动态规划

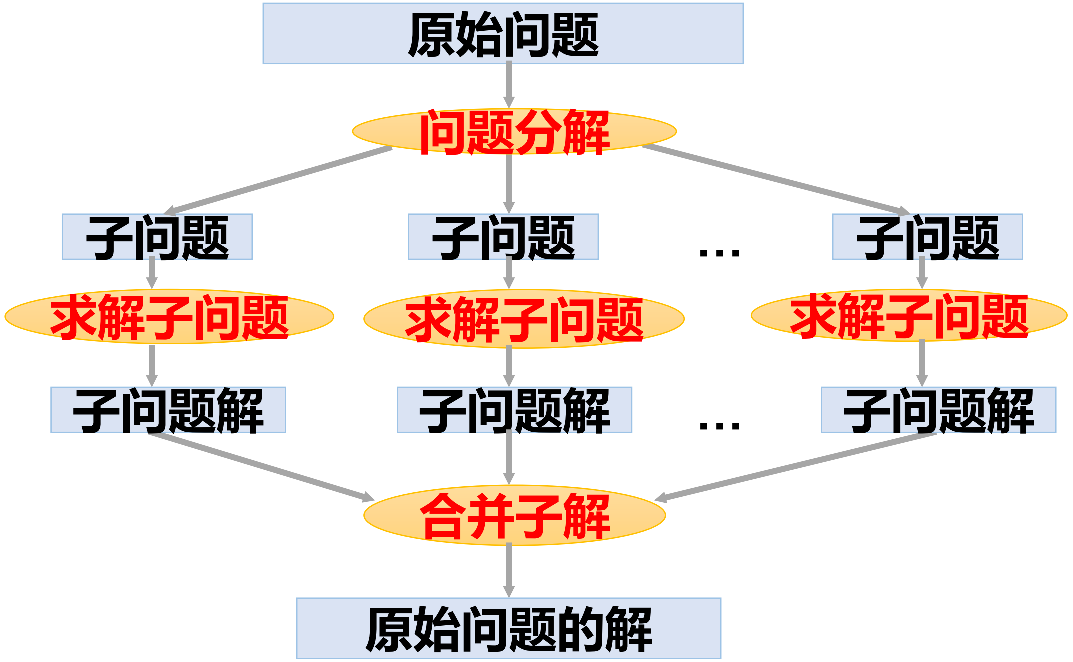
朱同鑫

本章内容

- 动态规划算法原理
- 矩阵链乘法问题
- 钢条切割问题
- 最长公共子序列问题
- 最优二叉搜索树问题
- 流水作业调度问题
- 0/1背包问题

动态规划算法原理

- 动态规划算法的研究动机 (Why?)
  - 分治算法存在的问题



问题：如果分解的子问题之间不是相互独立的，分治方法将重复计算公共子问题，效率很低

动态规划算法原理

- 动态规划算法的研究动机 (Why?)
  - 分治算法存在的问题
    - 例子：计算矩阵链乘法 $A_1 \times A_2 \times A_3 \times A_4$ 所需的最小乘法次数

$$A_1 \times A_2 \times A_3 \times A_4$$

P_1: $(A_1) \times (A_2 \times A_3 \times A_4)$

P_2: $(A_1 \times A_2) \times (A_3 \times A_4)$

P_3: $(A_1 \times A_2 \times A_3) \times (A_4)$

P_{11}: $(A_2 \times A_3)$

P_{12}: $(A_3 \times A_4)$

P_{21}: $(A_1 \times A_2)$

P_{22}: $(A_3 \times A_4)$

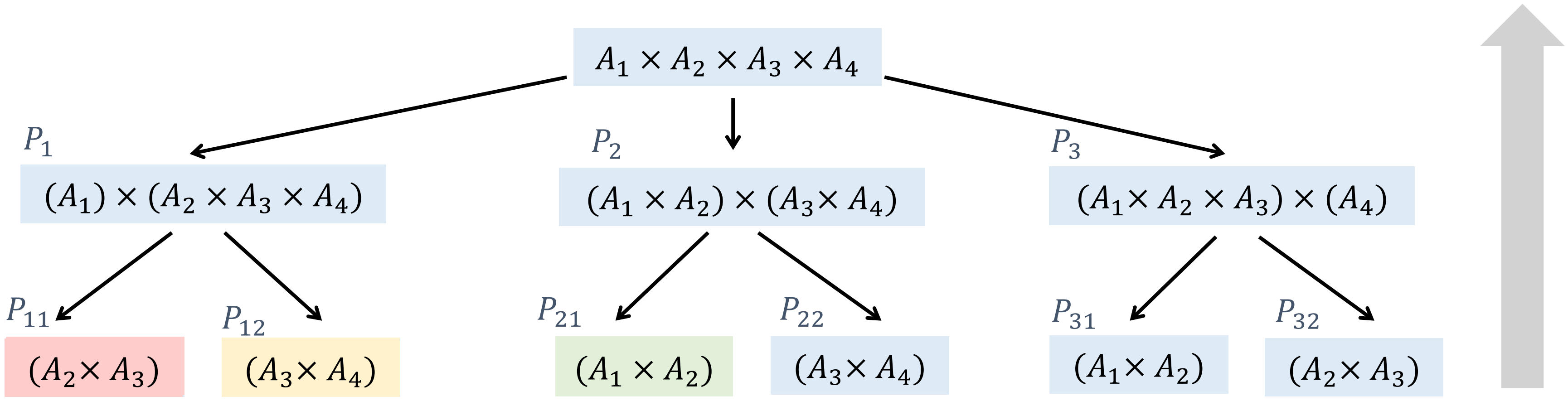
P_{31}: $(A_1 \times A_2)$

P_{32}: $(A_2 \times A_3)$

可以考虑利用空间换时间——动态规划！

动态规划算法原理

- 动态规划算法的思想和适用范围 (What?)



- 动态规划算法的思想
  - 把原始问题划分成一系列子问题
  - 自底向上地计算子问题
  - 每个重叠的子问题仅求解一次，并将其结果保存在一个表中，以后再次求解该子问题，直接从表中取得结果，不重复计算，节省计算时间

动态规划算法原理

- 动态规划算法的思想和适用范围 (What?)
  - 动态规划算法的适用范围
    - 优化问题：给定一个代价函数，在解空间中搜索具有最小或最大代价的解
    - 优化子结构 (Optimal substructure)： 当一个问题的最优解包含了子问题的最优解时，我们说这个问题具有优化子结构。
    - 重叠子问题 (Subteties)：问题的求解过程中，很多子问题的解将被多次使用

矩阵链乘法问题

- 问题定义
  - 输入：$<A_1, A_2, \ldots, A_n>$, A_i是$p_{i-1} \times p_i$矩阵
  - 输出：计算$A_1 \times A_2 \times \cdots \times A_n$的最小代价方法

矩阵乘法的代价/复杂性：计算乘法的次数

若A是$p \times q$矩阵，B是$q \times r$矩阵，则$A \times B$的代价是$O(pqr)$

矩阵链乘法问题

- 矩阵链乘法的实现
  - 矩阵乘法满足结合率
  - 计算一个矩阵链的乘法可有多种方法:

例如， $(A_1 \times A_2 \times A_3 \times A_4)$

$$= \left(A_1 \times \left(A_2 \times (A_3 \times A_4)\right)\right)$$

$$= \left((A_1 \times A_2) \times (A_3 \times A_4)\right)$$

$$\dots$$

$$= \left(\left((A_1 \times A_2) \times A_3\right) \times A_4\right)$$

矩阵链乘法问题

- 矩阵链乘法的代价与计算顺序的关系
  - 设A_1是一个10×100的矩阵，A_2是一个100×5的矩阵，A_3是一个5×50的矩阵
    - 方法一：$((A_1 \times A_2) \times A_3)$

    计算代价：$10 \times 100 \times 5 + 10 \times 5 \times 50 = 7500$
    - 方法二：$(A_1 \times (A_2 \times A_3))$

    计算代价：$100 \times 5 \times 50 + 10 \times 100 \times 50 = 75000$

结论：不同计算顺序会产生不同的计算代价，因此需要为矩阵链乘法找到最优的计算顺序使得其代价最小。

矩阵链乘法问题

- 矩阵链乘法优化问题的解空间
  - 设 $p(n)$ = 计算n个矩阵乘积的方法数
  - $p(n)$的递归方程

$$(A_1 \times \cdots \times A_k) \times (A_{k+1} \times \cdots \times A_n)$$

$$\begin{cases} p(n) = 1, & n = 1 \\ p(n) = \sum_{k=1}^{n-1} p(k) \times p(n-k), & n > 1 \end{cases}$$

$p(n) = C(n-1) =$Catalan数$= \frac{1}{n}\binom{2(n-1)}{n-1} = \Omega\left(\frac{4^n}{n^{3/2}}\right)$

⚠ 如此大的解空间是无法用枚举方法求出最优解的！

矩阵链乘法问题

- 动态规划算法步骤（How）
  1. 分析矩阵链乘法问题的最优解的结构
  2. 递归地定义最优解的计算代价
  3. 递归地划分问题，直至不可划分
  4. 自底向上求解各个子问题
     - 计算最优解代价并保存
     - 获取构造最优解的信息
  5. 根据构造最优解的信息构造矩阵链乘法的最优计算顺序

矩阵链乘法问题

1. 分析矩阵链乘法问题的最优解的结构

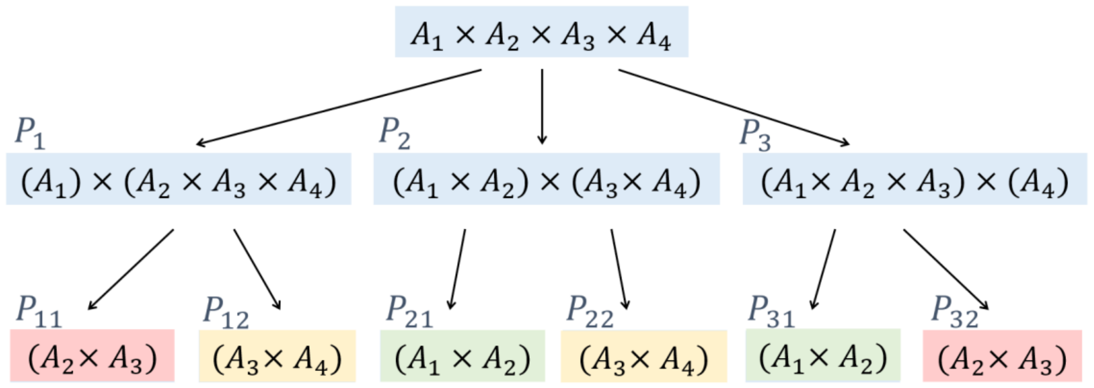
一个记号：$A_{i\sim j} = A_i \times \cdots \times A_j$

- 优化子结构

定理.

若计算 $A_{1\sim n}$ 的最优解在矩阵 A_k 和 A_{k+1} 之间加括号，即 $A_{1\sim n} = A_{1\sim k} \times A_{k+1\sim n}$，则在 $A_{1\sim n}$ 的最优解中，对应于子问题 $A_{1\sim k}$ 的解必为 $A_{1\sim k}$ 的最优解，对应于子问题 $A_{k+1\sim n}$ 的解必为 $A_{k+1\sim n}$ 的最优解。

- 子问题重叠性



矩阵链乘法问题

2. 递归地定义最优解的计算代价

- 矩阵链乘法的代价记号
  - $m[i,j] =$ 计算 $A_{i\sim j}$的最小乘法次数
- 假设 $A_{i\sim j}$的最优解在矩阵 A_k和 A_{k+1}之间加括号，即$(A_i \times \cdots \times A_k) \times (A_{k+1} \times \cdots \times A_j)$，代价方程为
  - $\boldsymbol{m[i,j] = m[i,k] + m[k+1,j] + p_{i-1}p_kp_j}$

子问题：$A_{i\sim k}$的最小乘法次数

子问题：$A_{k+1\sim j}$的最小乘法次数

$A_{i\sim k}$的结果× $A_{k+1\sim j}$的结果所需的乘法次数

矩阵链乘法问题

2. 递归地定义最优解的计算代价

- $(A_i \times \cdots \times A_k) \times \left(A_{k+1} \times \cdots \times A_j\right)$

考虑k的所有$(j-i)$个取值，矩阵链乘法 $A_{i\sim j}$的代价方程为：

- $m[i,j] = 0, \quad \text{if} \;\; i = j$
- $m[i,j] = \min\limits_{i \le k < j} \{m[i,k] + m[k+1,j] + p_{i-1}p_kp_j\}, \quad \text{if } i < j$

矩阵链乘法问题

3. 递归地划分问题，直至不可划分

- $m[i,j]=\min_{i\le k<j}\{m[i,k]+m[k+1,j]+p_{i-1}p_kp_j\}$

$m[i,i]$	$m[i,i+1]$	……	$m[i,j-1]$	$m[i,j]$
$\boldsymbol{k=i}$	$\boldsymbol{k=i+1}$		$\boldsymbol{k=j-1}$	$m[i+1,j]$
				$m[i+2,j]$
				……
				$m[j,j]$

矩阵链乘法问题

3. 递归地划分问题，直至不可划分

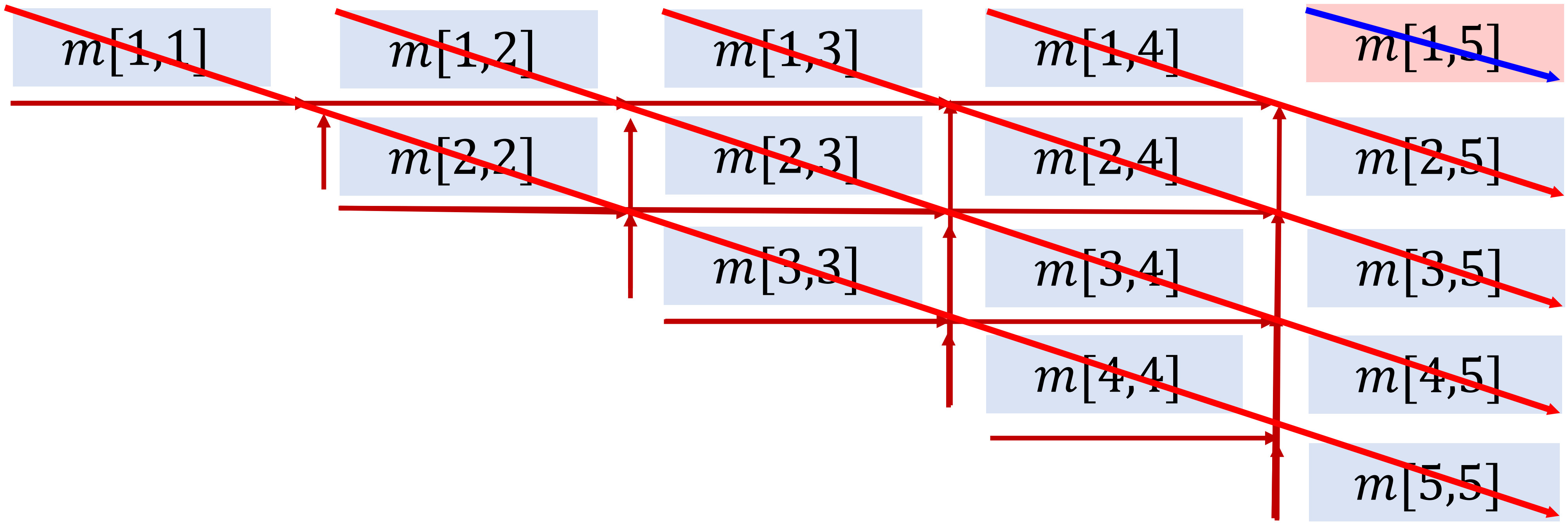
- $m[i,j]=\min\limits_{i\le k<j}\{m[i,k]+m[k+1,j]+p_{i-1}p_kp_j\}$

$m[1,1]$	$m[1,2]$	$m[1,3]$	$m[1,4]$	$m[1,5]$
	$m[2,2]$	$m[2,3]$	$m[2,4]$	$m[2,5]$
		$m[3,3]$	$m[3,4]$	$m[3,5]$
			$m[4,4]$	$m[4,5]$
				$m[5,5]$

矩阵链乘法问题

4. 自底向上求解各个子问题——计算最优解代价并保存

- $m[i,j] = \min_{i \le k < j} \{ m[i,k] + m[k+1,j] + p_{i-1} p_k p_j \}$

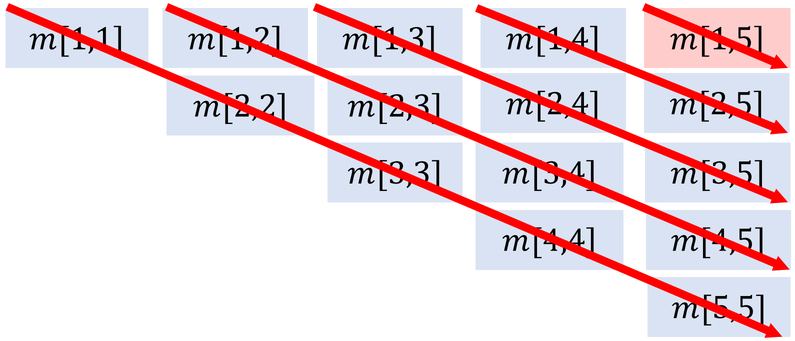


矩阵链乘法问题

4. 自底向上求解各个子问题——计算最优解代价并保存

Matrix-Chain-Order (n)

for $i = 1$ **_to_** n **_do_**
    $m[i,i] = 0;$
for $l = 2$ **_to_** n **_do_**
/* 计算 l 对角线 */
    for $i = 1$ **_to_** $n - l + 1$ **_do_**
        $j = i + l - 1;$
        $m[i,j] = \infty;$
        for $k = i$ **_to_** $j - 1$ **_do_**
         /* 计算$m[i,j]$ */
            $q = m[i,k] + m[k+1,j] + p_{i-1}p_kp_j;$
            _if_ $q < m[i,j]$ **_then_** $m[i,j] = q;$
return m



时间复杂性： $O(n^3)$

空间复杂度： $O(n^2)$

矩阵链乘法问题

4. 自底向上求解各个子问题——获取构造最优解的信息

- 算法伪代码

```
Matrix-Chain-Order (n)

for i = 1 to n do
    m[i,i] = 0;
for l = 2 to n do
/* 计算 l 对角线 */
    for i = 1 to n − l + 1 do
        j = i + l − 1;
        m[i,j] = ∞;
        for k = i to j − 1 do
         /* 计算m[i,j] */
            q = m[i,k] + m[k+1,j] + p_{i−1} p_k p_j;
            if q < m[i,j] then m[i,j] = q;   s[i,j] = k;
return m;
```

$$m[i,j] = \min_{i \le k < j}\{m[i,k] + m[k+1,j] + p_{i-1}p_kp_j\}$$

$s[i,j] = k;$

每次遇到使得 $A_{i\sim j}$的计算代价最小的k值时，使用$s[i,j] = k$记录下来

矩阵链乘法问题

5. 根据构造最优解的信息构造矩阵链乘法的最优计算顺序

- 算法伪代码

```
Print-Optimal-Parens (s, i, j)
if j = i then Print "A"_i ;
else
    Print "(";
    Print-Optimal-Parens (s, i, s[i, j]);
    Print-Optimal-Parens (s, s[i, j] + 1, j);
    Print ")";
```

- $s[i,j]$记录 $A_{i\sim j}$的最优划分处
- $s[i,s[i,j]]$记录 $A_{i\sim s[i,j]}$的最优划分处
- $s[s[i,j]+1,j]$记录 $A_{s[i,j]+1\sim j}$的最优划分处

调用**Print-Optimal-Parens** $(\boldsymbol{s}, \mathbf{1}, \boldsymbol{n})$，即可输出 $A_{1\sim n}$的优化计算顺序

钢条切割问题

- 问题定义
  - 输入：长度为n英寸的钢条，和一个价格表$p_i(i = 1,2,\cdots n)$，其中p_i表示长度为i英寸钢条的价格
  - 输出：使得收益最大的钢条切割方案
- 问题实例
  - 长度为4英寸的钢条进行切割
  - 价格表:

长度i	1	2	3	4
价格p_i	1	5	8	9

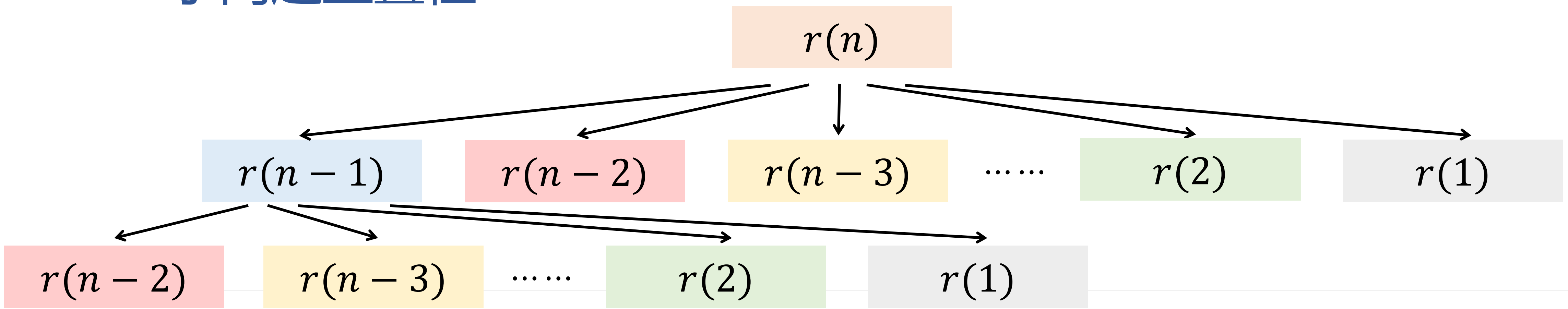
  - 分割为两段2英寸钢条，收益最大

钢条切割问题

- 优化子结构
  - 设长度为n的钢条切割的最大收益为$r(n)$，有

$$r(n) = \max_{1 \le i \le n}\{p_i + r(n-i)\}$$

  - 具有优化子结构：问题的最优解包含子问题的最优解
- 子问题重叠性



钢条切割问题

- 算法伪代码

自底向上方法

<u>***Bottom-Up-Cut-Rod*** (p, n)</u>

Let $r[0 \cdots n]$ *be a new array;*
$r[0] = 0;$
/*依次计算$r[1], r[2], \cdots, r[n]$*/
for $j = 1$ ***to*** n ***do***
  $q = -\infty;$
  for $i = 1$ ***to*** j
    $q = \max(q, p[i] + r[j - i]);$
  $r[j] = q;$
return $temp;$

递归备忘录方法

预处理，$r[i] = -\infty$

<u>***Memorized-Cut-Rod*** (p, n, r)</u>
if $r[n] \geq 0$ ***then***
  return $r[n];$ /*$r[n]$不用重复计算了*/
if $n == 0$ ***then***
  $temp = 0;$
else
  $temp = -\infty;$
  for $i = 1$ ***to*** n
    $temp = \max(temp, p[i] +$
      Memorized–Cut–Rod$(p, n - i, r)$);
  $r[n] = temp;$
return $temp;$

止于至善

最长公共子序列问题

- 子序列
  - $X = (A, B, C, D, E, F, G)$
  - $Z = (B, C, E, F)$是 X 的子序例
  - $W = (B, D, A)$不是X的子序例
- 公共子序列
  - Z是X的子序列，Z也是Y的子序列，那么Z是序列X与Y的公共子序列
    - $X = (A, B, C, D, E)$，$\mathrm{Y} = (A, C, E, B, F, G)$
    - $Z = (A, C, E)$是X与Y的公共子序列， $Z = (A, B)$也是X与Y的公共子序列

最长公共子序列问题

- 最长公共子序列（LCS）问题定义
  - 输入：$X = (x_1, x_2, \cdots, x_m)$，$Y = (y_1, y_2, \cdots, y_n)$
  - 输出：Z = X与Y的最长公共子序列

- 第i前缀
  - 设$X = (x_1, x_2, \cdots, x_n)$是一个序列，则$X_i = (x_1, \cdots, x_i)$是$X$的第$i$前缀
  - 例：$X = (A, B, D, C, A)$，$X_1 = (A)$，$X_2 = (A, B)$，$X_3 = (A, B, D)$

最长公共子序列问题

- 优化子结构
  - 设$X = (x_1, x_2, \cdots, x_m)$和$Y = (y_1, y_2, \cdots, y_n)$的最长公共子序列为$LCS_{XY} = (z_1, \cdots, z_k)$

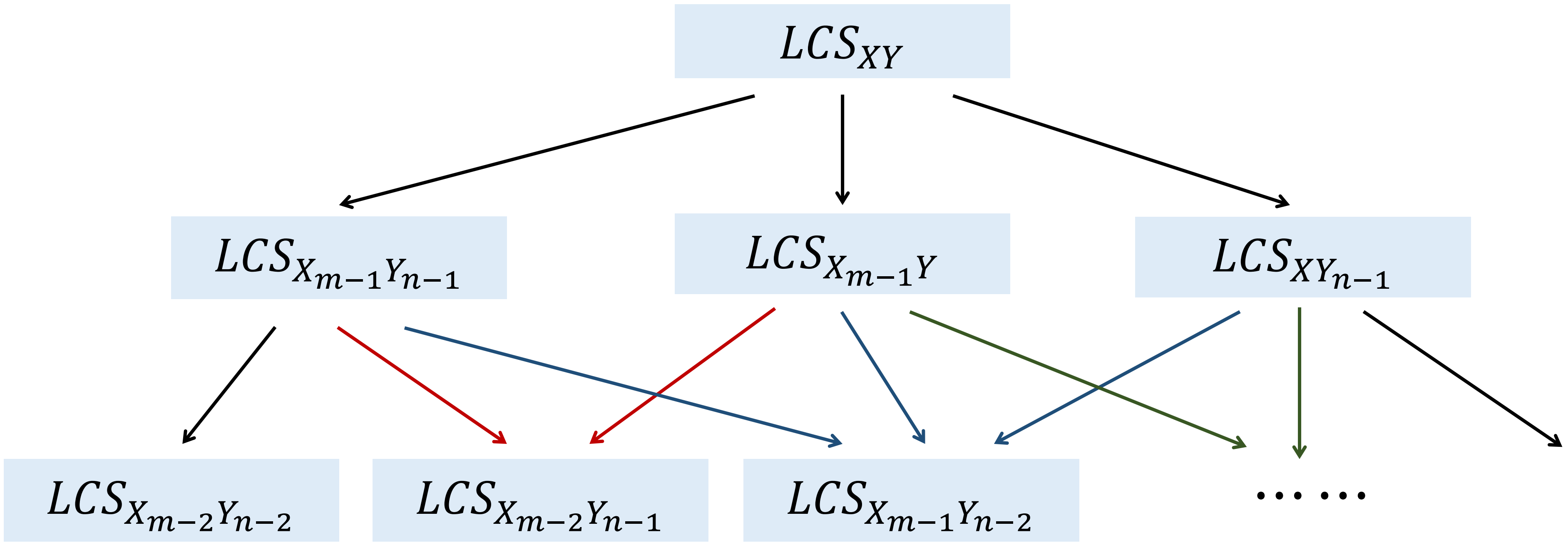
具有优化子结构：问题的最优解包含子问题的最优解

- $\boldsymbol{x_m = y_n}$
  - $\boldsymbol{LCS_{XY} = LCS_{X_{m-1}Y_{n-1}} + < x_m = y_n >}$
- $\boldsymbol{x_m \neq y_n}$
  - $\boldsymbol{z_k \neq x_m}$
    - $LCS_{XY} = LCS_{X_{m-1}Y}$
  - $\boldsymbol{z_k \neq y_n}$
    - $LCS_{XY} = LCS_{XY_{n-1}}$

$$\boldsymbol{LCS_{XY} = \max\{LCS_{X_{m-1}Y}, LCS_{XY_{n-1}}\}}$$

最长公共子序列问题

- 子问题重叠性



最长公共子序列问题

- 递归方程
  - 设$C[i,j]$表示X_i和Y_j的LCS长度
    - $C[i,j] = 0$ $\quad if \;\; i = 0 \;\; or \;\; j = 0$
    - $C[i,j] = C[i-1,j-1] + 1$ $\quad if \;\; i,j > 0 \; and \; x_i = y_j$
    - $C[i,j] = \max\{C[i,j-1], C[i-1,j]\}$ $\quad if \;\; i,j > 0 \; and \; x_i \neq y_j$

	$\boldsymbol{C[i-1,j-1]}$	$\boldsymbol{C[i-1,j]}$	
	$\boldsymbol{C[i,j-1]}$	$\boldsymbol{C[i,j]}$	

最长公共子序列问题

- 递归划分子问题与自底向上求解过程

$C[0,0]$	$C[0,1]$	$C[0,2]$	$C[0,3]$	$C[0,4]$	$C[0,5]$
$C[1,0]$	$C[1,1]$	$C[1,2]$	$C[1,3]$	$C[1,4]$	$C[1,5]$
$C[2,0]$	$C[2,1]$	$C[2,2]$	$C[2,3]$	$C[2,4]$	$C[2,5]$
$C[3,0]$	$C[3,1]$	$C[3,2]$	$C[3,3]$	$C[3,4]$	$C[3,5]$

- $C[i,j]$记录X_i和Y_j的LCS长度，$B[i,j]$记录最优解的信息

最长公共子序列问题

$$
\begin{array}{ll}
C[i,j] = 0 & if\ \ i = 0\ \ or\ \ j = 0 \\
C[i,j] = C[i-1,j-1] + 1 & if\ \ i,j > 0\ and\ x_i = y_j \\
C[i,j] = \max\{C[i,j-1], C[i-1,j]\} & if\ \ i,j > 0\ and\ x_i \neq y_j
\end{array}
$$

- 记录最优解信息

x_i \ y_j		B	D	C	A	B	A
	0	0	0	0	0	0	0
A	0	↑0	↑0	↑0	↖1	←1	↖1
B	0	↖1	←1	←1	↑1	↖2	←2
C	0	↑1	↑1	↖2	←2	↑2	↑2
B	0	↖1	↑1	↑2	↑2	↖3	←3
D	0	↑1	↖2	↑2	↑2	↑3	↑3
A	0	↑1	↑2	↑2	↖3	↑3	↖4
B	0	↖1	↑2	↑2	↑3	↖4	↑4

- 若$C[i,j] = C[i-1,j-1] + 1$，$B[i,j] ='↖'$;
- 若$C[i,j] = C[i,j-1]$，$B[i,j] ='←'$;
- 若$C[i,j] = C[i-1,j]$，$B[i,j] ='↑'$

止于至善

最长公共子序列问题

$$
\begin{aligned}
&C[i,j] = 0 && if\ \ i = 0\ \ or\ \ j = 0 \\
&C[i,j] = C[i-1,j-1] + 1 && if\ \ i,j > 0\ and\ x_i = y_j \\
&C[i,j] = \max\{C[i,j-1], C[i-1,j]\} && if\ \ i,j > 0\ and\ x_i \neq y_j
\end{aligned}
$$

- 算法伪代码

LCS-Length (X, Y)

for $i = 0$ ***to*** m ***do***
  $C[i,0] = 0;$
for $j = 1$ ***to*** n ***do***
  $C[0,j] = 0;$
for $i = 1$ ***to*** m ***do***
  for $j = 1$ ***to*** n ***do***
    if $x_i = y_j$ ***then***
      $C[i,j] = C[i-1,j-1] + 1;\quad B[i,j] = {}'\nwarrow';$
    else if $C[i-1,j] \geq C[i,j-1]$ ***then***
      $C[i,j] = C[i-1,j];\quad B[i,j] = {}'\uparrow';$
    else
      $C[i,j] = C[i,j-1];\quad B[i,j] = {}'\leftarrow';$
return B and C;

Print-LCS (B, X, i, j)

if $i = 0$ or $j = 0$ ***then***
  return;
if $B[i,j] = {}'\nwarrow'$ ***then***
  Print-LCS$(B, X, i-1, j-1)$; Print x_i;
else if $B[i,j] = {}'\uparrow'$ ***then***
  Print-LCS$(B, X, i-1, j)$;
else
  Print-LCS$(B, X, i, j-1)$;

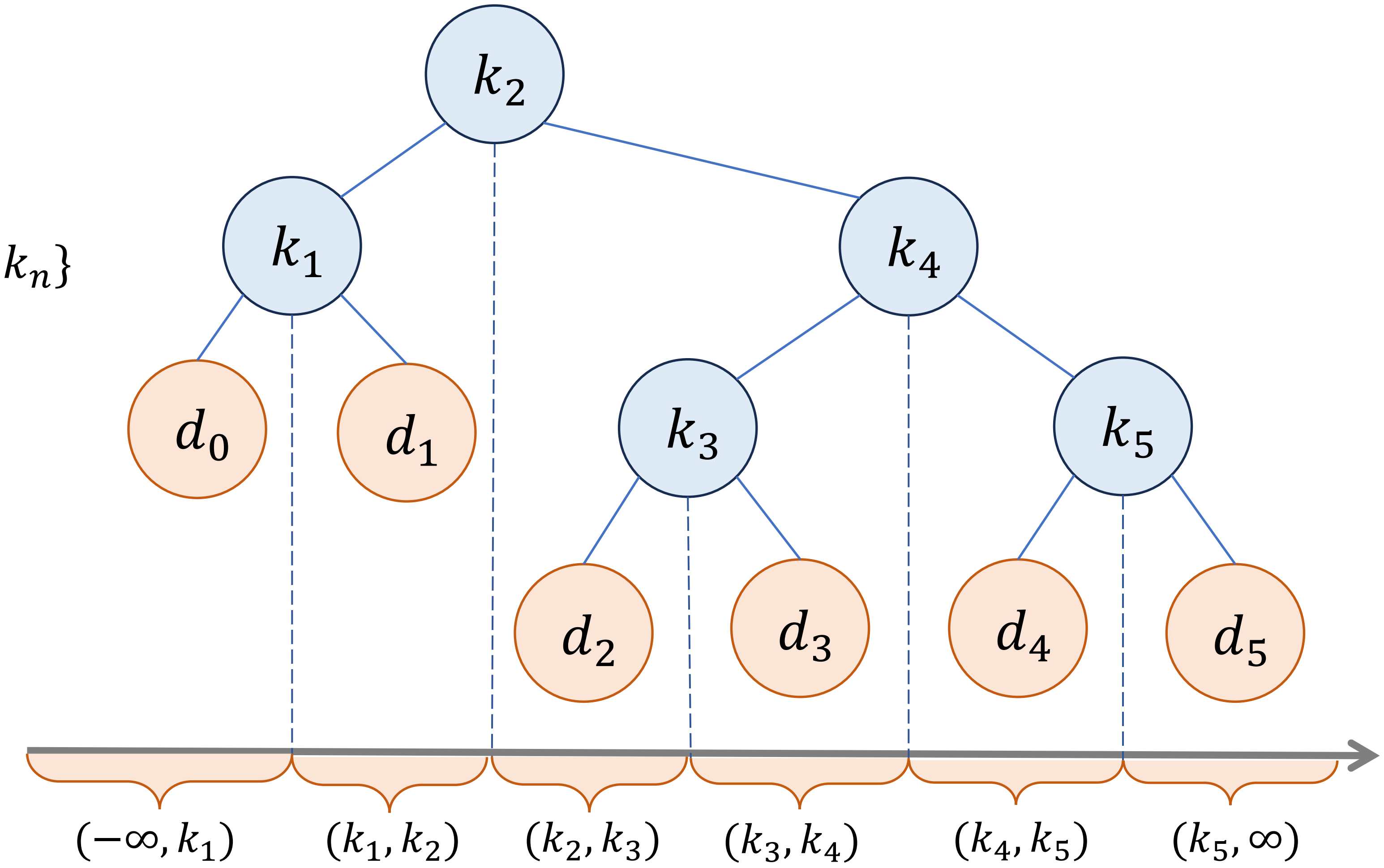
时间复杂度$O(mn)$

空间复杂度$O(mn)$

止于至善

最优二叉搜索树问题

- 二叉搜索树
  - 节点
    - 有序关键字$K = \{k_1, \cdots, k_n\}$
    - 区间$D = \{d_0, d_1, \cdots, d_n\}$
  - 附加信息
    - 搜索k_i的概率为p_i
    - 搜索d_j的概率为q_j
    - $\sum_{i=1}^{n} p_i + \sum_{j=0}^{n} q_j = 1$



搜索树的期望代价：$E(T) = \sum_{i=1}^{n}(D_T(k_i) + 1)\, p_i + \sum_{j=0}^{n}\left(D_T\left(d_j\right) + 1\right)q_j$

最优二叉搜索树问题

- 问题定义
  - 输入：$K = \{k_1, \cdots, k_n\}$， $k_1 < \cdots < k_n$，
    $P = \{p_1, \cdots, p_n\}$，其中p_i为搜索k_i的概率,
    $Q = \{q_0, q_1, \cdots, q_n\}$，其中$q_j$为搜索$d_j$（即$(k_j, k_{j+1})$）的概率
  - 输出：构造K的二叉搜索树T，最小化期望代价

$$E(T) = \sum_{i=1}^{n} (D_T(k_i) + 1)\, p_i + \sum_{j=0}^{n} \left(D_T\left(d_j\right) + 1\right) q_j$$

最优二叉搜索树问题

- 优化子结构

定理.

若优化二叉搜索树T具有包含关键字集合$\{k_i, \cdots, k_j\}$的子树T'，则T'是关于关键字集合$\{k_i, \cdots, k_j\}$的子问题的最优解。

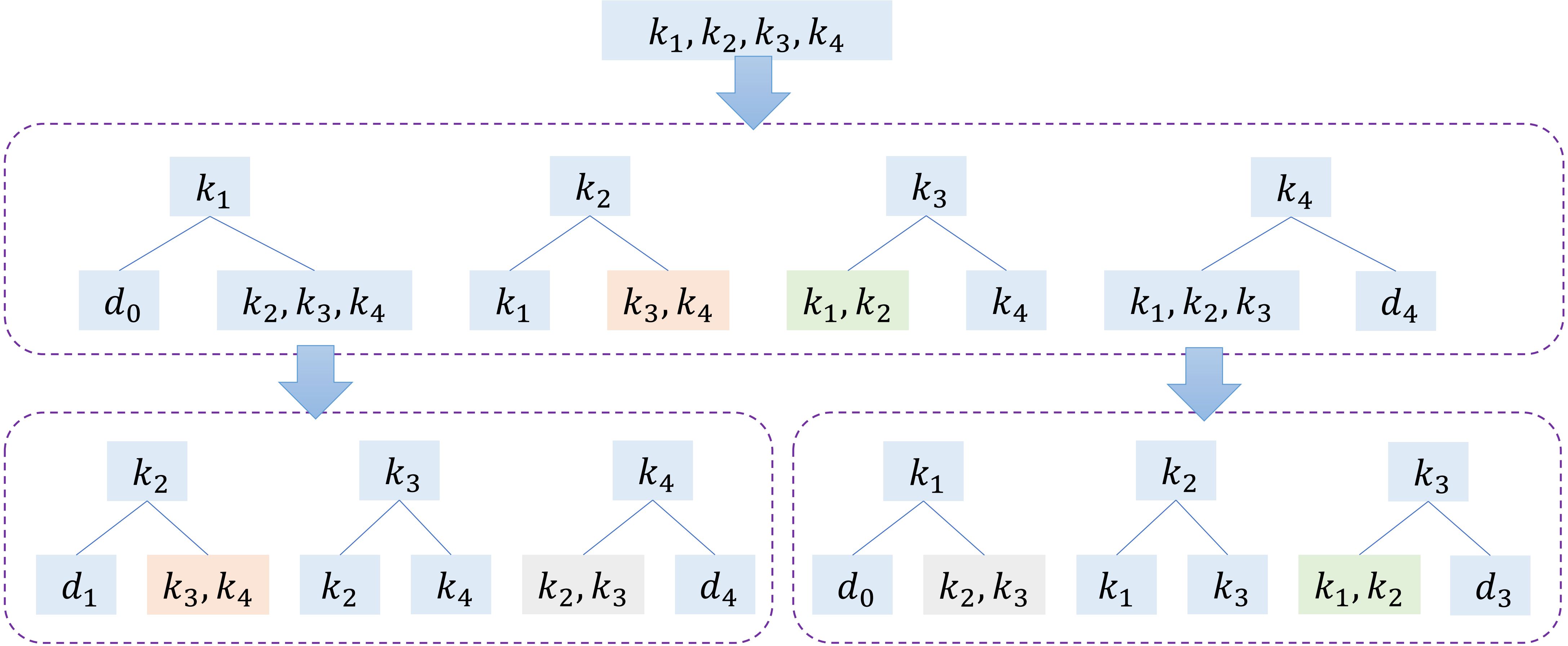
证明.

反证法。若T'不是关于关键字集合$\{k_i, \cdots, k_j\}$的子问题的最优解，而G'是关于关键字集合$\{k_i, \cdots, k_j\}$的子问题的最优解，用子树G'替换T'，会得到期望代价更小的优化二叉搜索树G，与T是优化二叉搜索树矛盾，得证。

具有优化子结构：问题的最优解包含子问题的最优解

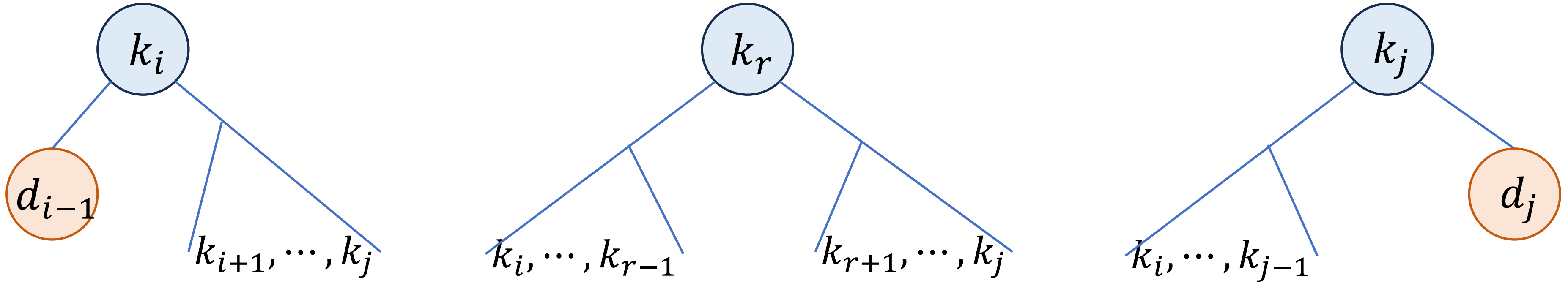
最优二叉搜索树问题

- 子问题重叠性



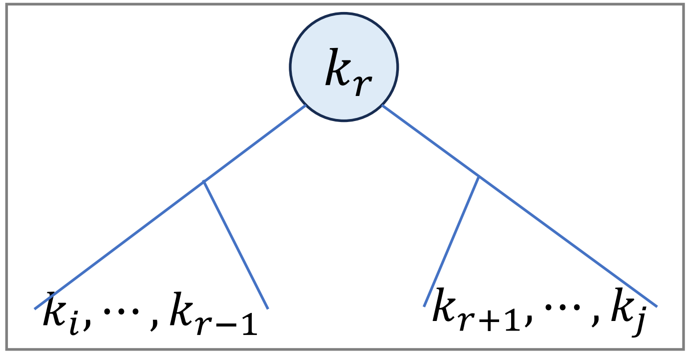
最优二叉搜索树问题

- 递归方程
  - $E(i,j)$表示$\{k_i,\cdots,k_j\}$的最优解T_{ij}的期望搜索代价
    1. 当$j=i-1$时，T_{ij}只有叶节点d_{i-1}，$E(i,i-1)=q_{i-1}$
    2. 当$j\geq i$时，选择一个$k_r\in\{k_i,\cdots,k_j\}$



$$E(i,j)=\underbrace{p_r}_{\text{根的代价}}+\underbrace{E(i,r-1)}_{\text{左子树的代价}}+\underbrace{\boxed{W(i,r-1)}}_{\text{左子树增加一层的代价}}+\underbrace{E(r+1,j)}_{\text{右子树的代价}}+\underbrace{\boxed{W(r+1,j)}}_{\text{右子树增加一层的代价}}$$

最优二叉搜索树问题



- 递归方程
  - $E(i,j) = \underbrace{p_r}_{\text{根的代价}} + \underbrace{E(i,r-1)}_{\text{左子树的代价}} + \boxed{W(i,r-1)} + \underbrace{E(r+1,j)}_{\text{右子树的代价}} + \boxed{W(r+1,j)}$
  - $W(i,r-1) = \sum_{l=i}^{r-1} p_l + \sum_{l=i-1}^{r-1} q_l$，$W(r+1,j) = \sum_{l=r+1}^{j} p_l + \sum_{l=r}^{j} q_l$
  - $W(i, r-1)$为区间范围(k_{i-1}, k_r)的概率乘以高度1
  - 即$k_i, \cdots k_{r-1}$的概率和$\sum_{l=i}^{r-1} p_l$ 与 $d_{i-1}, \cdots, d_{r-1}$的概率和$\sum_{l=i-1}^{r-1} q_l$
  - 设$W(i,j) = p_r + W(i,r-1) + W(r+1,j) = \sum_{l=i}^{j} p_l + \sum_{l=i-1}^{j} q_l$
  - $E(i,j) = E(i,r-1) + E(r+1,j) + W(i,j)$

$$E(T) = \sum_{i=1}^{n} (D_T(k_i) + 1)\, p_i + \sum_{j=0}^{n} (D_T(d_j) + 1) q_j$$

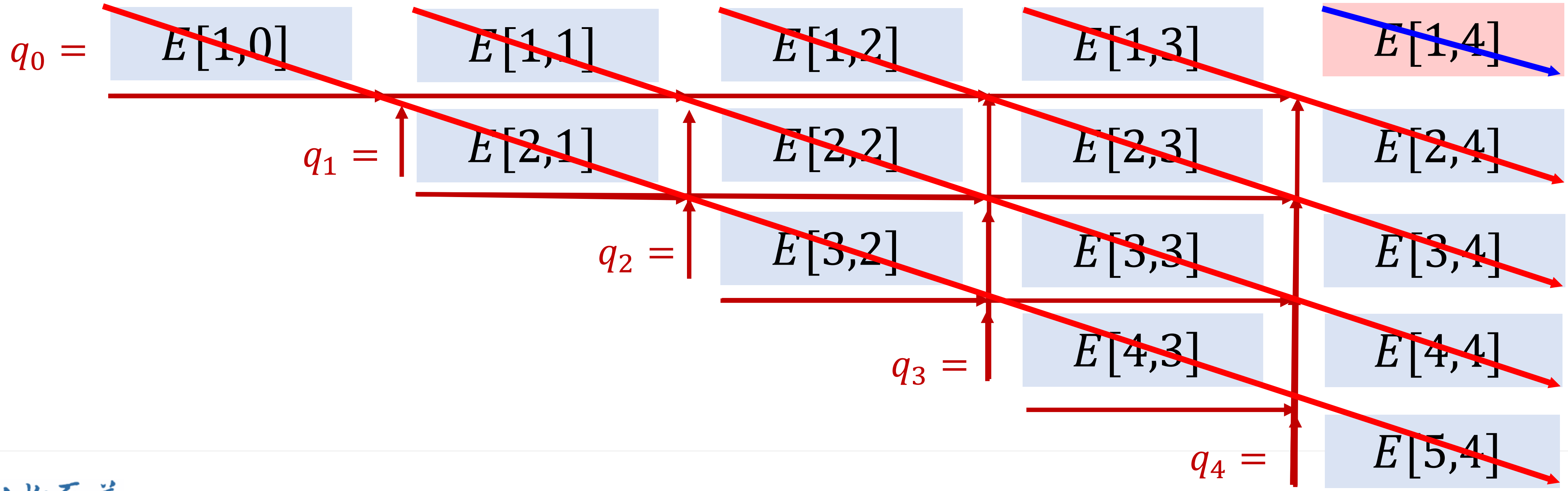
最优二叉搜索树问题

- 递归方程
  - $E(i, i-1) = q_{i-1}$ $\quad if \;\; j = i-1$
  - $E(i,j) = \min\limits_{i \le r \le j}\{E(i, r-1) + E(r+1, j) + W(i,j)\}$ $\quad if \;\; j \ge i$

$E[i, i-1]$	$E[i, i]$	……	$E[i, j-1]$	$E[i, j]$
$\boldsymbol{r = i}$	$\boldsymbol{r = i+1}$		$\boldsymbol{r = j}$	$E[i+1, j]$
				$E[i+2, j]$
				……
				$E[j+1, j]$

最优二叉搜索树问题

- 递归划分子问题与求解过程
  - $E(i, i-1) = q_{i-1}$ $\qquad R(i,j) = r$ $\qquad$ *if* $j = i - 1$
  - $E(i,j) = \min\limits_{i \le r \le j} \{E(i, r-1) + E(r+1, j) + W(i,j)\}$ $\qquad$ *if* $j \ge i$



最优二叉搜索树问题

- 算法伪代码

<u>***Optimal-BST***</u> (P, Q, n)

for $i = 1$ ***to*** $n+1$ ***do***

  $E(i, i-1) = q_{i-1};\ \ W(i, i-1) = q_{i-1};$

for $l = 1$ ***to*** n ***do***

  for $i = 1$ ***to*** $n - l + 1$ ***do***

    $j = i + l - 1;$

    $E(i, j) = \infty;$

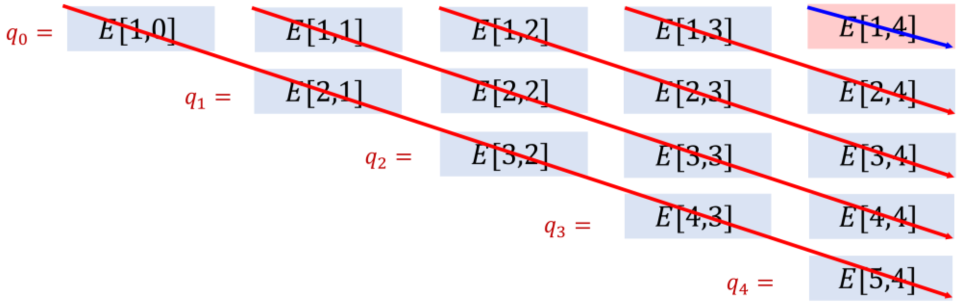
    $W(i, j) = W(i, j-1) + p_j + q_j;$

    for $r = i$ ***to*** j ***do***

      $t = E(i, r-1) + E(r+1, j) + W(i, j);$

      if $t < E(i, j)$ ***then*** $E(i, j) = t;\ \ R(i, j) = r;$

return E and R;



时间复杂度$O(n^3)$

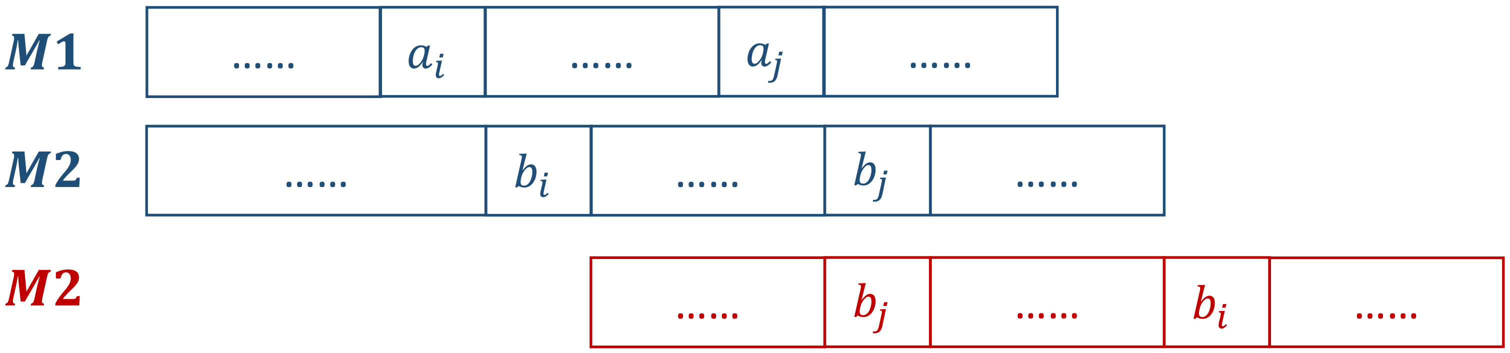
空间复杂度$O(n^2)$

流水作业调度问题

- 流水线作业
  - 每个作业在2台机器$M1$和$M2$组成的流水线上完成加工，每个作业先在$M1$上加工，然后在$M2$上加工，每台机器同一时间最多只能执行一个作业
- 问题定义：
  - 输入：n个作业$N = \{1, \cdots, n\}$，作业i在$M1$和$M2$加工的时间分别为a_i, b_i
  - 输出：n个作业在两台机器的执行顺序，使得所有作业在两台机器上都加工完成所需时间最少

流水作业调度问题

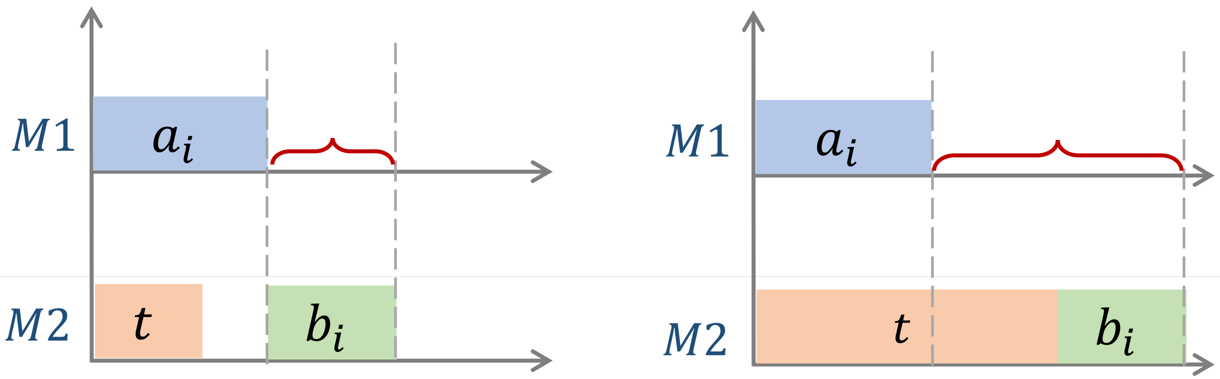
- 问题分析：
  - 一定存在最优调度使$M1$上的加工是无间断的
    - $M1$上的总加工时间是所有a_i之和
    - $M2$上的总加工时间不一定是b_i之和
  - 一定存在最优调度使作业在两台机器上的加工次序是完全相同的



若是b_i和b_j交换，则b_i需要a_j结束，等待时间更长

流水作业调度问题

- 优化子结构
  - $N = \{1, \cdots, n\}$，$S \subseteq N$
    - $M1$开始加工S中的作业时， $M2$还在加工其他作业，等待时间为t
    - 完成S中的作业的加工，所需最短时间记为$T(S, t)$
    - 完成N中的作业的加工，所需最短时间记为$T(N, 0)$
  - $T(N, 0) = \min_{i \in N}\{a_i + T(N - \{i\}, b_i)\}$
  - $T(S, t) = \min_{i \in S}\{a_i + T(S - \{i\}, b_i + \max\{t - a_i, 0\})\}$



流水作业调度问题

- 优化子结构

定理.

设Π是关于作业集合N的最优调度，其加工顺序为$\pi_1, \cdots, \pi_n$，其加工时间为$T(N,0) = a_{\pi_1} + T'$，T'为关于作业集合$S = N - \{\pi_1\}$的子问题的最优调度，即$T' = T(S, b_{\pi_1})$。

证明.

反证法。若T'不是关于作业集合$S = N - \{\pi_1\}$的子问题的优化解，而G'是，用G'替换T'，会得到加工时间更短的调度顺序G，与T是最优调度矛盾，得证。

具有优化子结构：问题的最优解包含子问题的最优解

流水作业调度问题

$$T(N,0) = \min_{i \in N}\{a_i + T(N-\{i\}, b_i)\}$$
$$T(S,t) = \min_{i \in S}\{a_i + T(S-\{i\}, b_i + \max\{t-a_i, 0\})\}$$

- 子问题重叠性

$T(N,0)$

$T(N-\{1\}, b_1)$ $T(N-\{2\}, b_2)$ …… $T(N-\{n\}, b_n)$

$T(N-\{1,n\}, b_n + \max\{b_1 - a_n, 0\})$

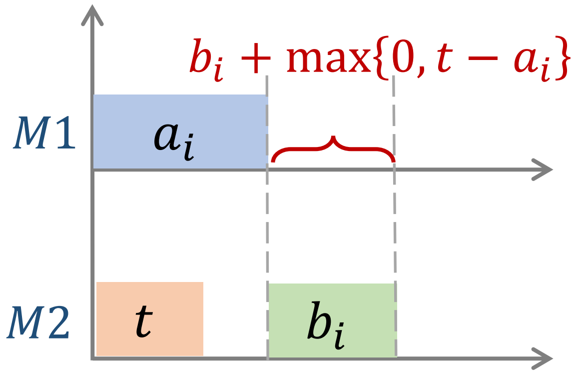
……

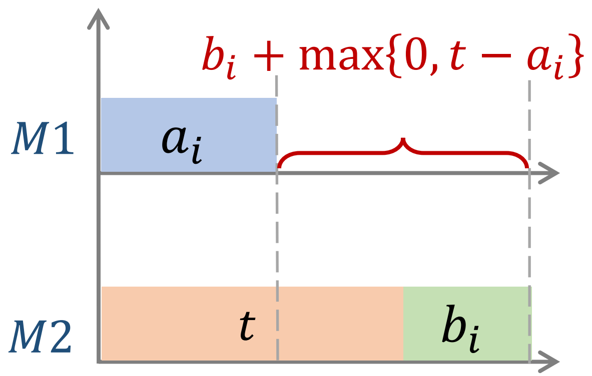
$T(N-\{1,3\}, b_3 + \max\{b_1 - a_3, 0\})$

$T(N-\{1,2\}, b_2 + \max\{b_1 - a_2, 0\})$ $T(N-\{1,2\}, b_1 + \max\{b_2 - a_1, 0\})$

问题 虽然，满足优化子结构，也一定程度满足子问题重叠性
但是，N的每个非空子集都需要计算一次，共2^N-1次，指数级

流水作业调度问题



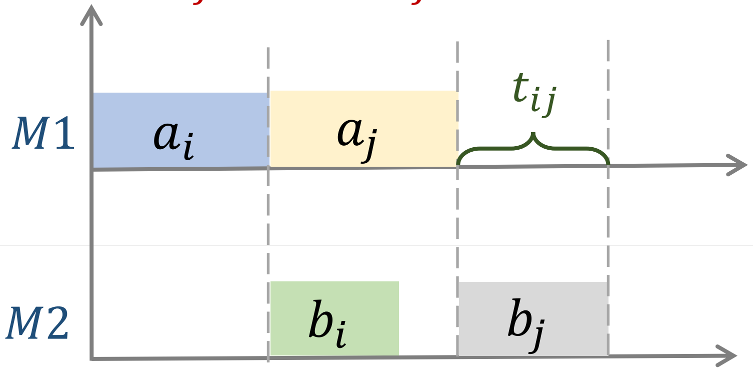


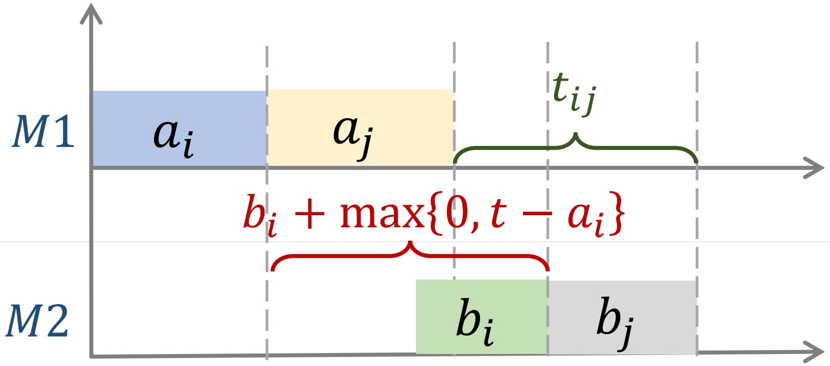
- Johnson不等式
  - 设一个最优调度中最先加工的两个作业是i和j
    - 如果作业i和j满足$\min\{b_i, a_j\} \geq \min\{b_j, a_i\}$，则称$i$和$j$满足Johnson不等式
    - $T(S,t) = a_i + T(S - \{i\}, b_i + \max\{t - a_i, 0\})$ $= a_i + a_j + T(S - \{i,j\}, t_{ij}\}$
    - $t_{ij} = b_j + \max\{0, b_i + \max\{0, t - a_i\} - a_j\}$

$= b_j + b_i - a_j + \max\{a_j - b_i, \max\{0, t - a_i\}\}$

$= b_j + b_i - a_j + \max\{a_j - b_i, 0, t - a_i\}$

$= b_j + b_i - a_j - a_i + \max\{a_i + a_j - b_i, a_i, t\}$





止於至善

流水作业调度问题

- Johnson不等式

如果i和j满足$t_{ij} \le t_{ji}$，可满足Johnson不等式，则作为最优调度顺序，不能调换

  - $T(S,t) = a_i + a_j + T(S-\{i,j\}, t_{ij}\}$
  - $t_{ij} = b_j + b_i - a_j - a_i + \max\{a_i + a_j - b_i,\ a_i,\ t\}$

调换i和j的顺序

  - $t_{ji} = b_j + b_i - a_j - a_i + \max\{a_i + a_j - b_j,\ a_j,\ t\}$
  - 如果i和j满足Johnson不等式： $\min\{b_i, a_j\} \ge \min\{b_j,\ a_i\}$
    - $\max\{-b_i, -a_j\} \le \max\{-b_j, -a_i\}$
    - $a_i + a_j + \max\{-b_i, -a_j\} \le a_i + a_j + \max\{-b_j, -a_i\}$
    - $\max\{a_i + a_j - b_i, a_i\} \le \max\{a_i + a_j - b_j, a_j\}$
    - $\max\{a_i + a_j - b_i,\ a_i,\ t\} \le \max\{a_i + a_j - b_j,\ a_j,\ t\}$

流水作业调度问题

- 计算过程
  - 如果i和j满足Johnson不等式： $\min\{b_i, a_j\} \geq \min\{b_j,\ a_i\}$
  - 则在最优调度中，作业i在作业j前执行
  - 推广到一般情况
    - 当$\min\{a_1, \cdots, a_n, b_1, \cdots, b_n\} = a_i$时，对$\forall k \neq i$，都有$\min\{b_i, a_k\} \geq \min\{b_k,\ a_i\}$，那么，在最优调度中，作业$i$排在最前边
    - 当$\min\{a_1, \cdots, a_n, b_1, \cdots, b_n\} = b_j$时，对$\forall k \neq j$，都有$\min\{b_k, a_j\} \geq \min\{b_j,\ a_k\}$，那么，在最优调度中，作业$j$排在最后边

流水作业调度问题

- 当$\min\{a_1,\cdots,a_n,b_1,\cdots,b_n\}=a_i$时，对$\forall k\neq i$，都有$\min\{b_i,a_k\}\geq\min\{b_k,\ a_i\}$，那么，在最优调度中，作业$i$排在最前边
- 当$\min\{a_1,\cdots,a_n,b_1,\cdots,b_n\}=b_j$时，对$\forall k\neq j$，都有$\min\{b_k,a_j\}\geq\min\{b_j,\ a_k\}$，那么，在最优调度中，作业$j$排在最后边

- 计算过程
  - 例：$(a_1,a_2,a_3,a_4)=(5,12,4,8)$

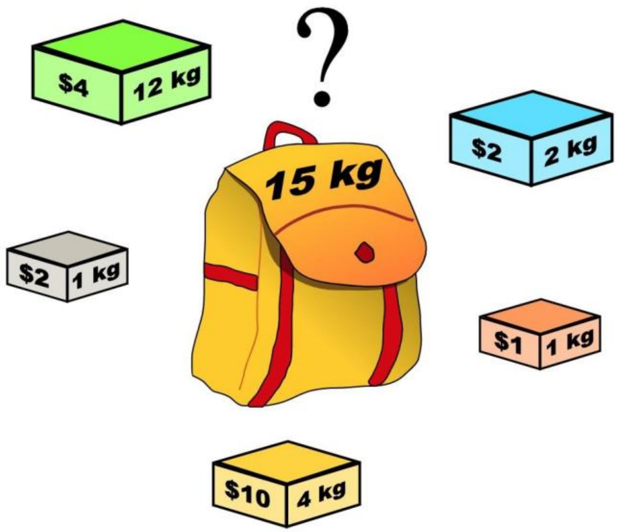
    $(b_1,b_2,b_3,b_4)=(6,2,14,7)$

作业	1	2	3	4
a	5	12	4	8
b	6	2	14	7

调度结果	3	1	4	2

0/1背包问题

- 问题定义
  - 给定n个物品和一个背包，物品i的重量是w_i，价值v_i，背包容量为C，问如何选择装入背包的物品，使装入背包中的物品的总价值最大?
    - 对于每种物品只能选择完全装入或不装入
    - 一个物品至多装入一次



0/1背包问题

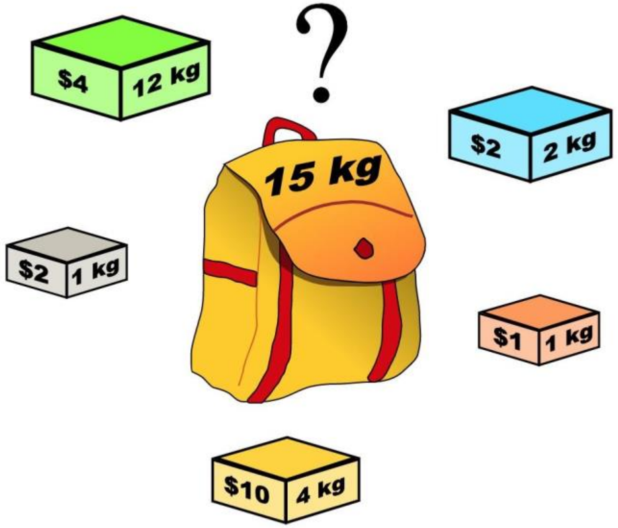
- 问题定义
  - 输入： $C > 0, w_i > 0, v_i > 0, 1 \le i \le n$
  - 输出：$(x_1, \cdots, x_n), x_i \in \{0,1\}$，满足$\sum_{i=1}^{n} w_i x_i \le C$，

    $\sum_{i=1}^{n} v_i x_i$最大
  - 整数规划问题

$$\max \quad \sum_{i=1}^{n} v_i x_i$$

$$s.t. \quad \sum_{i=1}^{n} w_i x_i \le C$$

$$x_i \in \{0,1\}, \quad 1 \le i \le n$$



枚举法需要枚举2^n个装取方案

0/1背包问题

- 优化子结构

定理.

设$(x_1,x_2,\cdots,x_n)$是0/1背包问题的最优解，则$(x_2,\cdots,x_n)$为如下子问题的最优解，

$$
\begin{aligned}
\max \quad & \textstyle\sum_{i=2}^{n} v_i x_i \\
s.t. \quad & \textstyle\sum_{i=2}^{n} w_i x_i \le C - w_1 x_1 \\
& x_i \in \{0,1\}, \quad 2 \le i \le n
\end{aligned}
$$

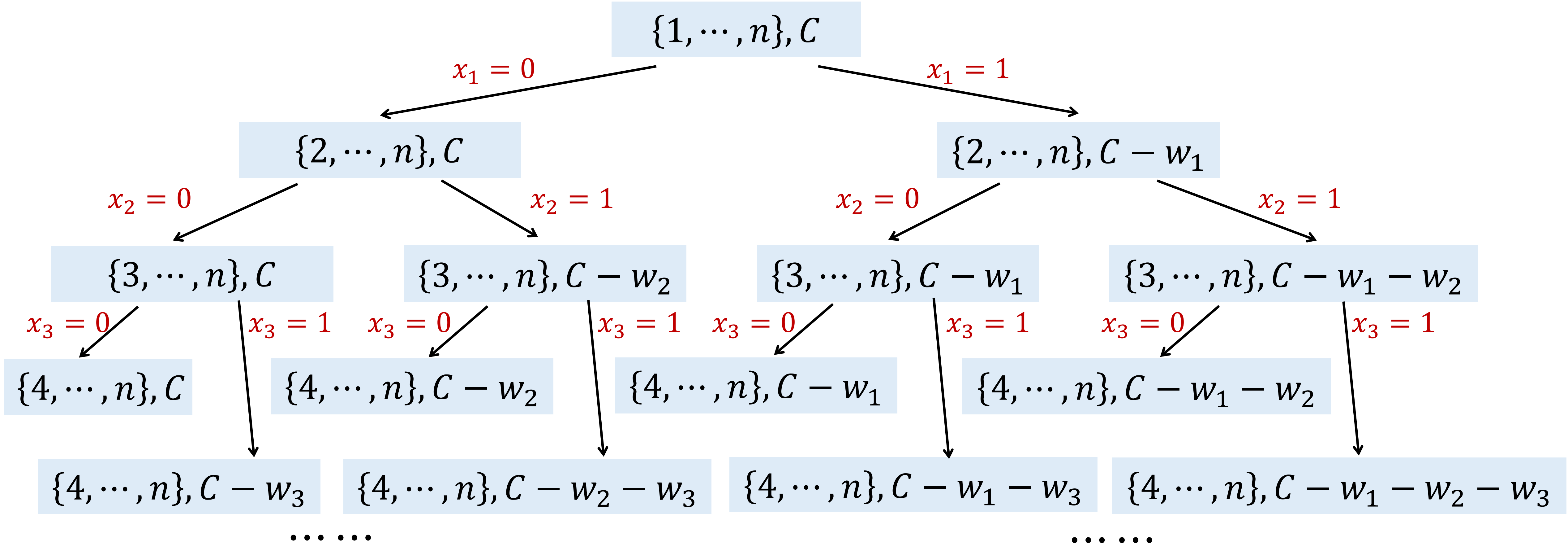
证明.

反证法。若$(x_2,\cdots,x_n)$不是该子问题的最优解，则存在子问题的最优解$(z_2,\cdots,z_n)$，那么以$(x_1,z_2,\cdots,z_n)$装背包获得的价值大于以$(x_1,x_2,\cdots,x_n)$装背包获得的价值，与$(x_1,x_2,\cdots,x_n)$是最优解矛盾，得证。

具有优化子结构：问题的最优解包含子问题的最优解

0/1背包问题

- 子问题重叠性



当w_i皆为1时，存在大量重叠子问题

0/1背包问题

- 递归方程
  - 针对子问题

$$\max \quad \sum_{k=i}^{n} v_k x_k$$
$$s.t. \quad \sum_{k=i}^{n} w_k x_k \le j$$
$$x_k \in \{0,1\}, \quad i \le k \le n$$

  - 其最优解的价值记为$m(i,j)$，表示将物品$i,\cdots,n$装入剩余容量为j的背包获得的最大价值，即$m(i,j)=\sum_{k=i}^{n} v_k x_k$，使得$\sum_{k=i}^{n} w_k x_k \le j$
    - $m(i,j)=m(i+1,j), \qquad 0 \le j < w_i$
    - $m(i,j)=\max\{m(i+1,j),\ m(i+1,j-w_i)+v_i\}, \qquad j \ge w_i$
    - $m(n,0)=0, \qquad 0 \le j < w_n$
    - $m(n,j)=v_n, \qquad j \ge w_n$

0/1背包问题

$$
\begin{aligned}
&m(i,j) = m(i+1,j), && 0 \le j < w_i \\
&m(i,j) = \max\{m(i+1,j),\ m(i+1,j-w_i)+v_i\}, && j \ge w_i \\
&m(n,0) = 0, && 0 \le j < w_n \\
&m(n,j) = v_n, && j \ge w_n
\end{aligned}
$$

- 递归方程

	$\boldsymbol{m(i,j)}$
$\boldsymbol{m(i+1,j-w_i)}$	$\boldsymbol{m(i+1,j)}$

$m(1,C)$

$m(2,C-w_1)$　$m(2,C)$

$m(3,C-w_1-w_2)$　$m(3,C-w_2)$　$m(3,C-w_1)$　$m(3,C)$

0/1背包问题

伪多项式算法！

1. 当$C = O(2^n)$时，$T(n) = O(n2^n)$
2. 当w_i不限定为正整数时，$T(n) = O(2^n)$

- 算法伪代码

```
0-1-Knapsack (C, W, V, n)
for j = 1 to min(w_n − 1, C) do
    m(n, j) = 0;
for j = w_n to C do
    m(n, j) = v_n;
for i = n − 1 to 2 do
    for j = 0 to min(w_i − 1, C) do
        m(i, j) = m(i + 1, j);
    for j = w_i to C do
        m(i, j) = max{m(i + 1, j), m(i + 1, j − w_i) + v_i};
if C < w_1 then m(1, C) = m(2, C);
else m(1, C) = max{m(2, C), m(2, C − w_2) + v_2};
return m;
```

```
Construct-0-1 (C, W, M, n)
j = C;
for i = 1 or n − 1 do
    if m(i, j) = m(i + 1, j) then x_i = 0;
    else x_i = 1; j = j − w_i;
if j < w_{n−1} then x_n = 0;
else x_n = 1;
return (x_1, ⋯, x_n);
```

时间复杂度$O(Cn)$

空间复杂度$O(Cn)$

作业&练习

1. 给出N个1-9的数字 $(v_1, \cdots, v_n)$，不改变它们的相对位置，在中间加入K个乘号和$N-K-1$个加号，（括号随便加）使最终结果尽量大，并说明其具有优化子结构性质及子问题重叠性。
   - 因为乘号和加号一共就是N-1个了，所以恰好每两个相邻数字之间都有一个符号。
   - 例：$N=5, K=2, (v_1, \cdots, v_n)=(1,2,3,4,5)$ 可以加成：

     (a). $1 \times 2 \times (3+4+5)=24$ (b). $1 \times (2+3) \times (4+5)=45$ (c). $(1 \times 2+3) \times (4+5)=45$

2. 给定长度为N的整数序列$(a_1, \cdots, a_n)$，将其划分成多个子序列（连续的一段整数），满足每个子序列中整数的和不大于一个数B，设计一种划分方法，最小化所有子序列中最大值的和，说明其具有优化子结构及子问题重叠性。
   - 例：整数序列$(2,2,2,8,1,8,2,1)$，$B=17$，可将其划分成三个子序列$(2,2,2)$；$(8,1,8)$；$(2,1)$，则可满足每个子序列中整数和不大于17，所有子序列中最大值的和12为最终结果。

作业&练习

3. 对一棵树进行着色，每个结点可着黑色或白色，相邻结点不能着相同黑色，但可着相同白色。令树的根为r，设计一种算法对树中尽量多的节点着黑色。

4. 在自然语言处理中一个重要的问题是分词，例如句子“他说的确实在理”中“的确”“确实”“实在”“在理”都是常见词汇，但是计算机必须为给定的句子准确判断出正确分词方法。一个简化的分词问题如下：给定一个长字符串$\boldsymbol{y} = (y_1, \cdots, y_n)$，分词是把其切分成若干连续部分，每部分都单独成为词汇。我们用函数$quality(x)$判断切分后的某词汇$\boldsymbol{x} = (x_1, \cdots, x_k)$的质量，函数值越高表示该词汇的正确性越高。分词的好坏用所有词汇的质量的和来表示。例如对句子“确实在理”分词，$quality$(确实) + $quality$(在理) > $quality$(确) + $quality$(实在) + $quality$(理)。请设计一个动态规划算法对字符串y分词，要求最大化所有词汇的质量和。（假定你可以调用$quality(x)$函数在一步内得到任何长度的词汇的质量）

作业&练习

5. 给定 n个活动，活动a_i表示为一个三元组(s_i, f_i, v_i)，其中s_i表示活动开始时间，f_i表示活动的结束时间，v_i表示活动的权重。带权活动选择问题是选择一些活动，使得任意被选择的两个活动a_i和a_j执行时间互不相交，即区间$[s_i, f_i]$与$[s_j, f_j]$互不重叠，并且被选择活动的权重和最大。请设计一种方法求解带权活动选择问题。

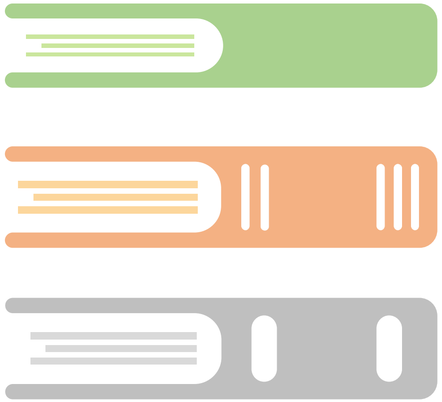
6. 受限最短路径长度问题：给定一无向图$G = (V, E, A, B)$，$A(e)$表示边e的长度，$B(v)$表示顶点v的花费，计算小明从顶点s到顶点d的最短路径长度，满足以下限制，初始时小明随身携带M元钱，每经过一个顶点v，须交$B(v)$的过路费，若身上有大于$B(v)$的钱则可以通过，否则不可以通过。求顶点s到顶点d的最短路径。

作业&练习

7. 给定n个物品，每个物品重量为s_i，价值v_i，背包容量为C。要求找到一组物品，这些物品整包完全占满背包容量C，且总体价值最大。请写出动态规划迭代公式。

8. 最大子数组问题：一个包含n个整数（有正有负）的数组A，设计$O(n\log n)$的算法找出和最大的非空连续子数组。（例如：$[0,-2,3,5,-1,2]$应返回9，$[-9,-2,-3,-5,-3]$应返回-2。）

9. 最长非降子序列：一个序列有N个数：$A[1],A[2],\ldots,A[N]$，求出最长非降子序列的长度。



Q&A