充分利用问题性质

——例析动态规划的“个性化”优化

华东师大二附中
项荣璟

动态规划的优化

- 优化势在必行。
- 一些适用一类状态转移方程的优化：利用四边形不等式、函数的凸性等。
- 大多数状态转移方程的求解需要采用“个性化”的优化手段。

动态规划的优化

- 优化的关键：减少冗余。

冗余
- 重叠子问题 → 空间换取时间，利用已知结果
- 无用的结果 → 分析问题性质，排除不必要的计算量

问题一——书稿复制(cerc98)

- n 本书，编号 1,2,…,n 。每本 p_i 页。
- 全部分给 m 个抄写员。每人分到顺序连续的若干本，每本只分给一人。
- 求一种方案，使每人分到的页数和的最大值为最小。
- 例子： n=9,m=3
- 100 200 300 400 500 / 600 700 / 800 900

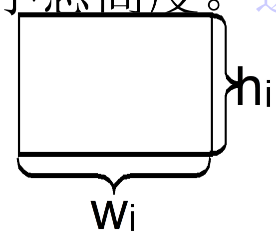
问题二——工作分批(ioi2002)

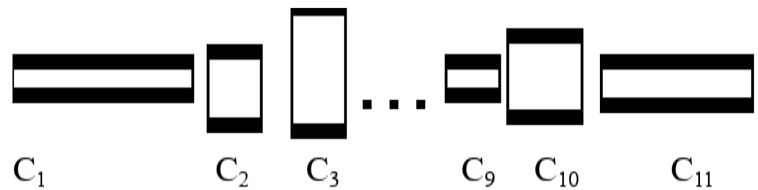
- n 项工作，编号为 1,2,…,n 。给定每项工作花费系数 F_i 和所需时间 T_i 。
- 可按序分成任意多批依次执行，每批包含编号连续的工作。
- 第一批开始于时间 0 。若某批包含工作 i,i+1,…,j ，开始于时间 t ，则该批中所有工作的完成时间是 $t+s+(T_i+T_{i+1}+\ldots+T_j)$ 。这也是下一批的开始时间。
- 一个工作的花费是其完成时间 $*F_i$ ，求最小可能的总花费。
- 例子： n=5,s=1　　$(T_1,T_2,\ldots,T_5)=(1,3,4,2,1)$　$(F_1,F_2,\ldots F_5)=(3,2,3,3,4)$

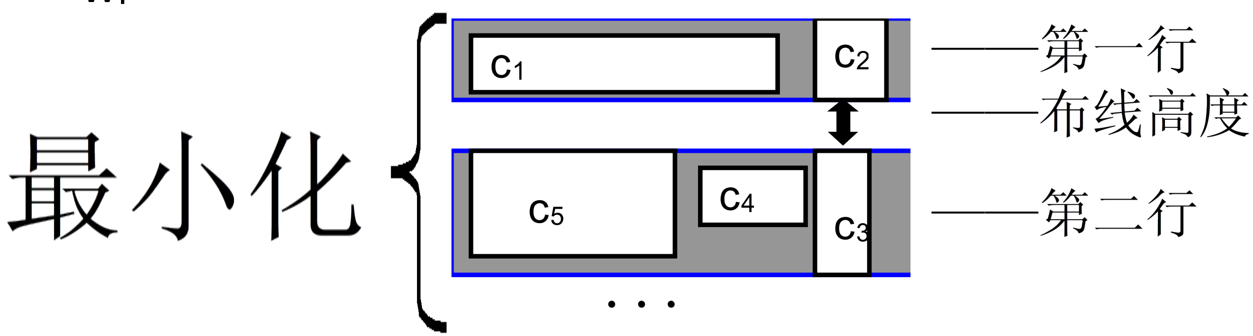
可分成三批 {1,2}{3}{4,5} ，完成时间为 (5,5,10,14,14) ，总花费 153 。

问题三——元件折叠

- 线性排列的元件 $C_1,C_2,\ldots,C_n$ 。 C_i 宽 w_i ， 高 h_i 。
- 要折叠成宽度为 W 的若干行（即每行元件总宽度≤ W)，每行高度为该行中最高元件高度。行与行之间为布线通道，若 C_i 与 C_{i+1} 之间折叠，则它们所在行之间布线高度为 l_i ， $l_n=0$ 。
- 求最小总高度。进入







“通用”的解法

- 共同点：给定一个序列，求一种满足一些条件的最优化划分，使题中定义的某种“花费”最小。
- 面对这三个相似的问题，我们大多会采用模式化的方法：

  以序列每个数为阶段

  以此前的每个数为最近的划分点

  按状态转移方程判断

  若给定划分的区间数目，则增加一维。
- 问题一 $O(n^3)$，问题二 $O(n^2)$，问题三 $O(n^2)$。

问题一——方程

- f(i,j)： $p_i+p_{i+1}+\ldots+p_j$ 。
- g(i,k)： 在将 [i,n] 中的数分成 k 份的最优划分中，花费最大区间的花费值。

$$g(i,1)=f(i,n)$$

$$g(i,n-i+1)=\max_{i\le j\le n} f(j,j)$$

$$g(i,k)=\min_{i\le j\le n-k+1}\max\{f(i,j),g(j+1,k-1)\}$$

问题一——分析（1）

如果

- j 是 [i+1, n] 的第一个划分点（即动态规划的决策）
- [i, j] 的花费不大于 [j+1, n] 中花费最大区间的花费值

那么： *j 也是 [i, n] 的第一个划分点。*

- **性质一： [i+1,j] 是 g(i+1,k) 对应的分划中的第一个区间，如果 f(i,j) ≤g(j+1, k-1) 那么 g(i, k)=g(j+1, k-1)，即 g(i, k)=g(i+1, k)。**

问题一——分析（2）

- **转折点是第一个这样的划分点 j，它使 [i, j] 的花费为 [i, n] 中所有区间花费的最大值。**
- 形式化定义为：

令 $0 \leqslant i \leqslant n, 2 \leqslant k \leqslant m, i \leqslant s_{i,k} \leqslant n$，如果 $f(i, s_{i,k}-1) < g(s_{i,k}, k-1)$ 且 $f(i, s_{i,k}) \geqslant g(s_{i,k}+1, k-1)$，则 $s_{i,k}$ 是一个“转折点”。

- **性质二：对 $0 \leqslant i \leqslant n, 2 \leqslant k \leqslant m$，**

转折点唯一存在

问题一——分析（3）

- 性质三：对 $k \geq 2$，

$$g(i,k)=\min\{f(i,s_{i,k}),g(s_{i,k},k-1)\}$$

最优决策是转折点或它之前的一点

问题一——分析（4）

- 计算 $g(i,k)$：
- $[i+1,j]$ 是 $g(i+1,k)$ 对应划分的第一个区间。
- $f(i,j)\leqslant g(j+1,k-1)$ 则根据性质一有 $g(i,k)=g(j+1,k-1)$；
- 否则 $f(i,j)>g(j+1,k-1)$，又根据 $s_{i,k}$ 定义及性质二，有 $\mathbf{i\leqslant s_{i,k}\leqslant j}$，从而容易确定 $s_{i,k}$，继而应用性质三。

问题一——算法分析

```
for i:=n downto 1 do g(i,1):=f(i,n);     ｛边界条件｝
for k:=2 to m do
      计算边界 g(n-k+1,k);  j:=n-k+1;
   for i:=n-k downto m-k+1 do
     if f(I,j)<=g(j+1,k-1) then g(i,k):=g(j+1,k-1) ｛性质一｝
      else if f(i,i)>=g(i+1,k-1) then 【  g(i,k):=f(i,i); j:=i 】
             else 【 while f(i,j-1)>=g(j,k-1) do j:=j-1; { 定 si,k}
                    g(i,k):=min{f(i,j),g(j,k-1)}  ｛性质三｝
                    if g(i,k)=g(j,k-1) then j:=j-1 】
end_for_k
```

- 外层每循环一次， j 递减的工作量是 O(n) 。因此总的复杂度 $O(n^2)$ 。

问题一——小结

分析问题性质：

- 深入挖掘题意

  寻找在最优性和可行性的约束下中间结果必须满足的必要条件。

- 由浅入深

  将不成熟的想法转化为言之有理的论断。

问题三

问题二——方程

- $t_{i,j}=T_i+T_{i+1}+\ldots+T_j$； $f_i=F_i+F_{i+1}+..+F_n$ 。
- D(i)： 划分 [i,n] 的最小总花费。
- C(i,k)： 划分 [i,n] 时第一个区间是 [i,k-1] 的最小总花费。则 $C(i,k)=D(k)+(S+t_{i,k-1})\times f_i$。

$$D(i)=\min_{i+1\le k\le n+1}\{C(i,k)\}$$

$$D(n)=0$$

问题二——分析（1）

- 对于 i<k<l,

$$C(i,k) \le C(i,l) \Leftrightarrow D(l) - D(k) + t_{k,l-1} \times f_i \ge 0$$
$$\Leftrightarrow (D(k) - D(l)) / t_{k,l-1} \le f_i$$

令 $g(k,l)=(D(k)-D(l))/t_{k,l-1}$

- **性质一：** $1 \leqslant i < k < l$，如果 $g(k,l) \leqslant f_i$，那么 $C(i,k) \leqslant C(i,l)$。
- 注意到 $1 \leqslant j < i$ 时，$f_j < f_i$ 于是有

  推论一：如果 $g(k,l) \leqslant f_i$，那么对所有 $1 \leqslant j \leqslant i$，$C(j,k) \leqslant C(j,l)$。
- 推论一暗示了计算 D(i),D(i-1),…D(1) 时都不必考虑在 l-1 处划分。

问题二——分析（2）

- **性质二：对 $1 \leqslant j<k<l$，若 $g(j,k) \leqslant g(k,l)$，则对所有 $1 \leqslant i<j$，有 $C(i,j) \leqslant C(i,k)$ 或 $C(i,l) \leqslant C(i,k)$ 成立。**
- 性质二暗示了，若存在 $g(j,k) \leqslant g(k,l)$，则计算 $D(j-1), D(j-2), \cdots, D(1)$ 时，是不必考虑在 $k-1$ 处的划分的。
- 由推论一和性质二可知，在倒推过程中，在第 i 阶段，在 $D(i+1), D(i+2), \cdots, D(n)$ 中只有一部分可能对计算 $D(i), D(i-1), \cdots, D(1)$ 有用。将这些可取的阶段记为 $i_1, i_2, \cdots, i_r$ 且 $i_1>i_2>\cdots>i_r$，则必须满足：

$$f_i<g(i_2, i_1)<g(i_3, i_2)<\ldots<g(i_r, i_{r-1})$$

问题二——分析（3）

- $f_i < g(i_2, i_1) < g(i_3, i_2) < \ldots < g(i_r, i_{r-1})$
- 由上式和性质一， $C(i, i_1) < C(i, i_2) < \cdots < C(i, i_r)$
- 从而 $D(i) = C(i, i_1)$ 。
- 动态维护 $i_1, i_2, \cdots, i_r$ 的数据结构：

  线性表 lst ，头指针 head ，尾指针 tail 。表头表尾删除，表尾添加。

问题二——算法分析

```
head:=1; tail:=1; lst[1]:=n+1; c[n+1]:=0; {表初始化}
for i:=n downto 1 do
    while (head<tail) and  (f_i>=g(lst[head+1],lst[head])) do
        inc(head);    {按推论一删除}
      D(i):=C(i,lst[head]);
      while (head<tail) and
            (g(i,lst[tail])<=g(lst[tail],lst[tail-1])) do
          dec(tail);    {按性质二删除}
     inc(tail); lst[tail]:=i
endfor
```

- 由于每个元素进出 lst 各一次，复杂度 O(n)。

问题三——方程

- $w(i,j)=w_i+w_{i+1}+\ldots+w_j$
- $R(i,j)=\max\limits_{i\le k\le j}\{h_k\}$
- F(i)：划分区间 [i+1,n] 的最小总花费。若定义 $l_0=0$，则 F(0) 为答案。

$$F(i)=l_i+\min_{i+1\le j\le n,\, w(i+1,j)\le W}\{F(j)+R(i+1,j)\} \qquad 0\le i<n$$

$$F(n)=0$$

问题三——分析（1）

- S_i：划分 [i+1,n] 时所有可行决策集合
- S_i 首先要满足：若 j 是 S_i 中元素，则 w(i+1,j)>W 。S_i 可由 S_{i+1} 得到：从 S_{i+1} 中除去 w(i+1,j)>W 的元素 j，再添加 i 。

注意到 j<i 时， w(j,k)>w(i,k) 及 R(j,k)≥ R(i,k)，则有

性质一：如果 a,b 是 S_i 中元素，且 a<b， F(a)≤F(b)，则可以将 b 从 S_i 中除去而不会影响 F(i),F(i-1),…,F(0) 的计算。

如何维护可行决策集 S？

选一种数据结构，支持两种删除和一种插入操作

问题三——分析（2）

- 用线性表 lst 表示可行决策集 S：

  lst[head]>lst[head+1]>…>lst[tail]

  F(lst[head])<F(lst[head+1])<…<F(lst[tail])

- 根据 w(i,j) 在 i<j≤n 上单调增，从表头删数能完成删除一；从表尾删数能完成删除二，然后将 i 添加到表尾，依然能保持表中元素有序性。

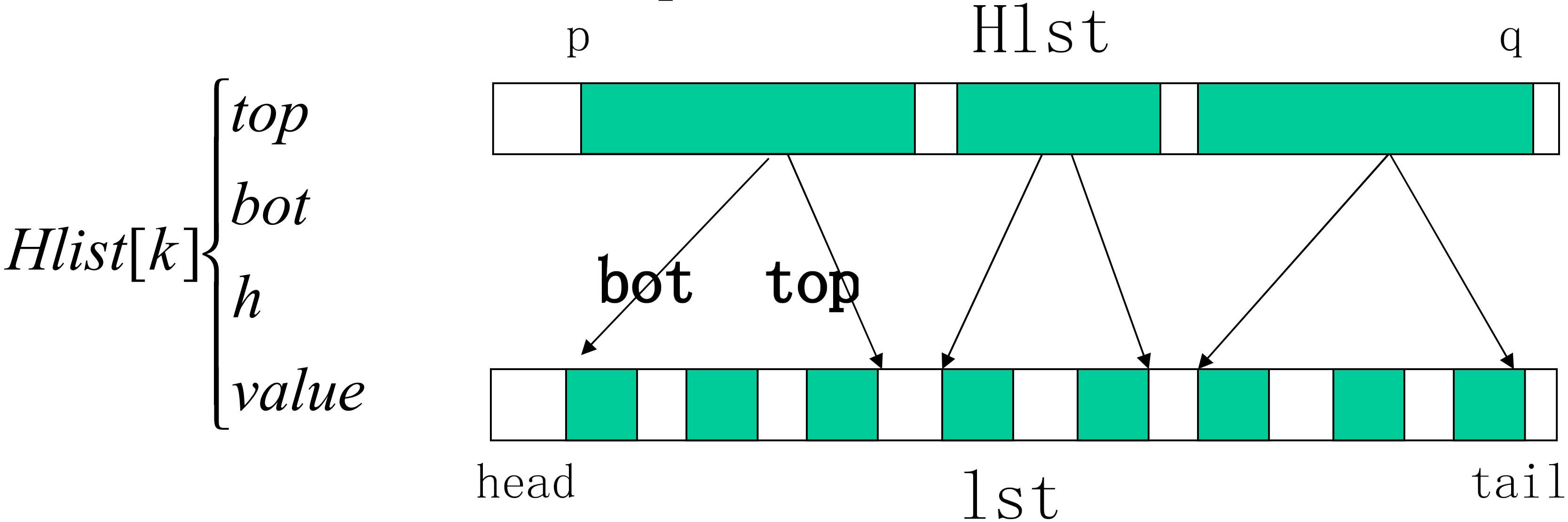
问题三——分析（3）

$$F(i)=l_i+\min_{j\in S_i}\{F(j)+R(i+1,j)\}$$

- **性质二： i≤j ，如果 $h_{j+1}\le h_j$ ，则 R(i,j+1)=R(i,j)**
- 考虑在计算 F(i) 时，将 F(j) 与 R(i+1,j) 联系起来。
- 由性质二连续的 R 值可能是相等的，即 R(i+1,j)=R(i+1,j+1)=…=R(i+1,k)=h ，此时我们可以将 F(j),F(j+1),…,F(k) 都与 h 相联系。

问题三——分析（4）

- 维护一个表 Hlst，表头指针 p，表尾指针 q。
- 在递推到第 i 阶段，bot…top：S_i 中第 bot 到第 top 个元素（即：lst[bot],lst[bot+1],…,lst[top]) 都与 h 联系。
- Hlst[k].bot=Hlst[k-1].top+1。



问题三——分析（5）

- F(lst[Hlist[k].bot])<F(lst[Hlst[k].bot+1)<…<F(lst[Hlist[k].top])
- Hlst[k].value=Hlst[k].h+F(lst[Hist[k].bot])

$$F(i) = l_i + \min_{p \le k \le q}\{Hlst[k].value\}$$

- value 值不断更新，在求 F(n),F(n-1),…,F(0) 时都要察看最小的 value。

堆！

问题三——算法分析

- 依次求 F(n),F(n-1),…F(0) 。
- 每次只从 lst 表头或表尾开始删除元素，对应地也只须从 Hlst 表头或表尾开始移动 bot 或 top 指针， bot>top 时删除 value ，且移动 p 或 q 。当改变 bot 指针时，须改变 value 值；
- 将 i 添加到 lst 表后，也更新 Hlst 的表尾，然后从 Hlst 表尾开始不断按照性质二合并表中元素，使 Hlst[q].h<Hlst[q-1].h<…<Hlst[p].h 。
- 某个 value 值被改变、添加或删除时，同时调整堆。
- 堆调整一次的复杂度是 O(log n) 。每个元素进出 lst 各一次， Hlst 的维护与 lst 是同步的， Hlst 合并的总次数不会超过 n ，因此堆调整的次数 O(n) 。总的时间复杂度： O(n log n) 。

问题三——小结

根据问题性质选择恰当的数据结构：

- 扎实的基础
- 灵活和富有创造性的运用能力

本题的数据结构：

- lst 是观察到性质 1 而设的有序表，它结构上既不同于队列又不同于栈。满足了动态规划各阶段对插入和删除候选决策的需要。
- Hlst 利用了性质 2 合并 lst 中元素，从而保证了堆调整的次数为 O(n) 。
- 堆的设置是建立在对基础数据结构的算法和性能充分熟悉的基础上的。本例对堆的维护彻底解决了每阶段的决策问题。

总结

	书稿复制	工作分批	元件折叠
优化前	$O(n^3)$	$O(n^2)$	$O(n^2)$
优化后	$O(n^2)$	$O(n)$	$O(n \log n)$

- 优化前三个问题算法描述大同小异，优化后迥然不同。
- 我们已充分挖掘并利用了问题的“个性”。

总结

磨刀不误砍柴功

动态规划的个性化优化

仔细分析问题性质

↓

根据问题性质选择恰当的数据结构

深入思考 把握本质

因题制宜 不拘一格

将优化落到实处

谢谢！