MATH 3011 运筹学 **秋季 2023**

Lecture 7: 动态规划

Lecturer: 陈士祥 *Scribes:* 陈士祥

1 动态规划介绍

动态规划是一种算法设计技巧，广泛应用于解决优化问题，尤其是那些可以分解为重叠子问题的复杂问题。动态规划主要思想：将问题分解为子问题，并重复使用已有的结论（即写出递归式）。

1.1 最短路问题

Example 7.1 最短路径问题的一个例子，回顾 *Dijkstra* 算法。

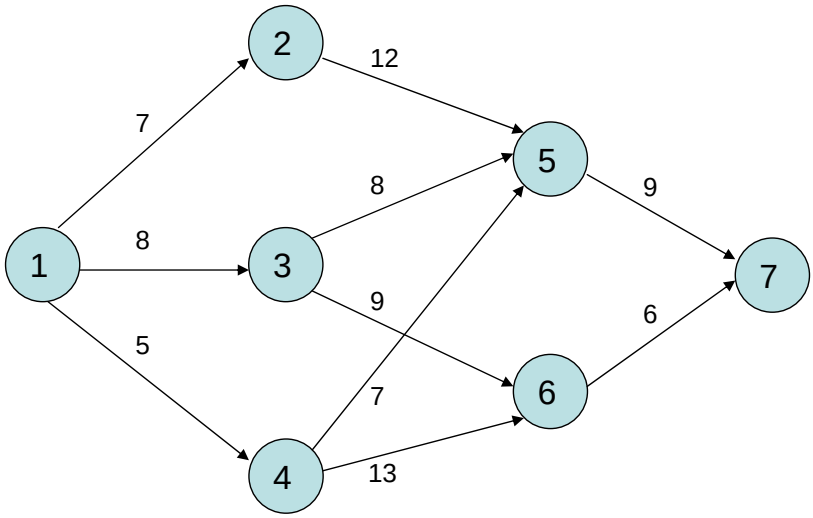

图 7.1: 最短路图

对问题进行分阶段考虑。与上节课的 *Dijkstra* 算法稍有区别，对于图 *7.1*中的有向图，我们可以分为如图 *7.2*中的 $\{0,1,2,3\}$ 共 *4* 个阶段，每个阶段有不同的顶点，分别为 $\{0\},\{2,3,4\},\{5,6\},\{7\}$。令 $f_i(x_i)$ 表示不同阶段中所有点到起点的最短路值。开始阶段，$f_0(x_0)=0$. 阶段 *1*，我们有 $f_1(x_1=2)=$

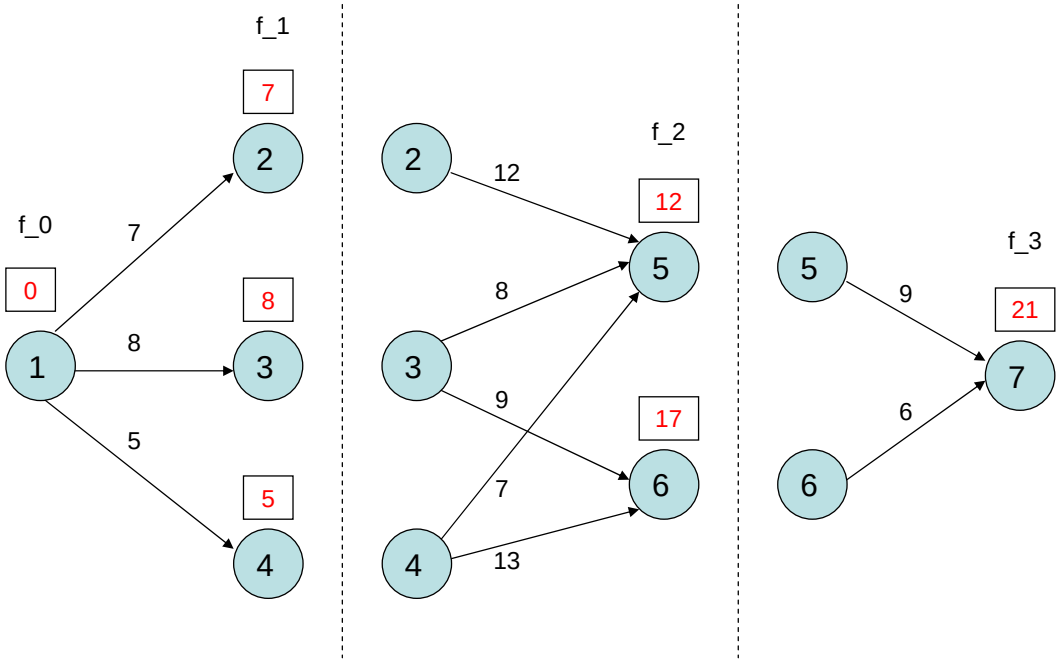

图 7.2: 有向图最短路分阶段

$7, f_1(x_1 = 3) = 8, f_1(x_1 = 4) = 5$. 对于阶段 *2*，我们有

$$f_2(x_2) = \min_{(x_1,x_2)\in E}\{d(x_1, x_2) + f_1(x_1)\}$$

故，$f_2(x_2 = 5) = f_1(x_1 = 4) + d(4, 5) = 12, f_2(6) = f_1(x_1 = 3) + d(x_1 = 3, x_2 = 6) = 17$.

我们可以写出递归方程 *(Recursive Equation)*

$$\begin{cases} f_i(x_i) = \min\limits_{(x_{i-1},x_i)\in E}\{d(x_{i-1}, x_i) + f_{i-1}(x_{i-1})\}, \quad i = 1, 2, 3, \\ f_0(x_0) = 0. \end{cases}$$

总的来说，动态规划的几个关键要素：

- Stages 阶段
- Alternatives 选择
- States 状态
- Recursive Equations 递归方程、状态转移方程

通过下面的背包问题，我们进一步说明这几个要素。

1.2 背包/货物装载 (Knapsack/Cargo-Loading) 问题

01 背包问题：最基本的背包问题就是 01 背包问题（01 knapsack problem）：一共有 n 件物品，第 i（i 从 1 开始）件物品的重量为 w_i，价值为 r_i。在总重量不超过背包承载上限 W 的情况下，能够装入背包的最大价值是多少?
当然，我们可以使用枚举法，将所有的情况列举出来，最多有 2^n 种情况。
也可以写成如下的整数线性规划 (Integer Linear Programming)：

$$\begin{aligned} \max \quad & z = \sum_{i=1}^{n} r_i m_i \\ \text{s.t.} \quad & \sum_{i=1}^{n} w_i m_i \le W \\ & m_1, \cdots, m_n \in \{0, 1\} \end{aligned}$$

m_i 表示第 i 个物品是否装入背包。

完全背包问题：完全背包问题，就是每个物品可以有无穷个。
A general (n-Items, W-LB) knapsack problem can be represented by the following Integer Linear Programming:

$$\begin{aligned} \max \quad & z = \sum_{i=1}^{n} r_i m_i \\ \text{s.t.} \quad & \sum_{i=1}^{n} w_i m_i \le W \\ & m_1, \cdots, m_n \in \mathbb{Z}_+ \cup \{0\} \end{aligned}$$

在最差情况下，求解整数线性规划需要指数时间。

对于 01 背包问题，我们有如下动态规划解法：

- stage: 在 0/1 背包问题的动态规划中，阶段表示问题求解过程中的决策点或迭代。每个阶段对应于选择将要放入背包或拒绝的一项物品。通常，阶段的数量等于要考虑的物品数量。例如，如果有 5 个物品，动态规划过程将有 5 个阶段。
- state：状态表示在每个阶段定义问题所需的信息。在 0/1 背包问题中，每个阶段的状态通常包括以下组件：
  1. 当前考虑的物品（例如，物品 i）。
  2. 在该阶段背包的剩余容量（还可以添加多少重量到背包中）。

  因此，状态可以表示为一对 (i, w)，其中 $'i'$ 是当前物品的索引，$'w'$ 是该阶段背包的剩余容量。状态有助于跟踪子问题及其解决方案，随着阶段的进行而不断更新。
- alternative: 选择：选择指的是每个阶段可以做出的选择或决策。在 0/1 背包问题中，每个阶段有两种选择：
  1. 选择当前物品并将其添加到背包中，前提是其重量不超过剩余容量。

2. 拒绝当前物品，继续下一个物品，不将其添加到背包中。

我们用 $f(i,w)$ 表示前 i 个物品，放入最大承重 w 的背包最大价值。我们还有 $f(i,0)=0, i=0,1,...,n.$ 那么，根据上述分析，对于 $n \geq i > 1, W \geq w \geq 0$, 我们有如下递归方程：

$$f(i,w) = \max\{f(i-1,w), f(i-1,w-w_i)+r_i, (w \geq w_i)\}$$

其中，$f(i-1,w)$ 表示不放入物品 i 的价值；$f(i-1,w-w_i)+r_i$ 表示放入 i 的价值。使用动态规划可以将复杂度降至 $\mathcal{O}(nW)$.

注 7.1 *背包问题的判定形式（即是否能找到放入方式，使得不超过承重 W 的情况下达到价值 V?）是 NP-complete 问题。为什么是 NP 问题？对于 n 个物品，我们输入的是两个数组，分别表示每个物品的价值和重量，在计算机中表示他们的比特数和 n 线性相关。对于另外的参数，即总重量 W，需要 $m = \log W$ 的位数来表示. 因此，m 才是输入规模的一部分，表示 W 需要 $\mathcal{O}(2^m)$ 比特数。所以 $O(n \cdot W) = O(n2^m)$, 并非真正的多项式时间。*

完全背包问题 我们仍用 $f(i,w)$ 表示前 i 个物品，放入最大承重 w 的背包最大价值。01 背包只有两种情况即取 0 件和取 1 件，而这里是取 0 件、1 件、2 件... 直到超过限重（$k > w/w_i$）

1. Stage i is represented by item i, $i = 1, 2, \cdots, n$.
2. The alternatives at stage i are represented by m_i, the number of units of item i included in the knapsack. It follows that $k = 0, 1, \cdots, \lfloor \frac{w}{w_i} \rfloor$.

我们有边界条件：$f(i,0)=0$, $i=0,1,...,n$, 递归方程如下

$$f(i,w) = \max_{k=0,1,\cdots,\lfloor \frac{w}{w_i} \rfloor} \{r_i k + f(i-1, w-w_i k)\}, \quad i = 1, \cdots, n, 1 \leq w \leq W.$$

Example 7.2 *计算容量为 7 的 01 或者完全背包问题的最大价值。*

Item i	1	2	3
w_i	2	3	1
r_i	31	47	15

1.3 设备更新模型 (Equipment Replacement Model)

假设我们考虑的是一个跨度为 n 年的设备更新问题。每年初我们需要决定是否保留当前设备再使用一年或者更换一个新的设备。令 $r(t)$, $c(t)$, $s(t)$ 分别表示一台 t-年龄设备的年营业收入，年运营成本及其

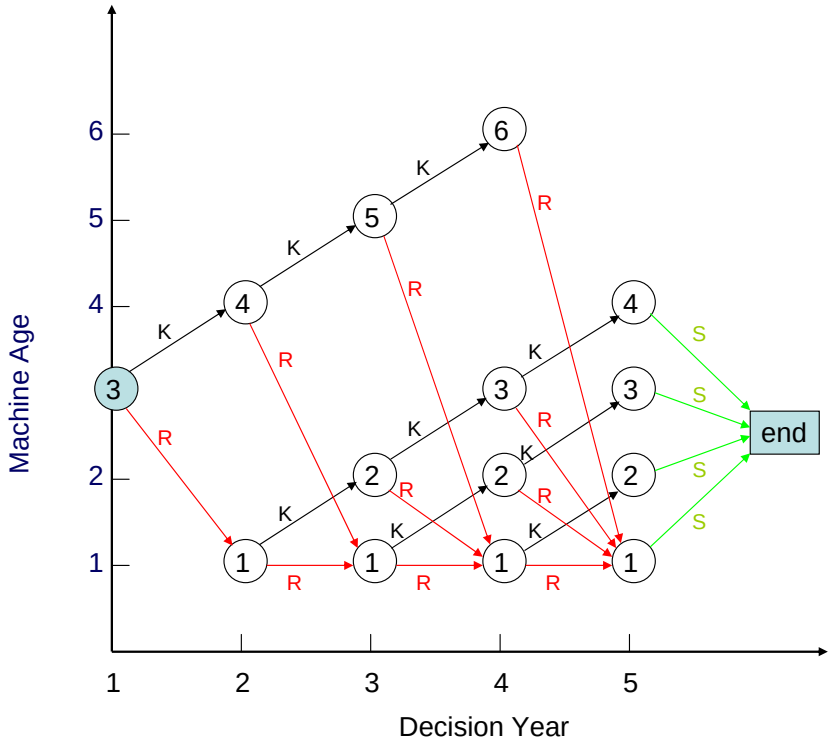

图 7.3: 设备更新模型图示

残余价值。另外，在规划期内的任何一年购置一台新设备的成本是 I. 设某公司现有一台 3 年龄的设备，需制定一个未来 4 年 ($n = 5$) 的设备更新最优策略。该公司还规定 6 年龄的设备必须得以更换。一台新设备的成本是 $100,000. 下表给出的是设备更新问题的相关数据，其中 t 是机器年龄，$r(t), c(t), s(t)$ 分别表示 t 年龄机器的年营业收入，年运营成本及残余价值。

- Stage i is represented by year i where $i = 1, \cdots, n$.
- The alternatives at stage i are either *keeping* or *replacing* the machine at the start of year i.
- The state at stage i is the age of the machine at the start of year i.
- Recursive equation:

$$f_i(t) = \max \begin{cases} r(t) - c(t) + f_{i+1}(t+1), t \leq 6 & \text{if Keeping} \\ r(0) - c(0) + s(t) - I + f_{i+1}(1), & \text{if Replacing} \end{cases}$$
$$f_{n+1}(t) = s(t)$$

  where $f_i(t)$ is defined as the maximum net income for years $i, i+1, \cdots, n$ by given a t-year-old machine at the start of year i.

表 7.1: 设备更新问题数据表

t	$r(t)$	$c(t)$	$s(t)$
0	20,000	200	–
1	19,000	600	80,000
2	18,500	1,200	60,000
3	17,200	1,500	50,000
4	15,500	1,700	30,000
5	14,000	1,800	10,000
6	12,200	2,200	5,000

与之前的背包问题所不同的是，设备更新问题中，我们的递归边界条件为 $f_{n+1}(t) = s(t)$, 即从最后一项向前递归这通常称为后向归纳法 (backward induction). 而之前的背包问题为前向递归 (forward induction).

作业 7.1 设现有一台 *2* 年龄的设备，另规定 *5* 年龄的设备必须更换。在规划期购置新设备的成本分别是

$$(p_1, p_2, p_3, p_4, p_5) = (100, 105, 110, 115, 120).$$

试构建表格 *7.3*中设备更新的动态规划模型并求其最优更新策略。

表 7.2: 五年期设备更新

设备年龄 t	残余价值 v_t	运行费用 c_t
0	-	30
1	50	40
2	25	50
3	10	75
4	5	90
5	2	-

表 7.3: 设备更新价值表格

作业 7.2 使用动态规划求解下面的问题。

问题 *1*: 如图 *7.4* 有一个移动机器人，可以在二维矩阵平面中移动，其移动的起点位于左上角，每次移动只能向右或者向下，其移动的终点是右下角，在这个矩阵形的平面中每个位置都有一个数值，代表机器人在经过这个区域时需要付出的代价。我们的目标就是，在这个平面中，找一条代价最小的路径，并

且给出最小代价路径。

问题 *2*: 若机器人从左上角出发，只能向右或者向下移动，起点坐标设为 $(0, 0)$，需要达到右下角，坐标设为 (M, N)。请问有多少种不同的路径（无需考虑每格代价）？

2	3	1
1	9	1
6	4	2

图 7.4: 机器人坐标代价表