Open Topic
通讯系统

曹恒源

Open Topic 1 通讯系统

- 某个通信系统由 n 个设备串联构成，每个设备可能有多个厂商生产，均有带宽和价格参数。系统的总带宽决定于某个设备的最小带宽，总价格是各个设备的价格总和。请你设计一个算法，以 “带宽/造价” 为最优目标，确定该通信系统的构成

Content

- 能否使用动态规划
- 最优子结构确定
- 确定递归表达式
- 非递归实现

能否使用动态规划

In this section, we examine the two key ingredients that an optimization problem must have in order for dynamic programming to apply: optimal substructure and overlapping subproblems.

--Introduction to Algorithms

能否使用动态规划

In this section, we examine the two key ingredients that an optimization problem must have in order for dynamic programming to apply: optimal substructure and overlapping subproblems.

--Introduction to Algorithms

能否使用动态规划-子问题重叠

- 暴力求解法

我们考虑这样一个通讯系统，一共需要三个站点，每个站点有两个可选的设备，来考虑这其中子问题的重叠情况。


A1

B1

C1


A2

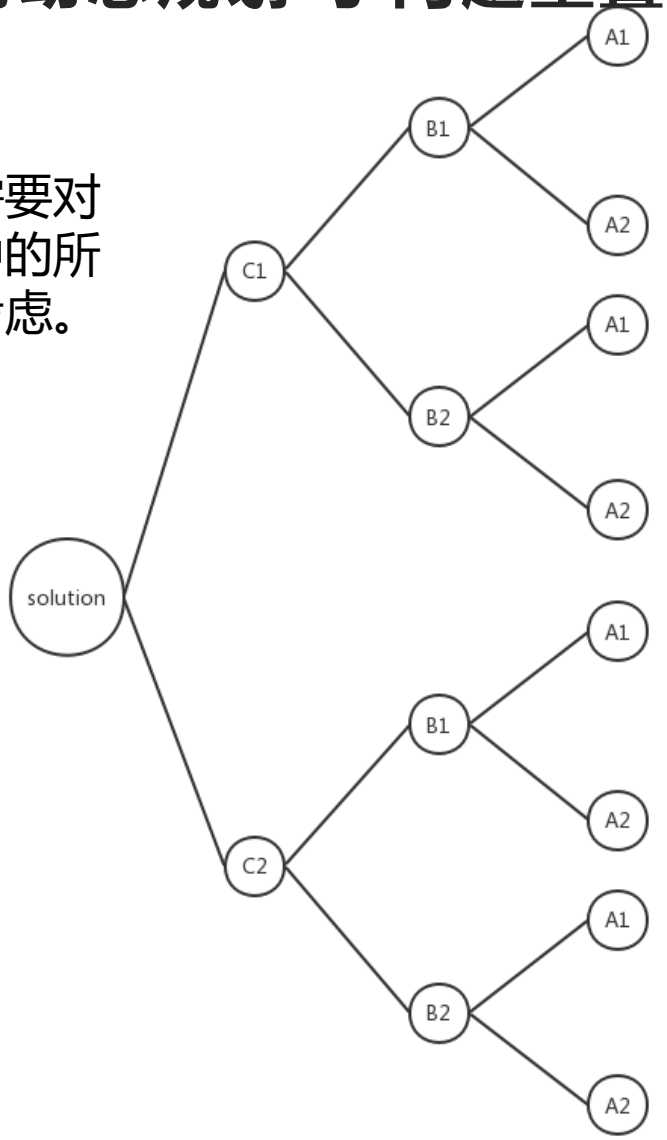
B2

C2

能否使用动态规划-子问题重叠

- 暴力求解法

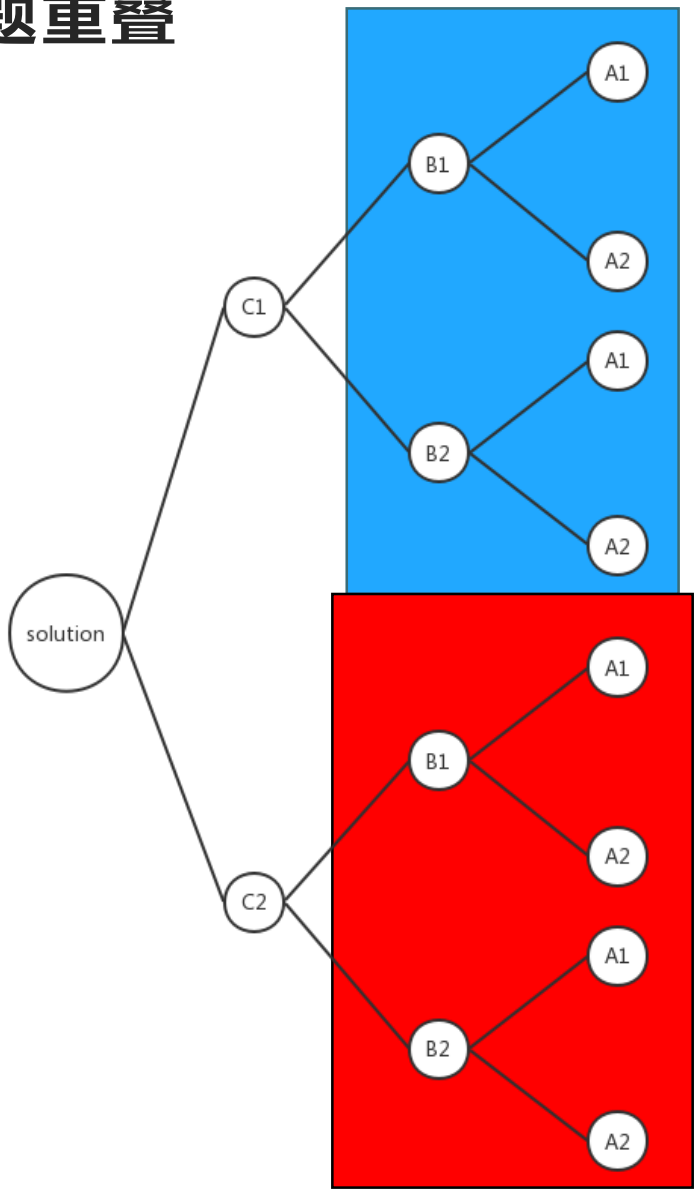
暴力求解法需要对这个树状图中的所有情况进行考虑。



能否使用动态规划-子问题重叠

- 暴力求解法

如图红色和蓝色都是重叠的子问题。



能否使用动态规划-最优子结构

B:带宽

P:价格

下面我们再来考量可以用动态规划求解的第二个要素：最优子结构。

我们再来考量这样一个问题，一共有两个站点，每个站点可选设备的带宽和价格如图。

B:100

P:50

B:2

P:100

B:3

P:10

能否使用动态规划-最优子结构

B:带宽

P:价格

B:100

P:50

B:2

P:100

B:3

P:10

如果按照最优子结构来计算，那么第一个站点应该选第一个设备。

能否使用动态规划-最优子结构

B:带宽

P:价格

B:100

P:50

B:2

P:100

B:3

P:10

但是整体来看第一个站点应该选择第二个设备。

能否使用动态规划

仍然可以使用动态规划！

能否使用动态规划

不把B/P作为我们想要的最终结果

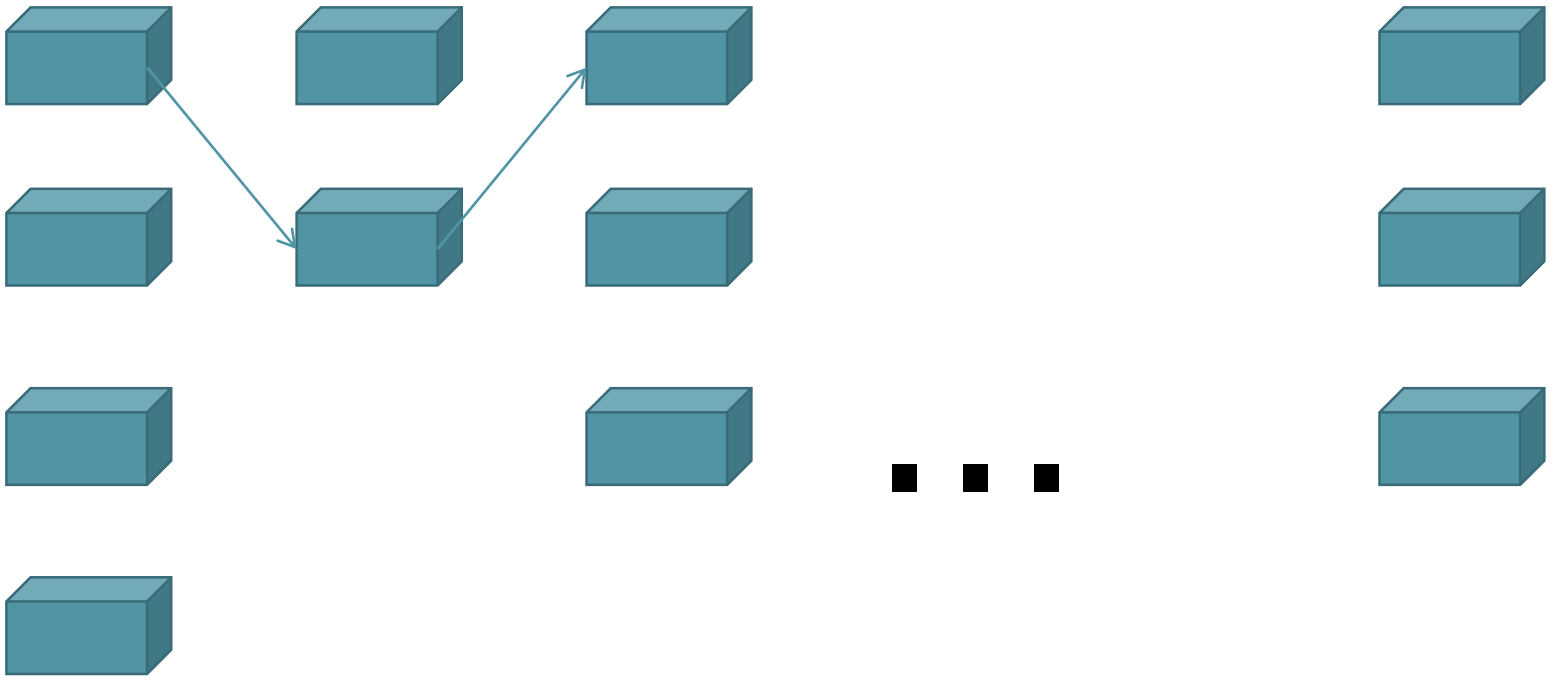
针对每一个可能的带宽给出造价的最小值

最后计算B/P，求出其最大值

最优子结构确定

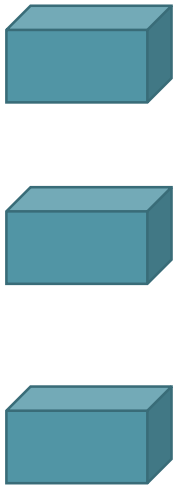
- 反证法
- CUT-PASTE策略

最优子结构确定



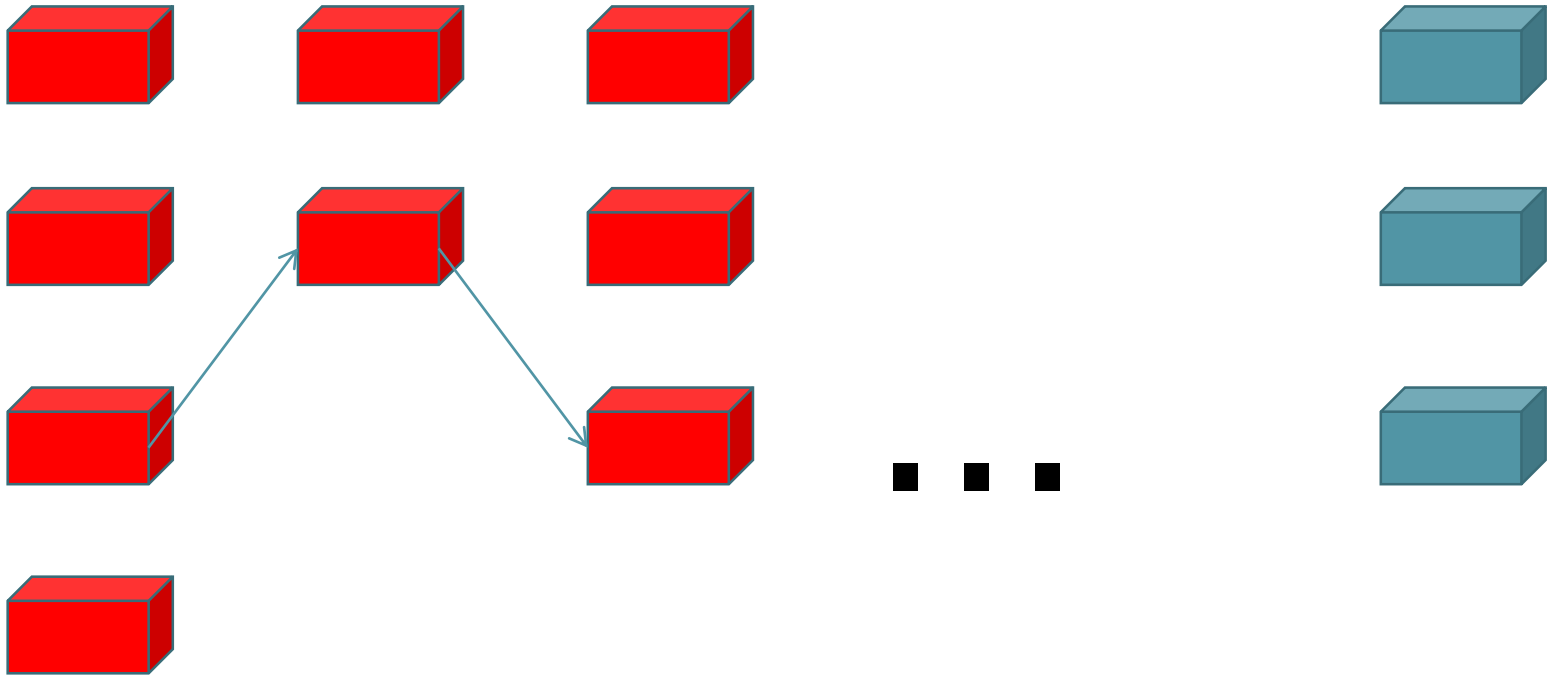
假设最终最优答案中含有一个不是最优的子问题。

最优子结构确定



我们将这个不是最优情况的子问题删去。

最优子结构确定



用这个子问题的最优解代替它。

我们会发现这个问题的答案变得更优了，这与原答案是最优解矛盾，所以该问题具有最优子结构。

确定递归表达式

针对每一种可能的带宽

$dp[i][j]$ 第i个设备，带宽为j的最小造价

p$[i][j]$ 第i个设备，第j个厂商给出的价格

b$[i][j]$ 第i个设备，第j个厂商给出的带宽

$$dp[i][k]=\min(\min_{\forall j,b[i][j]>k}(dp[i-1][k]+p[i][j]),\min_{\forall j,b[i][j]=k,\forall t,t\geq k}(dp[i-1][t]+p[i][j]))$$

确定递归表达式

$$dp[i][k]=\min(\min_{\forall j,b[i][j]>k}(dp[i-1][k]+p[i][j]),\min_{\forall j,b[i][j]=k,\forall t,t\geq k}(dp[i-1][t]+p[i][j]))$$

四重循环

上述递归表达式过于复杂

$$dp[i][b]=\min(dp[i-1][k]+p[i][j]),b=\min(b[i][j],k)$$

三重循环

确定递归表达式

$$dp[i][b] = \min(dp[i-1][k] + p[i][j]), b = \min(b[i][j], k)$$

边界情况

$$dp[1][b[1][j]] = p[1][j]$$

非递归实现

man[i][k]: 带宽为k时，选择的厂商，用于最后输出最优解

非递归实现

初始化

```
for i from 1 to n
    for j from b_min to b_max    //b_min和b_max分别指所有带宽中的最小值和最大值
        dp[i][j] = INF

for j from 1 to m[1]        //i=1时的边界情况    m[i]用来记录第i个设备有多少个厂商
    dp[1][b[1][j]] = p[1][j]
    man[1][b[1][j]] = j        //第一个设备选择的厂商
```

非递归实现

主体

```
for i from 2 to n
    for j from 1 to m[i]
        for k from b_min to b_max
            if k <= b[i][j]        //最小带宽不变
                if dp[i][k] > dp[i-1][k] + p[i][j]
                    dp[i][k] = dp[i-1][k] + p[i][j]
                    man[i][k] = j
            else        //最小带宽改变
                if dp[i][b[i][j]] > dp[i-1][k] + p[i][j]
                    dp[i][b[i][j]] = dp[i-1][k] + p[i][j]
                    man[i][b[i][j]] = j
```

非递归实现

输出

```
ans = 0
for i from b_min to b_max
    if dp[n][i] != INF && i / dp[n][i] > ans
        width = i
        ans = k / dp[n][i]

for i from 1 to n
    print(“man[i][width]  ”)
print(“the final answer is ans”)
```

有没有其他方法

这道题还可以使用贪心法来做，有兴趣的同学可以试一下

- 参考资料
- 算法导论

- 感谢 马老师的指导