动态规划算法的递归实现

部分伪码

子问题 i-j

算法1 **RecurMatrixChain** (P, i, j)

1. $m[i,j] \leftarrow \infty$
2. $s[i,j] \leftarrow i$
3. **for** $k \leftarrow i$ **to** $j-1$ **do**
4. $\quad q \leftarrow$ **RecurMatrixChain**(P,i,k) $+$ **RecurMatrixChain**$(P,k+1,j) + p_{i-1}\, p_k\, p_j$
5. $\quad$ **if** $q < m[i,j]$
6. $\quad$ **then** $m[i,j] \leftarrow q$
7. $\qquad\qquad s[i,j] \leftarrow k$
8. **return** $m[i,j]$

划分位置k

找到更好的解

这里没有写出算法的全部描述（递归边界）

算法分析

时间复杂度的递推方程

$$T(n) \geq \begin{cases} O(1) & n=1 \\ \sum_{k=1}^{n-1}(T(k)+T(n-k)+O(1)) & n>1 \end{cases}$$

$$T(n) \geq O(n)+\sum_{k=1}^{n-1}T(k)+\sum_{k=1}^{n-1}T(n-k)$$

$$\boxed{T(n) \geq O(n)+2\sum_{k=1}^{n-1}T(k)}$$

时间复杂度

数学归纳法证明：$T(n) \ge 2^{n-1}$

n=2, 显然为真.

假设对于任何小于n 的 k, 命题为真

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

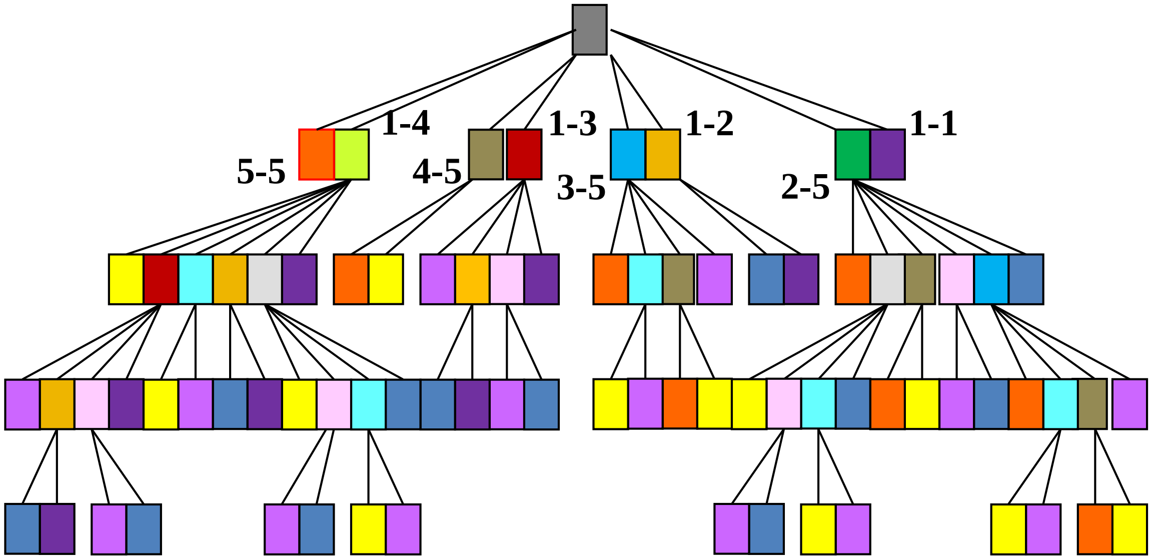
代入归纳假设

$$\ge O(n) + 2\sum_{k=1}^{n-1} 2^{k-1}$$

等比数列求和

$$= O(n) + 2(2^{n-1} - 1) \ge 2^{n-1}$$

子问题的产生, *n*=5



划分： $A_{1..4}A_{5..5}$, $A_{1..3}A_{4..5}$, $A_{1..2}A_{3..5}$, $A_{1..1}A_{2..5}$

产生 8 个子问题，即第一层的 8 个结点.

子问题的计数

边界	次数	边界	次数	边界	次数
1-1	**8**	1-2	**4**	2-4	**2**
2-2	**12**	2-3	**5**	3-5	**2**
3-3	**14**	3-4	**5**	1-4	**1**
4-4	**12**	4-5	**4**	2-5	**1**
5-5	**8**	1-3	**2**	1-5	**1**

边界不同的子问题：**15** 个

递归计算的子问题：**81** 个

结论

- 与蛮力算法相比较，动态规划算法利用了子问题优化函数间的依赖关系，时间复杂度有所降低
- 动态规划算法的递归实现效率不高，原因在于同一子问题多次重复出现，每次出现都需要重新计算一遍.
- 采用空间换时间策略，记录每个子问题首次计算结果，后面再用时就直接取值，每个子问题只算一次.