f[n-1]$ 为正数，则 $f[n]$ 一定等于 $f[n-1]+a[n]$ ；如果 $f[n-1]$ 为负数，则 $f[n]$ 一定等于 $a[n]$ 。因此我们可以推导出如下转移方程：

$$f[i]=max(f[i-1]+a[i],a[i])$$

根据上述转移方程，我们可以在 $O(n)$ 时间复杂度内求出最大的 $f[i]$ ，即将此题时间复杂度优化到 $O(n)$ ，而这个优化的过程就是「动态规划」的过程。

宝石挑选问题

最优子结构

什么是「最优子结构」？将原有问题化分为一个个子问题，即为子结构。而对于每一个子问题，其最优值均由「更小规模的子问题的最优值」推导而来，即为最优子结构。

因此「DP 状态」设置之前，需要将原有问题划分为一个个子问题，且需要确保子问题的最优值由「更小规模子问题的最优值」推出，此时子问题的最优值即为「DP 状态」的定义。

例如在「宝石挑选」例题中，原有问题是「最大连续区间和」，子问题是「以 i 为右端点的连续区间和」。并且「以 i 为右端点的最大连续区间和」由「以 $i-1$ 为右端点的最大连续区间和」推出，此时后者即为更小规模的子问题，因此满足「最优子结构」原则。

由此我们才定义 DP 状态 $f[i]$ 表示子问题的最优值，即「以 i 为右端点的最大连续区间和」。

其它典型问题

投资分配问题

10万元资金，投资三个项目，收益分别为

$$g_1(x_1)=4x_1,\ g_2(x_2)=9x_2,\ g_3(x_3)=2x_3^2$$

如何分配投资额？

上述问题可以表示成非线性规划问题

$$\begin{aligned}\max\ & 4x_1+9x_2+2x_3^2\\ \text{s.t.}\ & x_1+x_2+x_3=10\\ & x_i\geq 0,\ i=1,2,3\end{aligned}$$

不是线性规划，无法用单纯型算法

建立序贯决策模型

阶段　三个决策阶段，顺序决策三个项目投资额

状态　k 阶段开始可用投资额 s_k

决策　k 阶段实际投资额 x_k

状态集　$S_1=\{10\}, S_k=[0,10], k=2,3,4$

允许决策集　$U_k(s)=[0,s], k=1,2,3$

阶段指标　$d_1(s,x)=4x, d_2(s,x)=9x, d_3(s,x)=2x^2$

状态转移函数　$T_k(s,x)=s-x, k=1,2,3$

逆推求解（ $f_k(s)$ 表示 k 及以后阶段总收益）

逆推公式　$f_4(s)=0,\ \forall s\in S_4$

$$f_k(s)=\max_{x\in U_k(s)} d_k(s,x)+f_{k+1}\left(T_k(s,x)\right),\ \ \forall s\in S_k$$

$$\forall s\in S_3:\quad f_3(s)=\max_{0\le x\le s} 2x^2=2s^2$$

$$\forall s\in S_2:\quad f_2(s)=\max_{0\le x\le s} 9x+2(s-x)^2=\max\left\{2s^2,9s\right\}$$

一元二次函数，抛物线开口向上，顶点在定义域内，最大值必在定义域某一边界取得

$$\forall s\in S_1\Rightarrow s=10:\quad f_1(10)=\max_{0\le x\le 10} 4x+\max\left\{2(10-x)^2,9(10-x)\right\}$$

$$=\max\left\{2\times10^2,9\times10,4\times10\right\}=200$$

由最优值定最优决策

$$x_1^*=0,\ s_2^*=10-x_1^*=10,\ x_2^*=0,\ s_3^*=s_2^*-x_2^*=10,\ x_3^*=10$$

顺推求解（$\overline{f}_k(s)$ 表示 k 阶段以前总收益）

顺推公式 $\overline{f}_1(s)=0,\ \forall s\in S_1$

$$\overline{f}_{k+1}(s)=\max_{\substack{\overline{s}-x=s\\ \overline{s}\in S_k,x\in U_k(\overline{s})}} d_k(\overline{s},x)+\overline{f}_k(\overline{s}),\ \forall s\in S_{k+1}$$

$$\overline{f}_2(s)=\max_{\substack{\overline{s}-x=s\\ 0\le\overline{s}\in 10,0\le x\le\overline{s}}} 4x=\max_{0\le\overline{s}\le 10} 4(\overline{s}-s)=4(10-s)$$

$$\overline{f}_3(s)=\max_{\substack{\overline{s}-x=s\\ 0\le\overline{s}\le 10\\ 0\le x\le\overline{s}}} 9x+4(10-\overline{s})=\max_{0\le\overline{s}\le 10} 9(\overline{s}-s)+4(10-\overline{s})=9(10-s)$$

$$\overline{f}_4(s)=\max_{\substack{\overline{s}-x=s\\ 0\le\overline{s}\le 10\\ 0\le x\le\overline{s}}} 2x^2+9(10-\overline{s})=\max_{0\le\overline{s}\le 10} 2(\overline{s}-s)^2+9(10-\overline{s})$$

$$=\max\left\{2s^2+90,2(10-s)^2\right\}$$

$$\Rightarrow\quad s_4^*=0,\ x_3^*=10,\ s_3^*=10+0=10,\ x_2^*=0,\ s_2^*=0+10=10,\ x_1^*=0$$

连续状态变量的离散化求解方法

连续状态变量动态规划问题关键是求解下述递推方程

$$f_k(s)=\min_{u\in U_k(s)} d_k\left(s,u\right)+f_{k+1}\left(T_k\left(s,u\right)\right),\ \forall s\in S_k$$

只有在特殊情况下才能得到 $f_k(s)$ 的解析表达式

一般情况下，可以用离散点集近似连续状态变量集 S_k

将原问题近似转换成和最短路问题类似的多阶段问题

投资分配问题的离散化求解方法

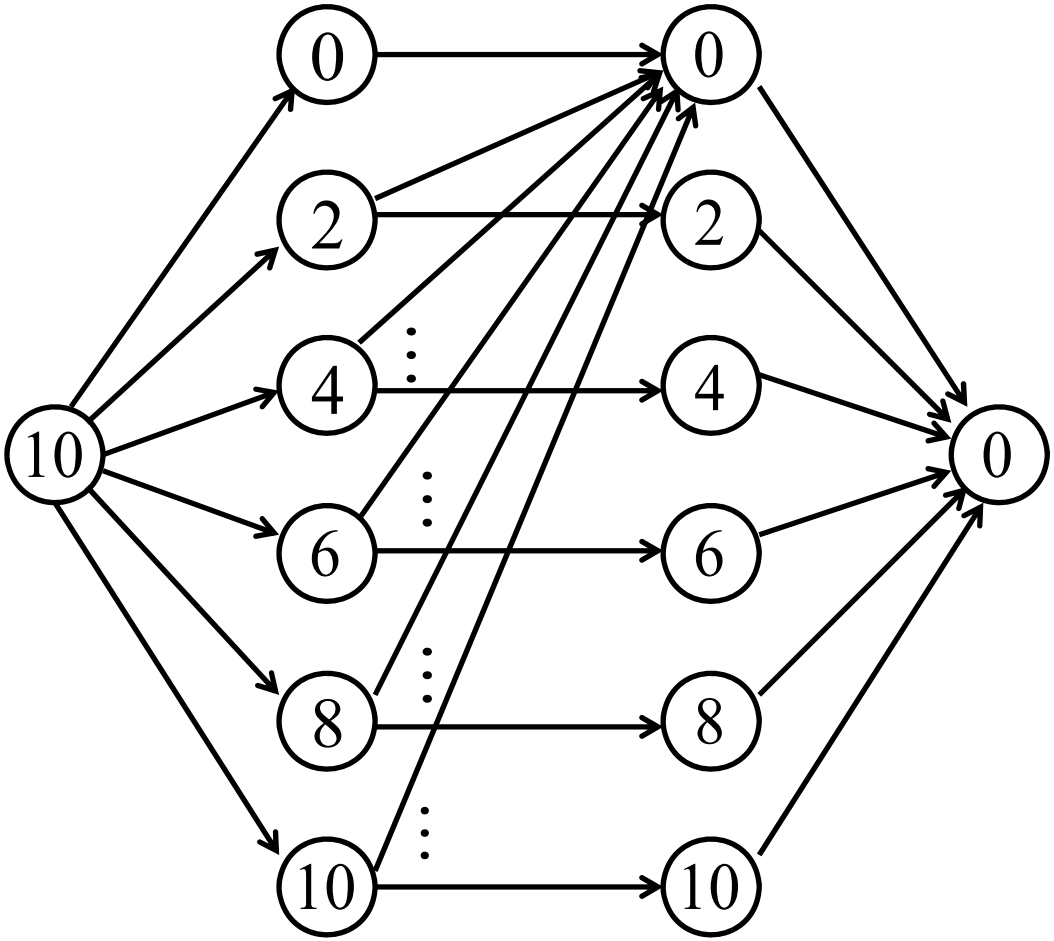
将连续区间 $[0,10]$ 离散化为有限点集 $\{0,2,4,6,8,10\}$

等价于

求右边

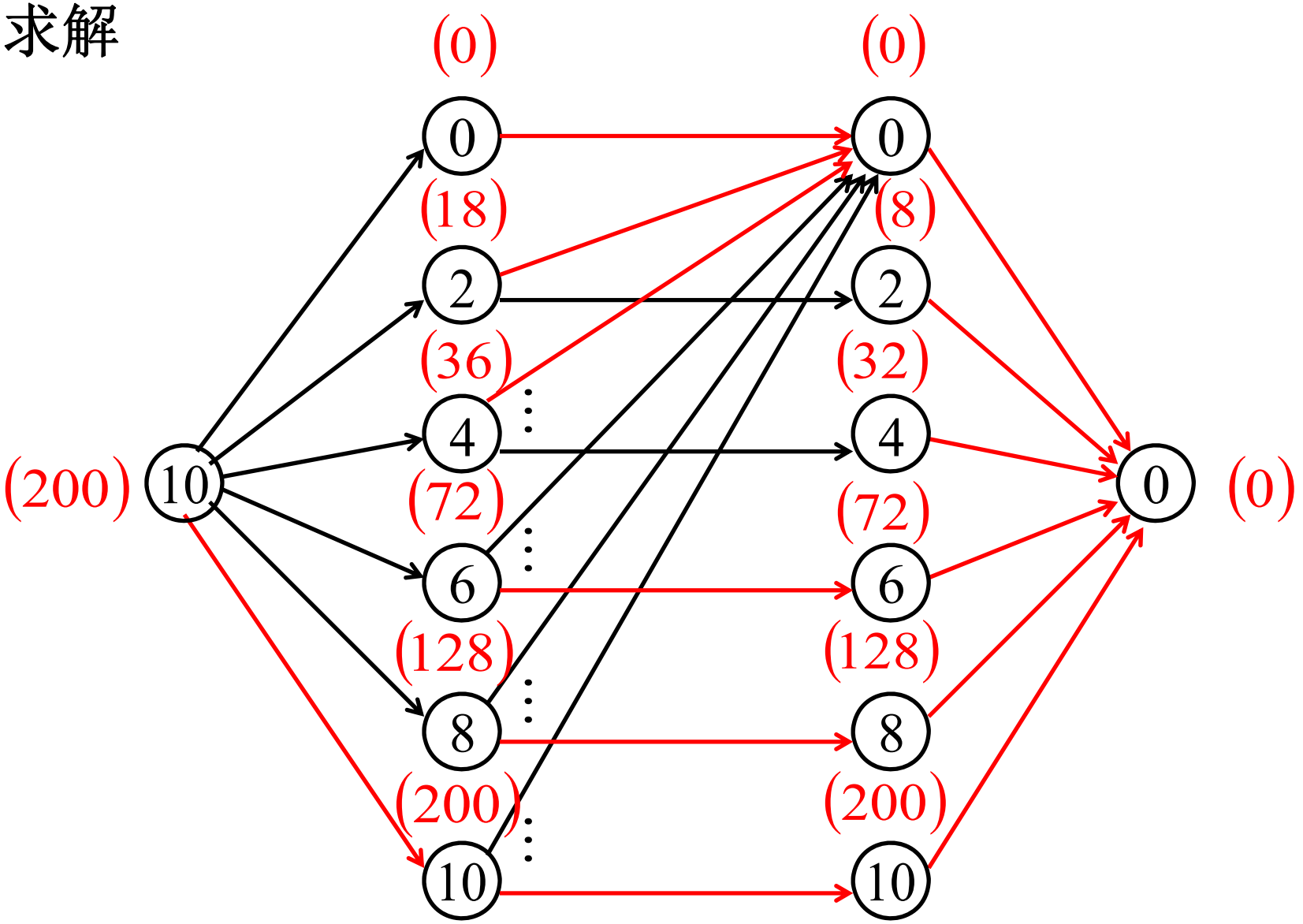
的最长

路问题



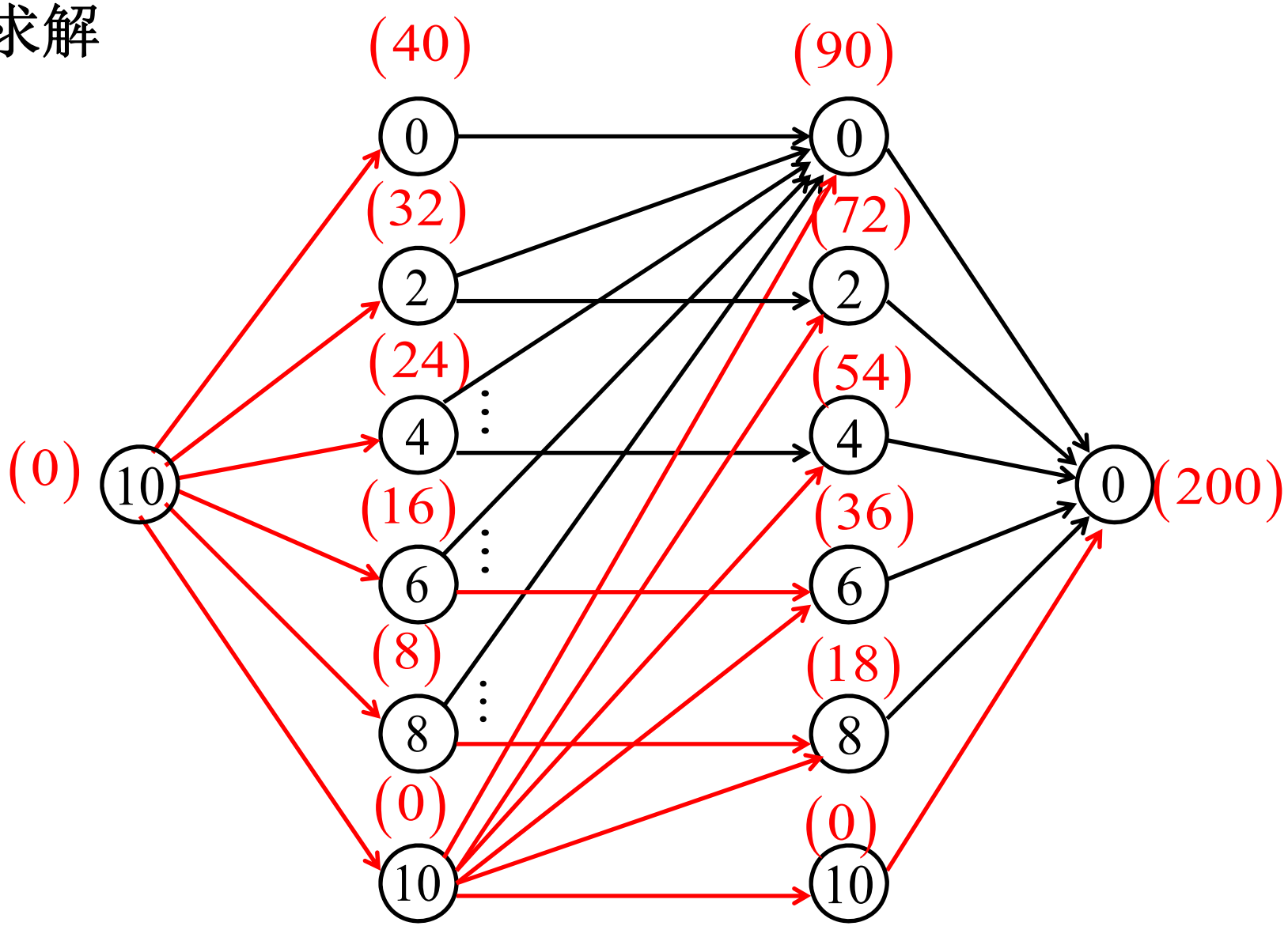
$4\times(s_1-s_2)$　　$9\times(s_2-s_3)$　　$2\times(s_3-s_4)^2$

逆序求解



$4\times(s_1-s_2)$ $9\times(s_2-s_3)$ $2\times(s_3-s_4)^2$

顺序求解



$4\times(s_1-s_2)$ $9\times(s_2-s_3)$ $2\times(s_3-s_4)^2$

背包问题

10吨卡车，装三种货物，单位重量和价值如下

货物编号	1	2	3
单位重量（吨）	3	4	5
单位价值	4	5	6

如何装载使总价值最大？

静态优化模型

$$\max \ 4x_1 + 5x_2 + 6x_3$$

$$\text{s.t.} \ \ 3x_1 + 4x_2 + 5x_3 \le 10$$

$$x_1, x_2, x_3 \ \text{非负整数}$$

这是一个**NP**难问题

建立序贯决策模型

阶段　三个决策阶段，顺序决策三种货物装载量

状态　k 阶段开始可用装载量 s_k，$s_1=10$

决策　k 阶段实际装载量 x_k

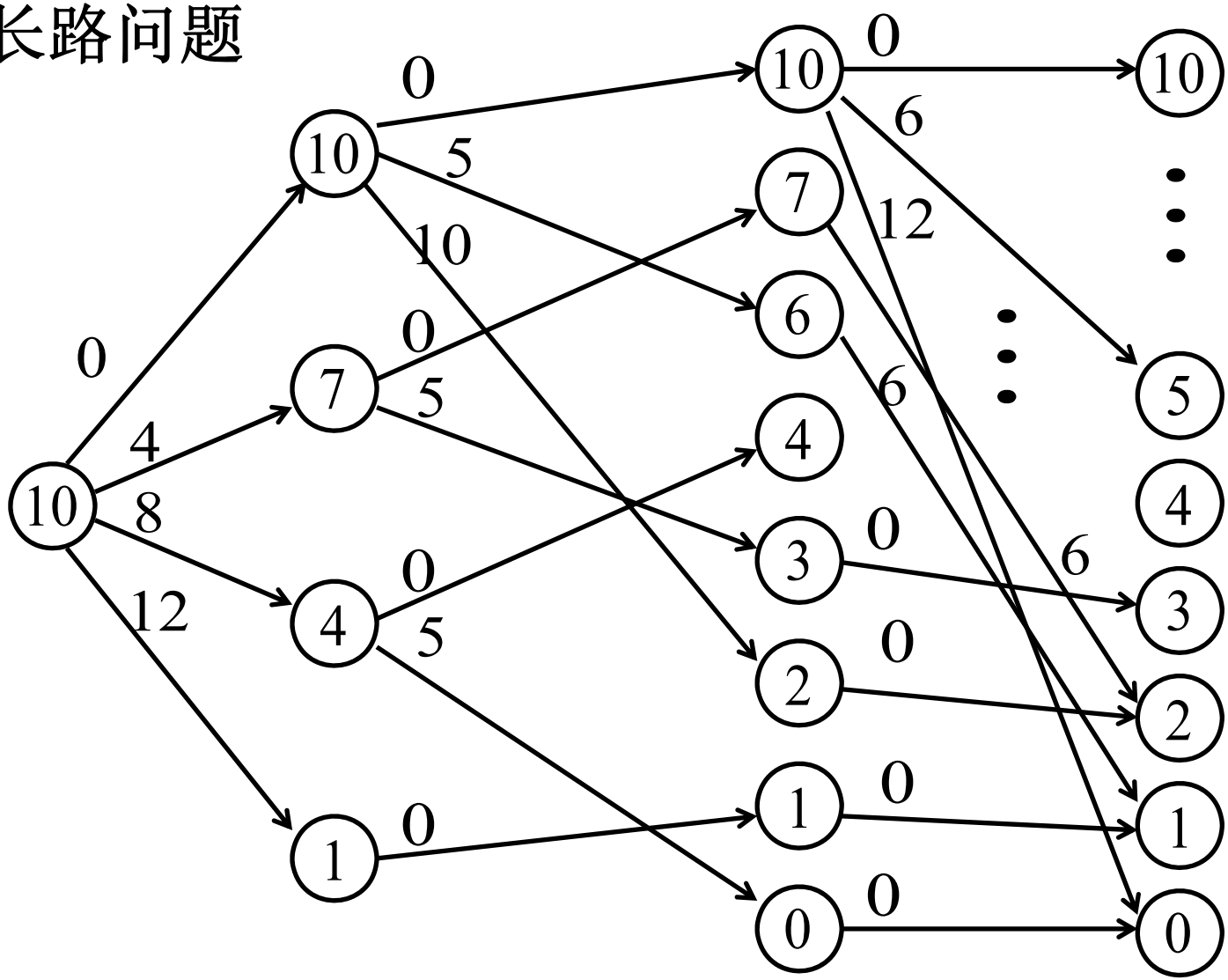
状态集　$0\le s_k\le s_1,\ s_4\ge 0$

允许决策集　$3x_1\le s_1, 4x_2\le s_2, 5x_3\le s_3$，$x_k$ 非负整数

阶段指标　$4x_1,\ 5x_2,\ 6x_3$

状态转移方程　$s_2=s_1-3x_1, s_3=s_2-4x_2, s_4=s_3-5x_3$

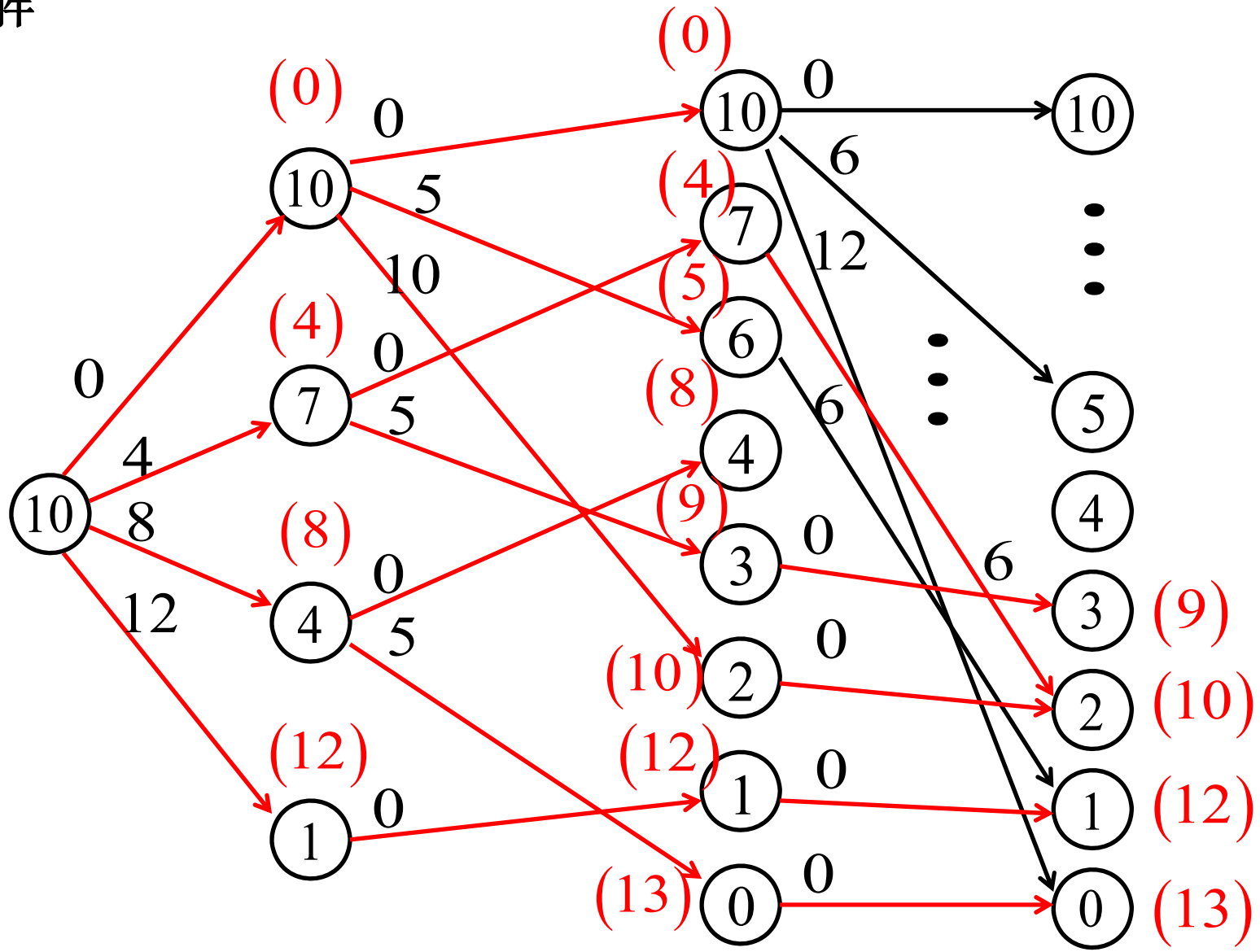
转化成最长路问题



状态转移　$s_2 = s_1 - 3x_1,\ s_3 = s_2 - 4x_2,\ s_4 = s_3 - 5x_3$

阶段指标　$4x_1,\quad 5x_2,\quad 6x_3$

顺序求解



最优解 $x_1^* = 2, x_2^* = 1, x_3^* = 0$ 最优目标 $\overline{f}_3(0) = 13$

多维状态

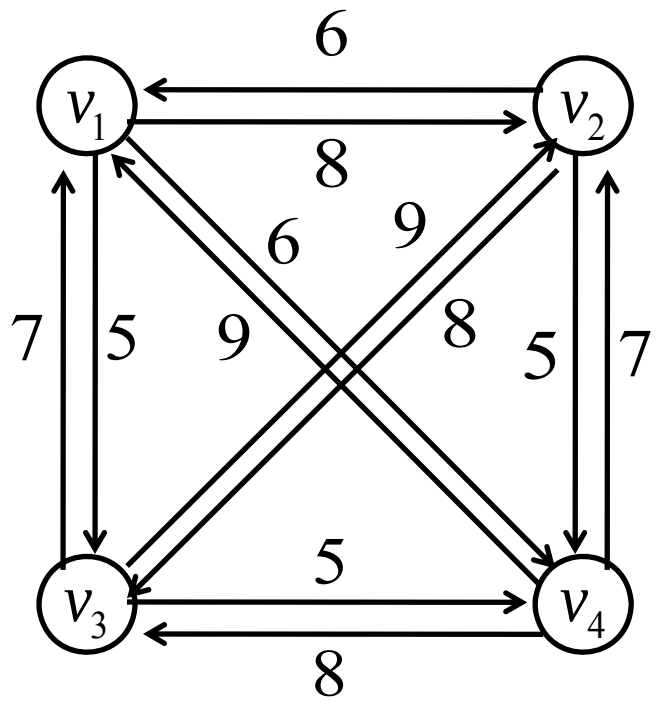
例、二维背包问题

$$\max \ 4x_1 + 7.5x_2 + 6x_3$$
$$\text{s.t.} \quad 0.2x_1 + 0.3x_2 + 0.25x_3 \le 5$$
$$0.3x_1 + 0.5x_2 + 0.2x_3 \le 9$$
$$x_1, x_2, x_3 \text{ 非负整数}$$

主要区别：每阶段开始时状态是二维向量，分别表示两个不等式的可用资源量

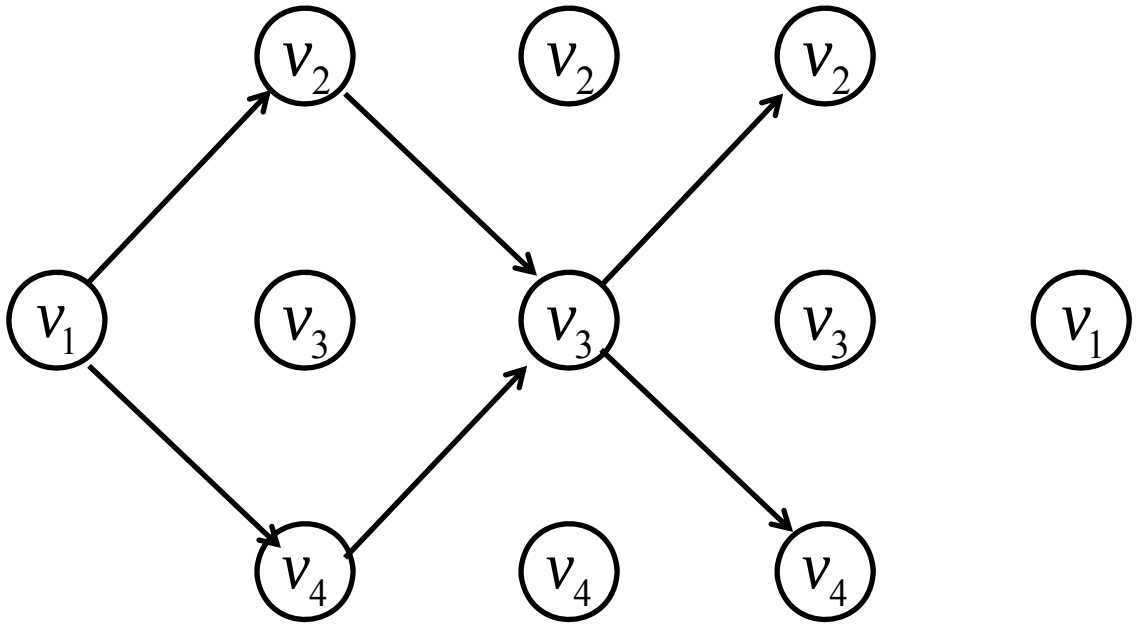
旅行商问题

下图四个城市，任何两个城市间均有道路相连，路程由图中数字所示。从 v_1 出发，找出一条经过其他每个城市最终回到 v_1 的最短路线



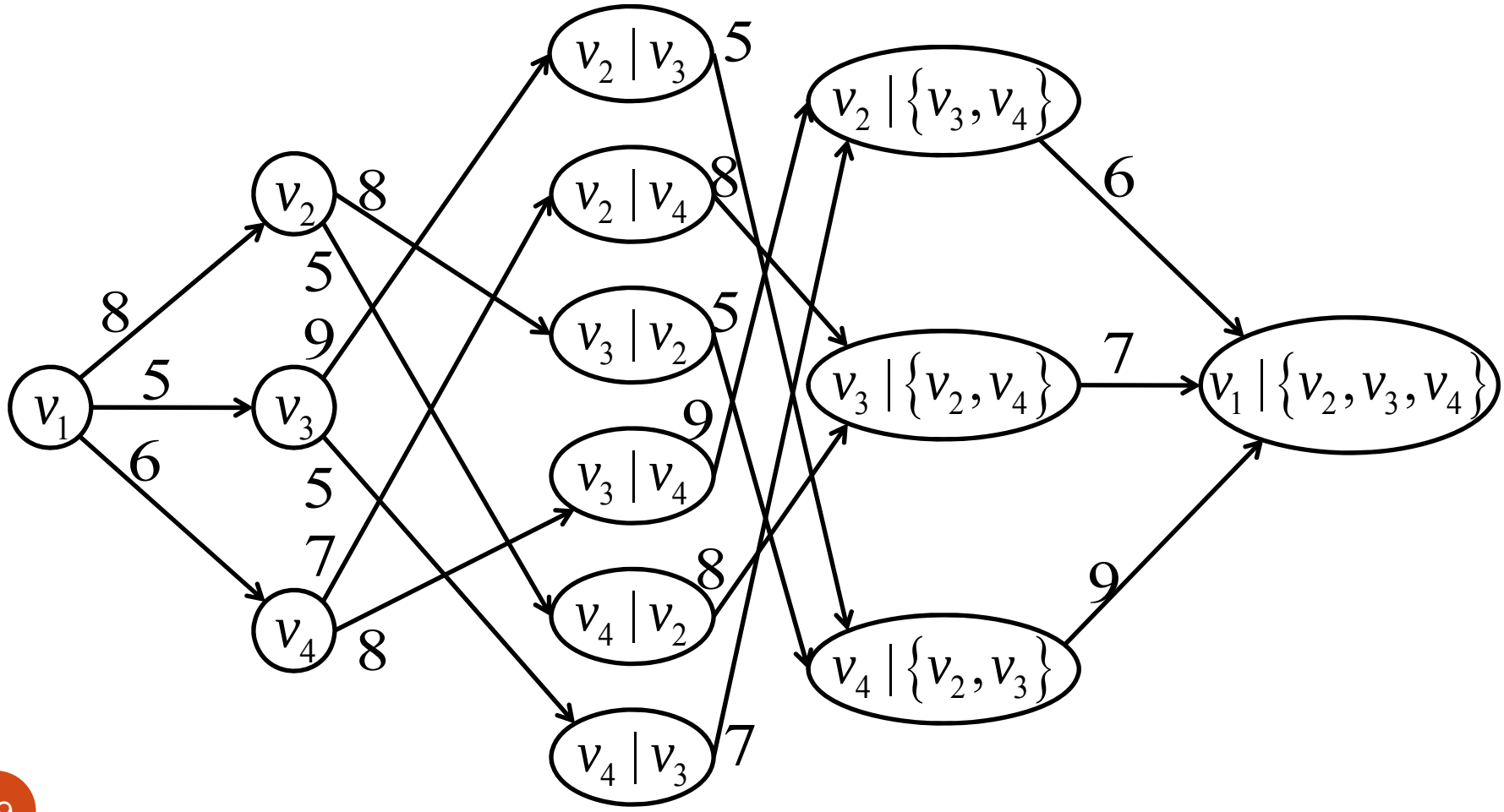
转换成多阶段决策问题

直接以城市为状态会存在后效性问题，如下图，v_3 处的容许决策取决于第二阶段状态值

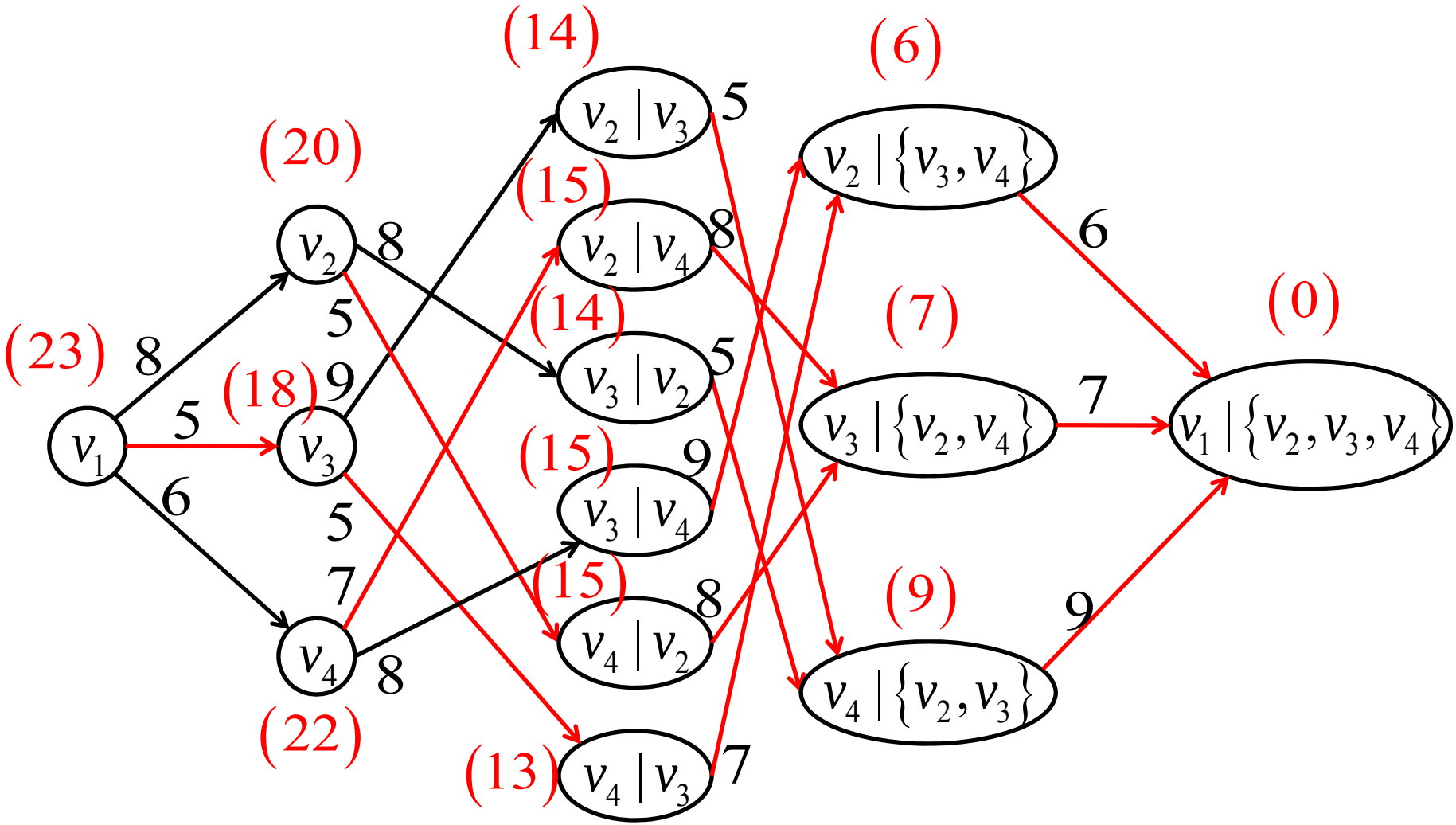


无后效性的状态设置方法

$s_{k,i}$： k 步到达城市 v_i ，$1 \le k \le 3, 2 \le i \le 4$



逆推求解

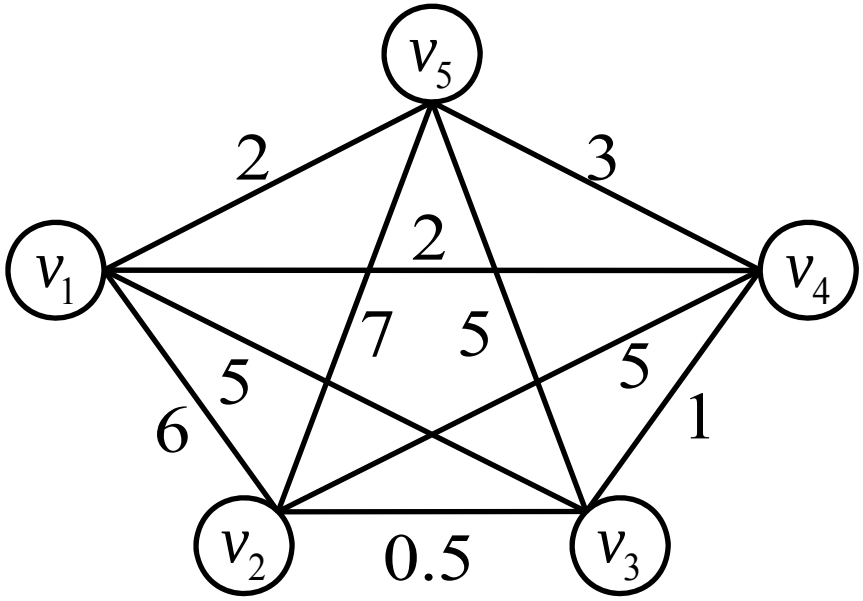


最优路径：$v_1 \to v_3 \to v_4 \to v_2 \to v_1$　最优路程：23

迭代求解方法

例、不定期（无阶段划分）最短路问题

下图五个城市，任何两个城市间均有道路相连，往返路程一样，由图中数字所示。求每个城市到第五个城市的最短路线和最短路程



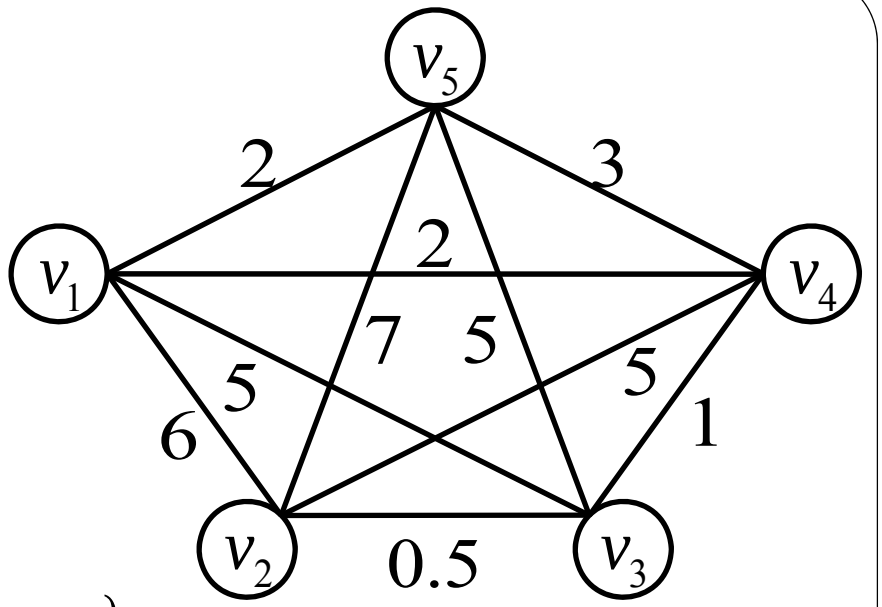
状态与状态集 $S=\{v_i, i=1,\cdots,5\}$

决策与决策集

$$U(s)=\{v_i, i=1,\cdots,5\},\ \forall s\in S$$

策略与策略集

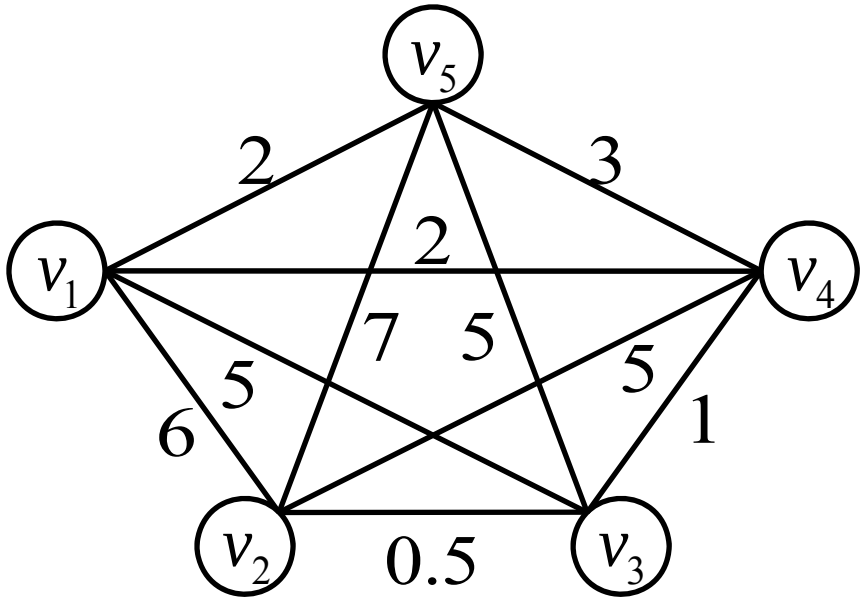
$$P=\{u_i\in U(v_i), i=1,\cdots,5\}$$



状态转移函数 $T(s,u)\in S,\ \forall s\in S, u\in U(s)$

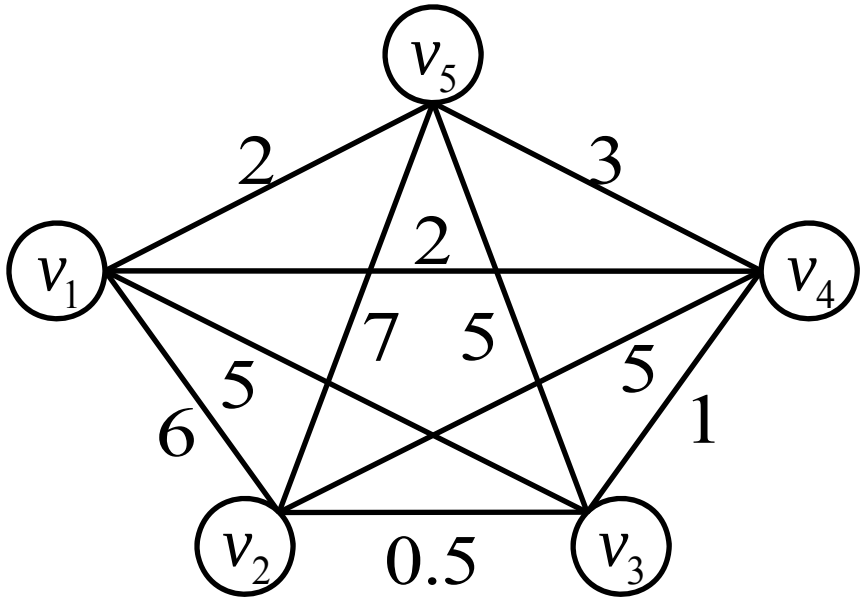
阶段指标函数 $d(s,u),\ \forall s\in S, u\in U(s)$ （$d(s,s)=0$）

问题 对每个初始状态，以极小化阶段指标函数之和为目标，确定一个能够转移到末状态 v_5 的最优策略



马尔可夫（**Markov**）性（或无后效性）

以任何状态为初始状态进行决策所产生的效果，不受如何到达这个状态的决策影响

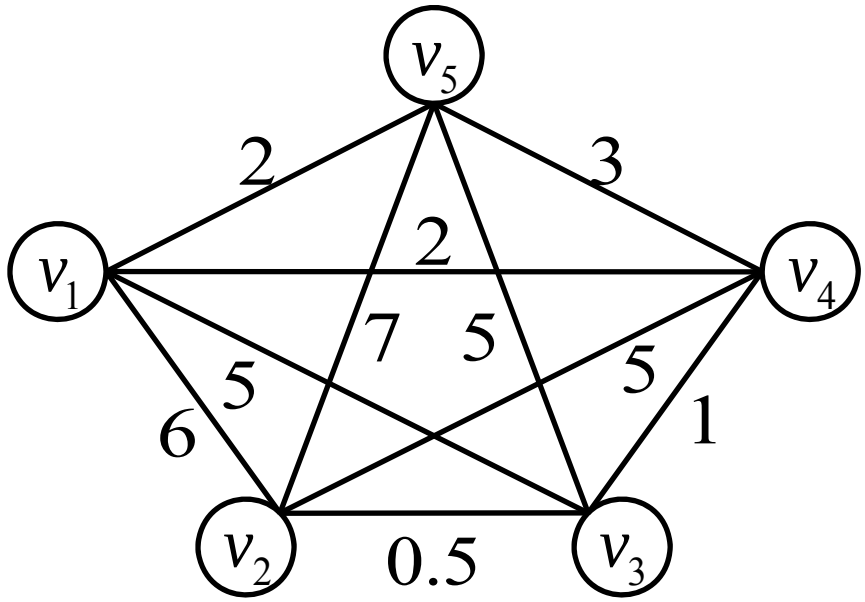


保证我们可以分解问题来“分而治之”

最优性原理

若 v_j 在自 v_i 到 v_5 的最优路线上，那么这条路线上自 v_j 到 v_5 的部分就是自 v_j 到 v_5 的最优路线

理由：马尔科夫性



问题的最优解所包含的子问题的解也是最优的

定义各点到目的地的<u>最优值函数</u> $f(s), \forall s \in S$

根据最优性原理，最优值函数一定满足<u>最优值方程</u>

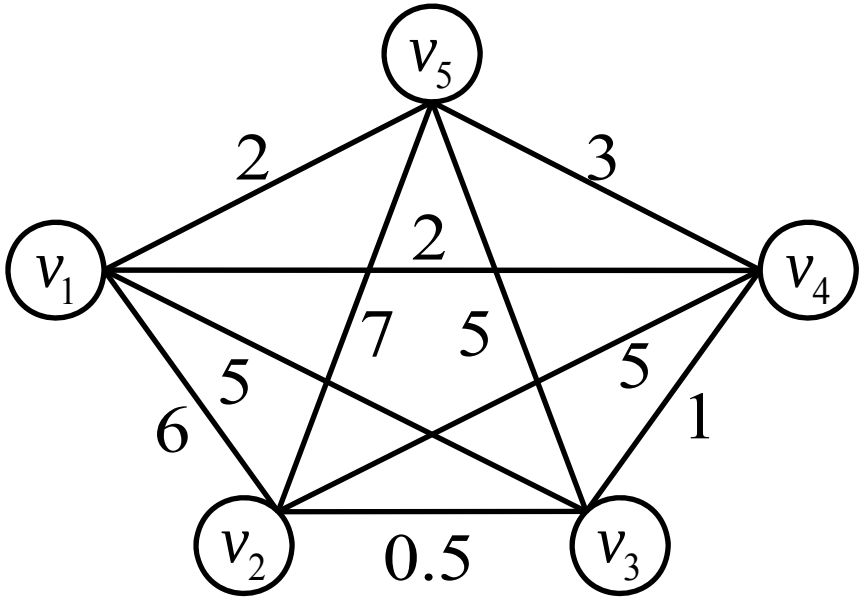
$$f(s) = \min_{u \in U(s)} \left\{ d(s,u) + f\left(T(s,u)\right) \right\}, \ \forall s \in S$$

<u>动态规划的核心是解最优值方程</u>

值迭代方法

（无阶段划分）<u>最短路问题</u>

下图五个城市，任何两个城市间均有道路相连，往返路程一样，由图中数字所示。求每个城市到第五个城市的最短路线和最短路程



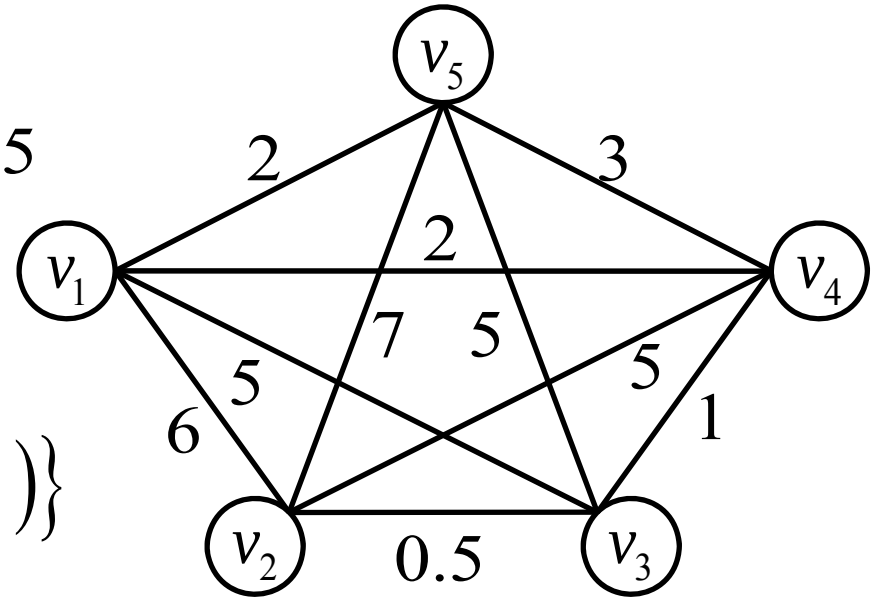
最优值方程：$f(s)=\min_{u\in U(s)}\left\{d(s,u)+f\left(T(s,u)\right)\right\},\ \forall s\in S$

值迭代方法

首先取 $f_1(v_i)=d(v_i,v_5),1\le i\le 5$

然后对 $k>1$ 按下述公式迭代

$$f_{k+1}(v_i)=\min_{1\le j\le 5}\left\{d(v_i,v_j)+f_k(v_j)\right\}$$



如果到某个 k 满足 $f_{k+1}(v_i)=f_k(v_i),\ \forall 1\le i\le 5$

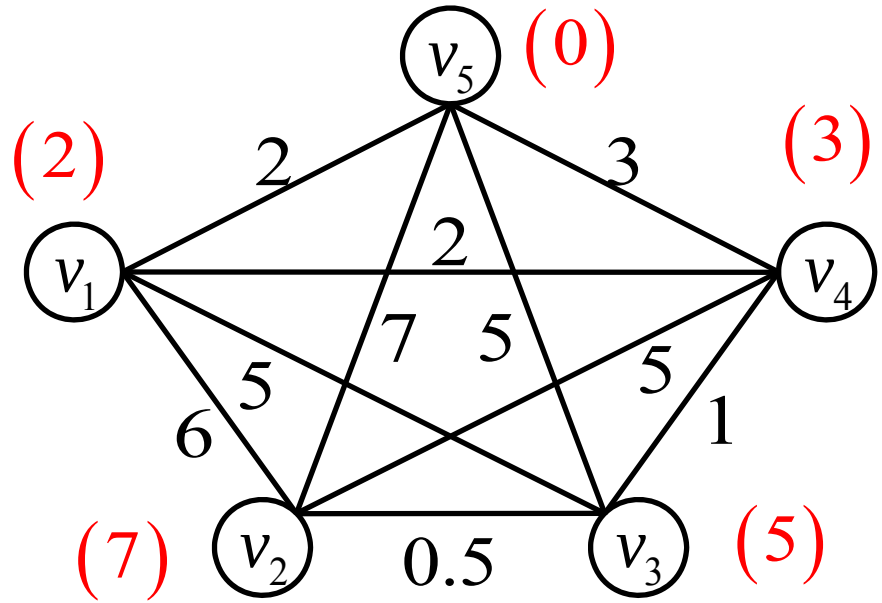
那么就成立 $f_k(v_i)=\min_{1\le j\le 5}\left\{d(v_i,v_j)+f_k(v_j)\right\},\ \forall 1\le i\le 5$

于是得到最优值方程的解 $f(v_i)=f_k(v_i),\ \forall 1\le i\le 5$

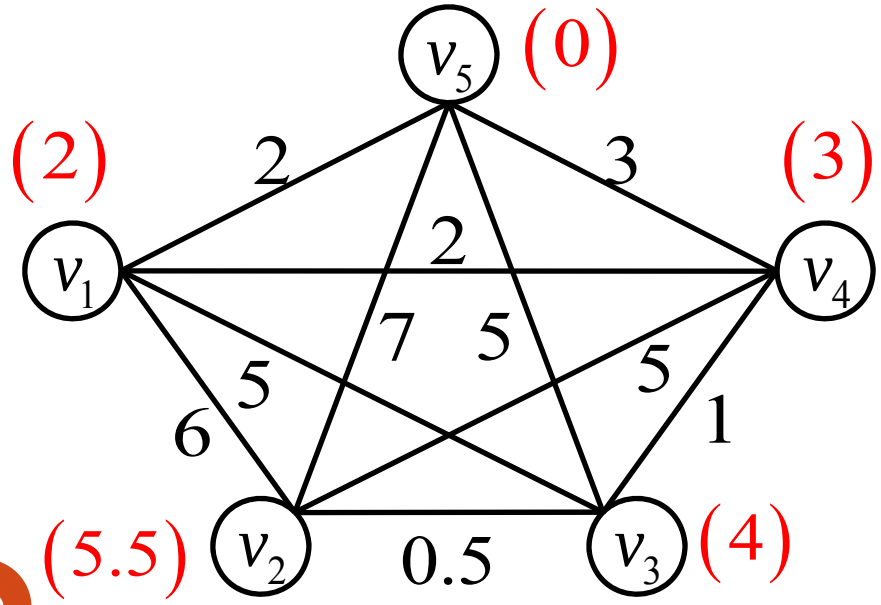
最优路线可以由最优值函数确定

直接在图上进行值迭代

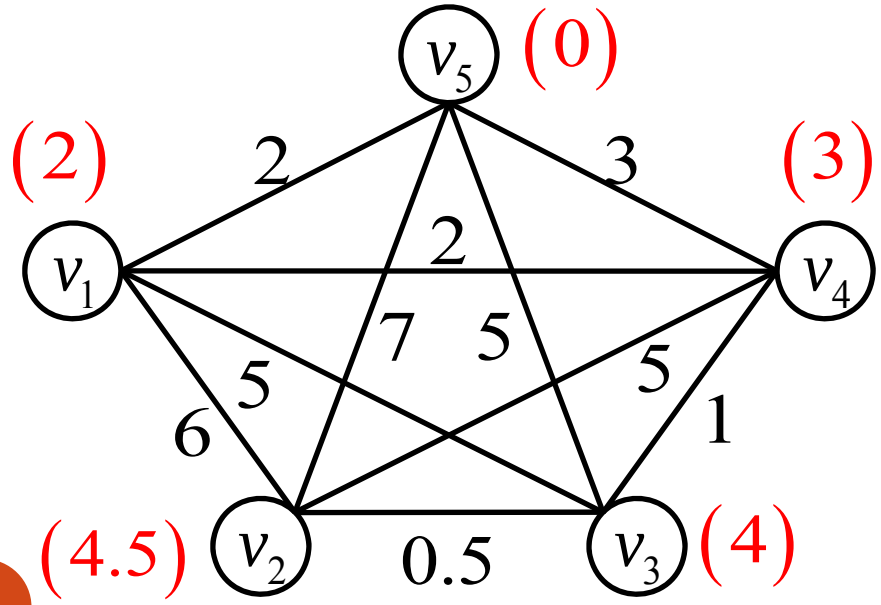
$$f_1(v_i) = d(v_i, v_5)$$



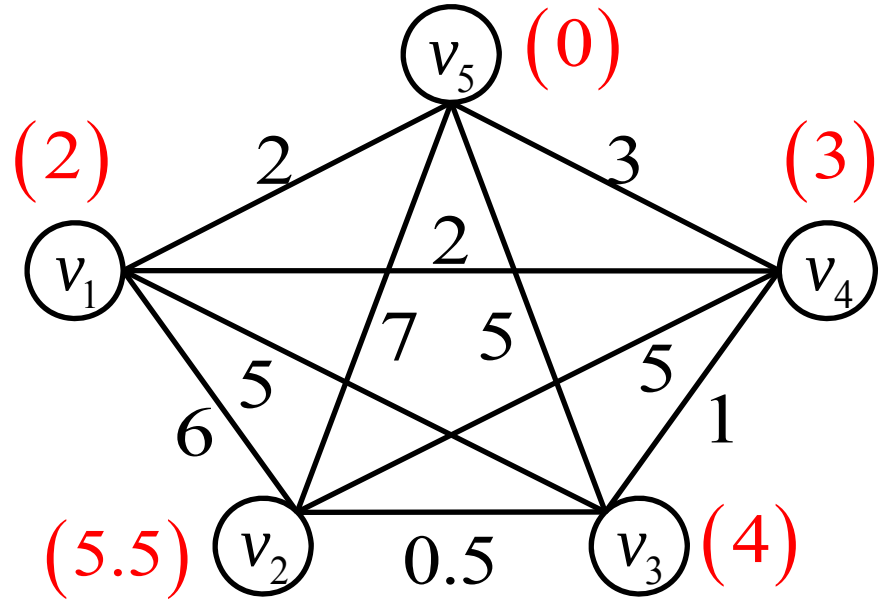
$$f_2(v_i) = \min_{1 \le j \le 5} \{ d(v_i, v_j) + f_1(v_j) \}$$



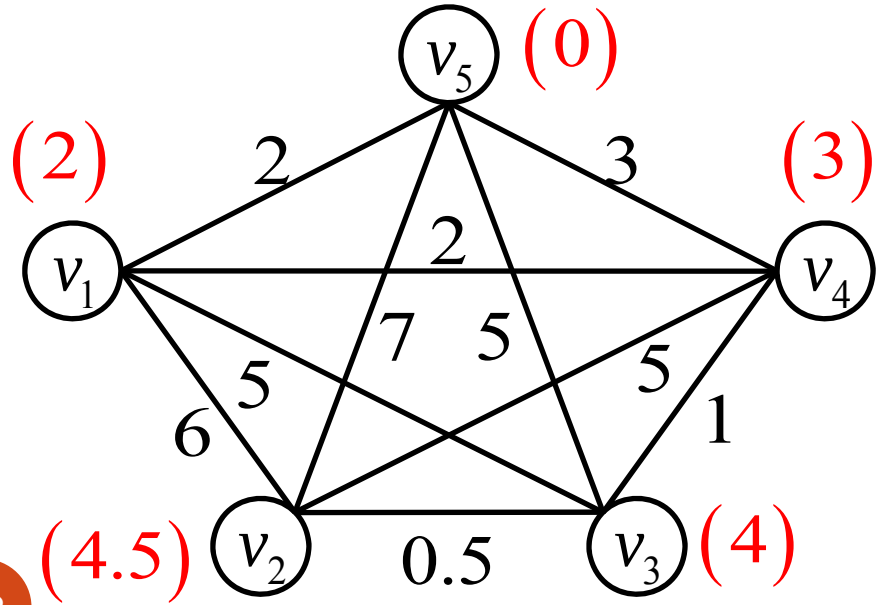
$$f_3(v_i) = \min_{1 \le j \le 5} \{ d(v_i, v_j) + f_2(v_j) \}$$



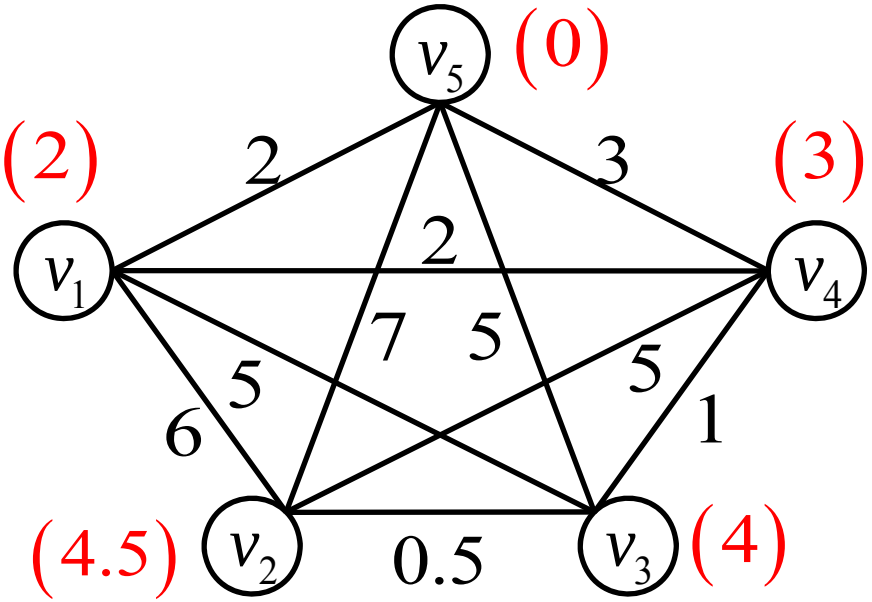
$f_2(v_i)$



$$f_4(v_i) = \min_{1 \le j \le 5} \left\{ d(v_i, v_j) + f_3(v_j) \right\}$$

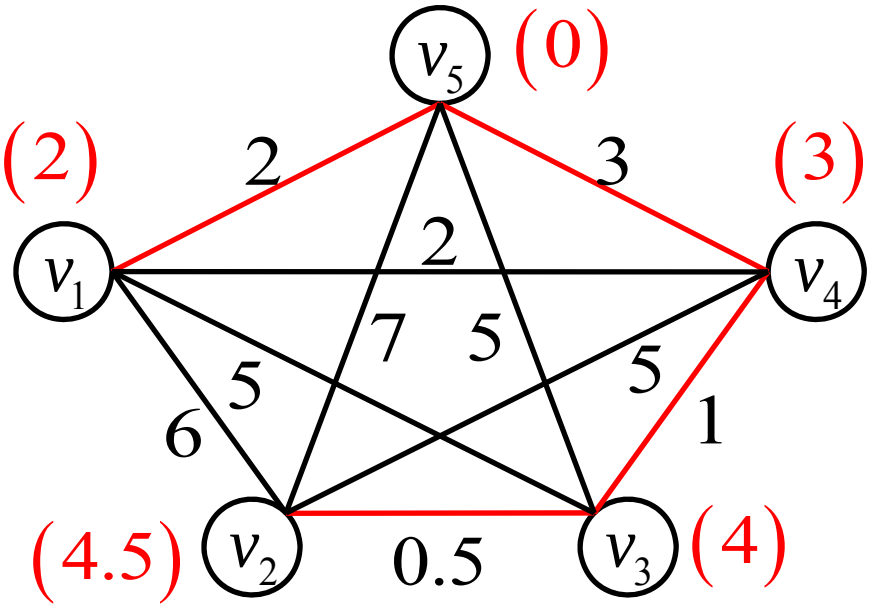


$f_3(v_i)$



$$\Rightarrow \quad f_4(v_i) = f_3(v_i), \forall i$$

根据最优值确定最优路线



最优路线：　$v_1 \to v_5$　　最优值：　2

$v_2 \to v_3 \to v_4 \to v_5$　　4.5

$v_3 \to v_4 \to v_5$　　4

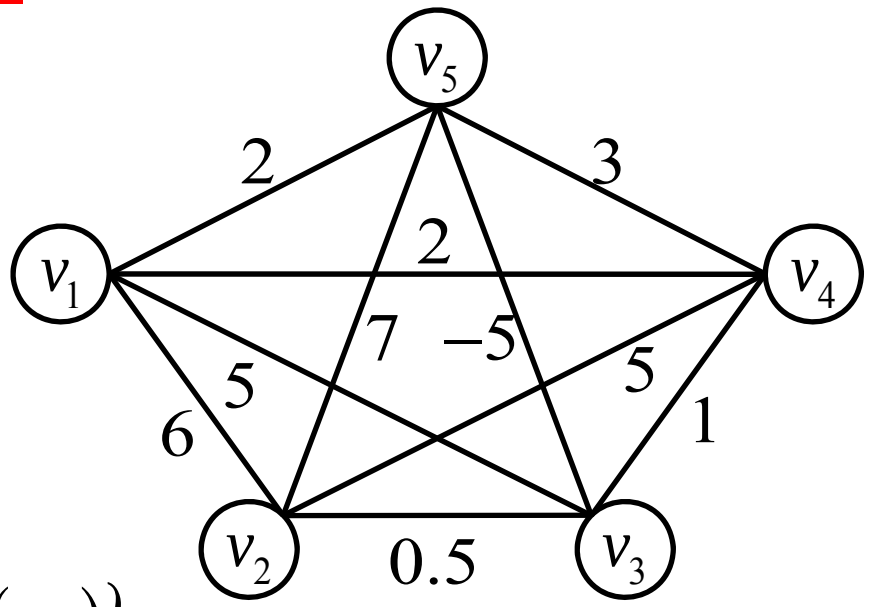
$v_4 \to v_5$　　3

保证最短路问题的值迭代法收敛的充要条件：

<u>没有总路程之和小于零的回路</u>

例如，右图不满足条件

$v_3 \to v_4 \to v_5 \to v_3 \quad \Rightarrow \quad -1$



理由： $f_1(v_i) = d(v_i, v_5)$

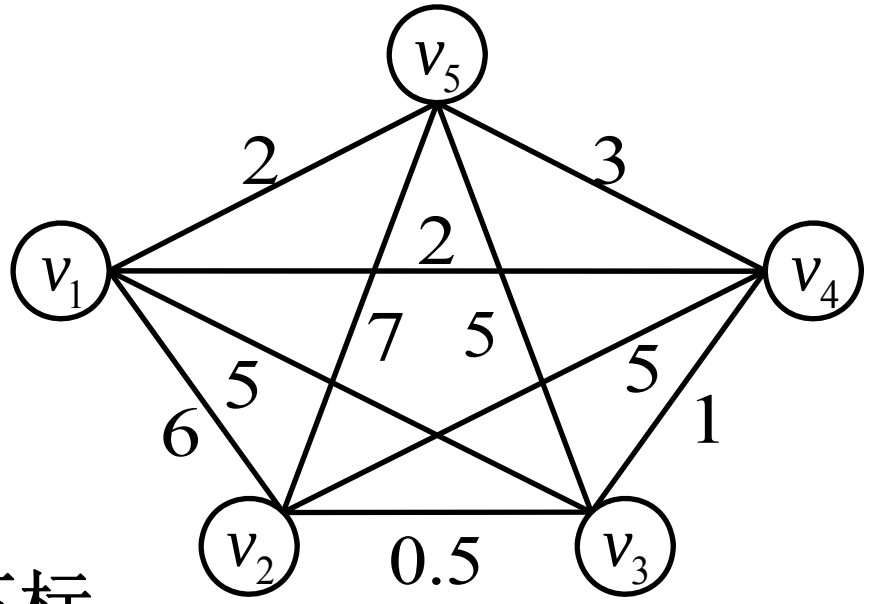
$$f_{k+1}(v_i) = \min_{1 \le j \le 5} \left\{ d(v_i, v_j) + f_k(v_j) \right\}$$

$f_{k+1}(v_i)$ 就是 v_i 经过不超过 k 个中转城市到达目的

$\Rightarrow$ 地的最短路，城市数有限，最后必重复，没有负回路的重复不可能减少任何点的总路程

策略迭代方法

无回路策略



一个策略就是在每个点的某种决策构成的集合，写成

$$P=\{p(v_i),1\le i\le 5\}$$

其中 $p(v_i)$ 表示 v_i 后面城市的下标

没有决策在一个回路上的策略称为无回路的策略

例如：下面是无回路的策略

$$p(v_1)=4,\ p(v_2)=4,\ p(v_3)=4,\ p(v_4)=5,\ p(v_5)=5$$

下面不是无回路的策略

$$p(v_1)=4,\ p(v_2)=1,\ p(v_3)=2,\ p(v_4)=3,\ p(v_5)=5$$

策略迭代法（策略空间迭代法）

任意选取一个无回路策略 $P_1 = \{p_1(v_i), 1 \le i \le 5\}$

求解线性方程组 $f_k(v_i) = c_{i\,p_k(v_i)} + f_k\left(v_{p_k(v_i)}\right),\ \forall 1 \le i \le 4$

$$f_k(v_5) = 0$$

得 $\hat{f}_k(v_i),\ \forall 1 \le i \le 5$（无回路的策略保证方程有唯一解）

利用 $\hat{f}_k(v_i),\ \forall 1 \le i \le 5$ 确定改进的策略

$$P_{k+1} = \{p_{k+1}(v_i), 1 \le i \le 5\}$$

改进途经

$$c_{i\,p_{k+1}(v_i)} + \hat{f}_k\left(v_{p_{k+1}(v_i)}\right) = \min_{1 \le j \le 5}\left\{c_{ij} + \hat{f}_k(v_j)\right\},\ \forall 1 \le i \le 4$$

重复上述过程直到策略不改变

有（无）回路的策略和方程组的关系

有回路策略

$$p(v_1)=v_4,\ p(v_2)=v_1$$
$$p(v_3)=v_2,\ p(v_4)=v_3$$
$$p(v_5)=v_5$$

$$\Rightarrow$$

$$f_k(v_1)=c_{14}+f_k(v_4)$$
$$f_k(v_2)=c_{21}+f_k(v_1)$$
$$f_k(v_3)=c_{32}+f_k(v_2)$$
$$f_k(v_4)=c_{43}+f_k(v_3)$$

无解

无回路策略

$$p(v_1)=4,\ p(v_2)=4$$
$$p(v_3)=4,\ p(v_4)=5$$
$$p(v_5)=5$$

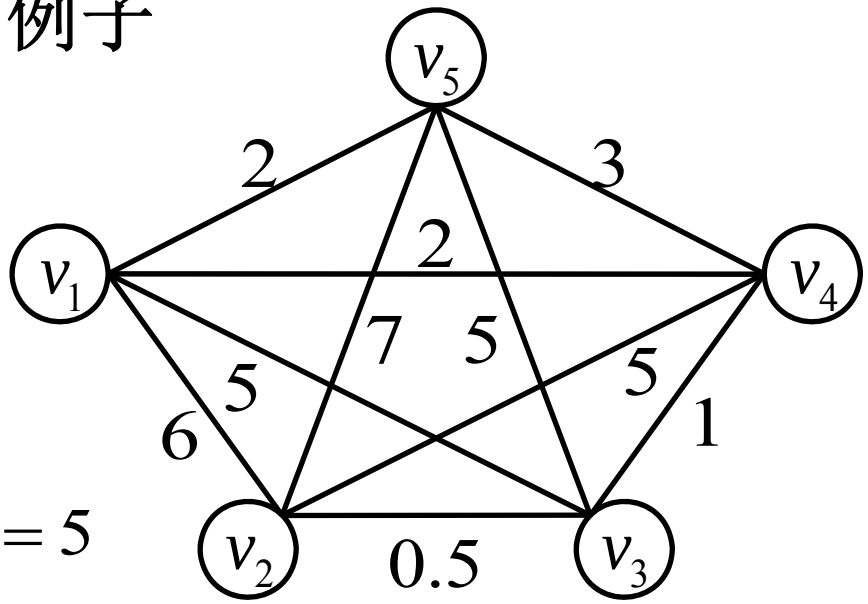
$$\Rightarrow$$

$$f_k(v_1)=c_{14}+f_k(v_4)$$
$$f_k(v_2)=c_{24}+f_k(v_4)$$
$$f_k(v_3)=c_{34}+f_k(v_4)$$
$$f_k(v_4)=c_{45}+f_k(v_5)$$

$$f_k(v_5)=0\Rightarrow \hat{f}_k(v_4)\Rightarrow \hat{f}_k(v_1),\hat{f}_k(v_2),\hat{f}_k(v_3)$$

用策略迭代法解最短路问题的例子

$$p_1(v_1)=4,\ p_1(v_2)=4$$
$$p_1(v_3)=4,\ p_1(v_4)=5$$
$$p_1(v_5)=5$$



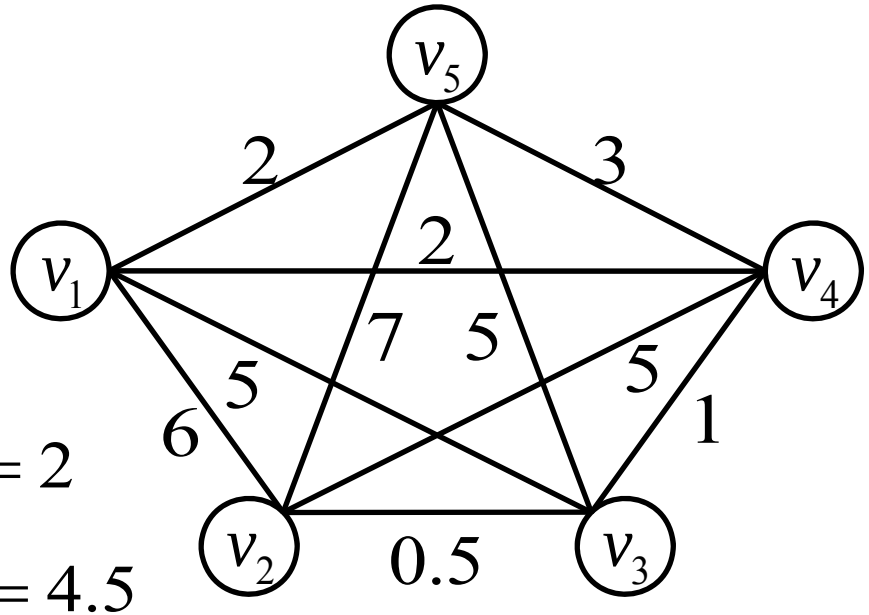
$$\begin{aligned} f_1(v_1) &= 2+f_1(v_4) \\ f_1(v_2) &= 5+f_1(v_4) \\ f_1(v_3) &= 1+f_1(v_4) \\ f_1(v_4) &= 3+f_1(v_5) \end{aligned} \Rightarrow \begin{aligned} \hat{f}_1(v_1) &= 5 \\ \hat{f}_1(v_2) &= 8 \\ \hat{f}_1(v_3) &= 4 \\ \hat{f}_1(v_4) &= 3 \end{aligned}$$

$$c_{i\,p_2(v_i)}+\hat{f}_1\left(v_{p_2(v_i)}\right)=\min_{1\le j\le 5}\left\{c_{ij}+\hat{f}_1(v_j)\right\},\ \forall 1\le i\le 4$$

$$\Rightarrow\quad p_2(v_1)=5,\ p_2(v_2)=3,\ p_2(v_3)=4,\ p_2(v_4)=5,\ p_1(v_5)=5$$

继续迭代

$$p_2(v_1)=5,\ p_2(v_2)=3$$
$$p_2(v_3)=4,\ p_2(v_4)=5,\ p_2(v_5)=5$$



$$\begin{aligned} f_2(v_1) &= 2+f_2(v_5) \\ f_2(v_2) &= 0.5+f_2(v_3) \\ f_2(v_3) &= 1+f_2(v_4) \\ f_2(v_4) &= 3+f_2(v_5) \end{aligned} \Rightarrow \begin{aligned} \hat{f}_2(v_1) &= 2 \\ \hat{f}_2(v_2) &= 4.5 \\ \hat{f}_2(v_3) &= 4 \\ \hat{f}_2(v_4) &= 3 \end{aligned}$$

$$c_{i\,p_3(v_i)}+\hat{f}_2\left(v_{p_3(v_i)}\right)=\min_{1\le j\le 5}\left\{c_{ij}+\hat{f}_2\left(v_j\right)\right\},\ \forall 1\le i\le 4$$

$$\Rightarrow\quad p_3(v_1)=5,\ p_3(v_2)=3,\ p_3(v_3)=4,\ p_3(v_4)=5,\ p_3(v_5)=5$$

$P_3=P_2$　停止

最优策略

$$p_3(v_1)=5,\ p_3(v_2)=3,\ p_3(v_3)=4,\ p_3(v_4)=5,\ p_3(v_5)=5$$

最优路径：

$v_1 \to v_5$

$v_2 \to v_3 \to v_4 \to v_5$

$v_3 \to v_4 \to v_5$

$v_4 \to v_5$

同值迭代法结果一样

策略迭代和值迭代

策略迭代法（Policy Iteration method）是动态规划中求最优策略的基本方法之一。它借助于动态规划基本方程，交替使用“策略评估”（Policy evaluation）和“策略改进”（Policy improvement）两个步骤，求出逐次改进的、最终收敛于最优策略序列。

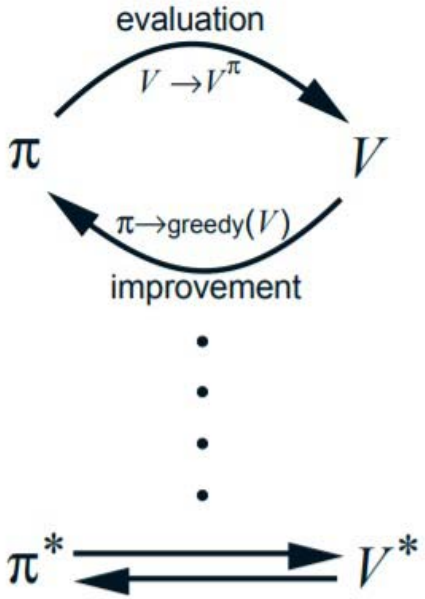
不难发现，如果想知道最优的策略，就需要能够准确估计值函数。然而想准确估计值函数，又需要知道最优策略，值函数才能够估计准确。所以实际上这是一个“鸡生蛋还是蛋生鸡”的问题。

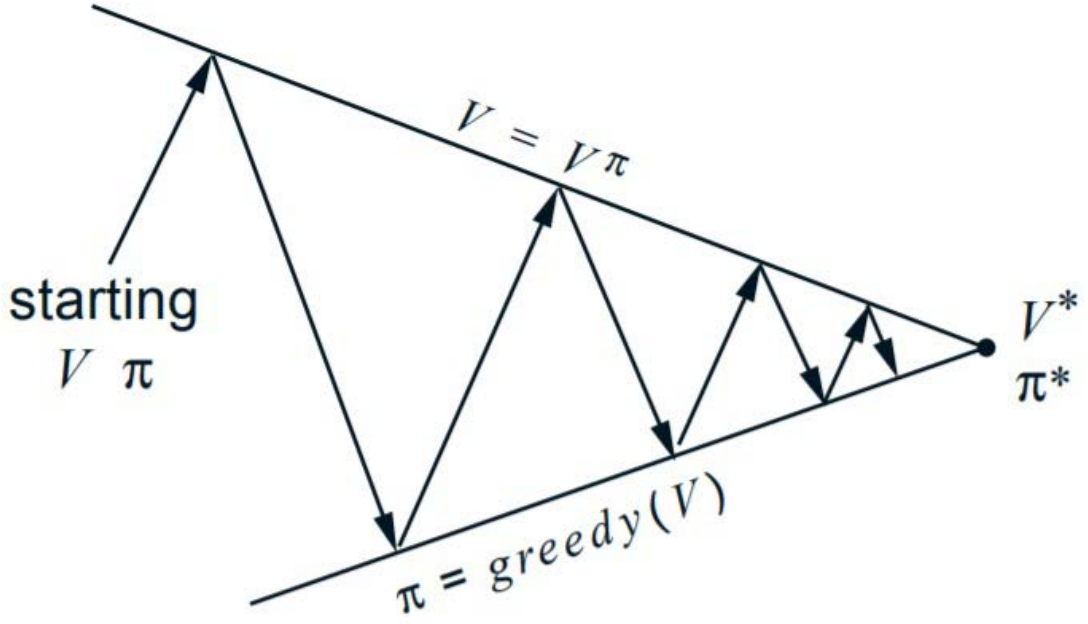
好在我们设计算法使得左右脚互踩可以飞升。

一般策略迭代法的思路可总结为以下步骤：

1. （Policy evaluation）以某种策略π开始，计算当前策略下的值函数v_π(s)。对于简单的决策问题，我们会得到一个线性方程组，求解该方程组可以得到值函数v_π(s)。
2. （Policy improvement）利用这个值函数，更新策略，得到π*。一般而言，我们可以贪婪地选取行为π* =greedy(v_π)，使得后继状态价值增加最多
3. 再用这个策略π*继续前行，更新值函数，得到v′_π。重复上面的过程，一直到v_π(s)不再发生变化。

策略迭代法的收敛性证明一般基于巴拿赫不动点（Banach Fixed Point）定理给出。需要首先证明上述迭代过程本质是一个压缩映射，然后证明最优解是一个不动点，最后由巴拿赫不动点定理导出结论。





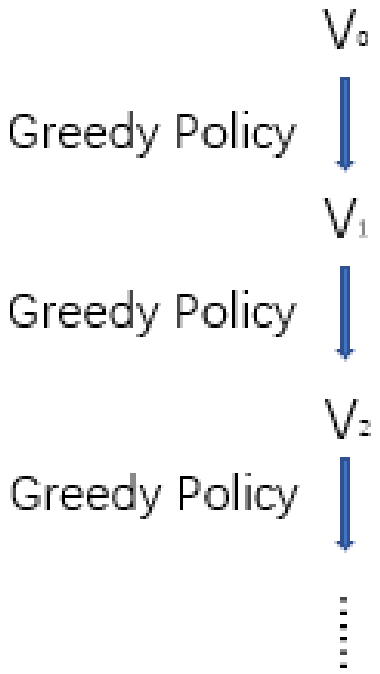
策略评估经常是非常耗时的，因为我们要等到当前策略下的值函数收敛。与其等到策略评估收敛，我们可以容忍一些错误，在某个点上截断策略评估过程，并在近似的价值函数上执行策略改进步骤。

现在，考虑一下极端情况，即我们只应用策略评估循环的一次迭代，然后改进策略。换句话说，我们可以把VI看作是PI的一个极端情况，即只用当前策略的一次迭代来估计当前策略下的值函数，完成一个非常简化的策略评估步骤。

考虑到值函数的单调有界性，我们同样能证明值迭代的收敛性。

因为每轮策略评估只迭代一次，所以价值函数很可能没有收敛到某个策略，即此时的价值函数没有对应的有实际意义的策略

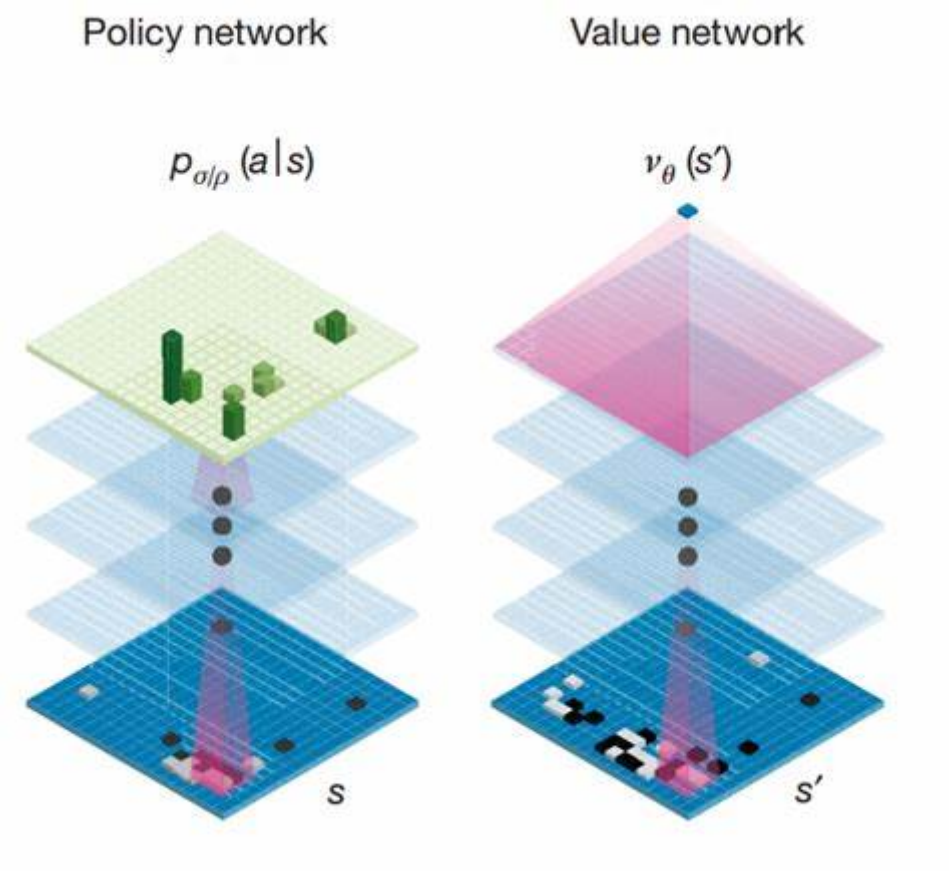
Intermediate value functions may not correspond to any policy



Remark 1: 一个决策问题的最优策略可能不唯一。使用不同的初始策略，通过策略迭代最后收敛到的最优策略可以是不一样。不过这些策略对应的值函数（value function）应该是一样的。

Remarks 2: 有的问题策略迭代快，有的问题值迭代快。这个问题比较复杂，我们暂时不在这门课上讨论了

现在有很多新方法来近似估计策略函数和值函数，深度强化学习是当前强有力的人工智能工具，强烈推荐了解一下



AlphaGo