实验舱
青少年编程
走近科学 走进名校

蛟龙五班

动态规划基础

Mas

动态规划

动态规划是一种通过把原问题分解为相对简单的子问题的方式求解复杂问题的方法

能用动态规划解决的问题,需要满足三个条件：最优子结构,无后效性和子问题重叠

对于一个能用动态规划解决的问题,一般采用如下思路解决：

- 将原问题划分为若干**阶段**,每个阶段对应若干个子问题,提取这些子问题的特征(称之为**状态**)
- 寻找每一个状态的可能**决策**,或者说是各状态间的相互转移方式(状态转移方程)
- 按顺序求解每一个阶段的问题

用图论的思想理解,建立一个DAG,每个状态对应图上一个节点,决策对应节点间的连边

问题就转变为了一个在DAG上寻找最长/短路的问题

动态规划

最优子结构

具有最优子结构也可能是适合用贪心的方法求解,要确保考察了最优解中用到的所有子问题

- 证明问题最优解的第一个组成部分是做出一个选择(存在子结构)
- 对于一个给定问题,在其可能的第一步选择中,界定已经知道哪种选择才会得到最优解(并不需要选择具体如何得到)
- 给定可获得的最优解的选择后,确定这次选择会产生哪些子问题,以及如何最好地刻画子问题空间
- 证明作为构成原问题最优解的组成部分,每个子问题的解就是它本身的最优解方法

考虑加入某个子问题的解不是其自身的最优解,那么就可以从原问题的解中用该子问题的最优解替换掉当前的非最优解

从而得到原问题的一个更优的解,从而与原问题最优解的假设矛盾(反证法)

无后效性

已经求解的子问题,不会再受到后续决策的影响

子问题重叠

若有大量的重叠子问题,可以用空间将这些子问题的解存储下来,避免重复求解相同的子问题,从而提升效率

#1898、硬币问题

题目描述

有 n 种硬币,面值分别为 $v_1, v_2, \ldots, v_n$,每种都有无限多

给定非负整数 S ,可以选用多少个硬币,使得面值之和恰好为 S ?

输出硬币数目的最小值

输入格式

第一行两个整数, n , $S(1 \leq n \leq 100, 0 \leq S \leq 100000)$
第二行 n 个整数 $v_i(1 \leq v_i \leq S)$

输出格式

一个整数,表示硬币数目的最小值 a ,无解则输出 -1

输入样例

```
6 666
1 5 10 20 50 100
```

输出样例

```
10
```

对于当前面额S,可以考虑n种选择方式变为$S - v_i$

直接进行搜索时间复杂度为指数级

考虑贪心

对于当前面额S,尽可能选取不超过x的v_i变为$S - v_i$

记$f(S)$为面额为S的硬币最优值

$$f(S) = 1 + f\left(S - \max_{v_i \leq S}(v_i)\right)$$

对于该最优子结构,极容易反驳是错误的

如 $S = 11, v = \{1,5,7\}$

#1898、硬币问题

将凑满面额S这一问题作为**阶段**,S的具体值作为**状态**(特征值),将从所有$S-v_i$面额选取一枚硬币作为**决策**

把当前阶段的最优解记为$f(S)$

对于$f(S)$,存在若干子问题 $f(S-v_i), v_i \le S$,可以得出状态转移方程

$$f(S)=\begin{cases}\min\{f(S-v_i)\}+1, & S>0\\ 0, & S=0\end{cases}$$

该问题可拆分,并显然存在子结构

证明最优子结构

设 $f(X)$ 为面额为X时最优解, $f(X-v_j)$为所有子问题中的最优解

若存在 $f(X-v_k)<f(X-v_j) \Rightarrow f(X-v_k)+1=f(X)'<f(X)=f(X-v_j)+1$ 与假设条件$f(X)$是最优解矛盾,证毕

对于该题,硬币数量没有限制当前决策不影响后续决策,即具备**无后效性**

#1032、最大子段和

题目描述

给出一段序列,选出其中连续且非空的一段使得这段和最大

输入格式

第一行是一个正整数 N ,表示了序列的长度

第二行包含 N 个绝对值不大于 10000 的整数 A_i ,描述了这段序列

输出格式

一个整数,为最大的子段和是多少。子段的最小长度为 1

输入样例

```
7
2 -4 3 -1 2 -4 3
```

输出样例

```
4
```

样例说明

在序列 $2, -4, 3, -1, 2, -4,$ 中,最大的子段和为 4 ,该子段为 $3, -1, 2$

数据规模与约定

对于 40% 的数据,有 $N \leq 2000$

对于 100% 的数据,有 $N < 100000$

枚举子段起点和终点再求和,时间复杂度$O(n^3)$

前缀和优化求和,时间复杂度$O(n^2)$

#1032、最大子段和

设 $dp[i]$表示以 $a[i]$ 结尾的最大子段和

一共有两种决策

选择与$a[i-1]$结尾的子段组合,那必然是和前一个位置的最大子段合并,即 $dp[i-1]+a[i]$

不与$a[i-1]$结尾的子段组合,重新开始一段,即 $a[i]$

状态转移方程为

$$dp[i]=\max(dp[i-1]+a[i],a[i])$$

答案为

$$\max_{1\le i\le n}\{\,dp[i]\,\}$$

时间复杂度$O(n)$

#1032、最大子段和

求$a[l\ \sim r]$中的最大子段和,不妨将区间分成$a[l\ \sim mid]$和$a[mid+1\ \sim r]$两个部分

不难发现最大子段和的存在仅由三种情况

- 仅在$a[l\ \sim mid]$中,往左递归时会计算到
- 仅在$a[mid+1\ \sim r]$中,往右递归时会计算到
- 横跨左右两个部分

  可以求出从 mid 往 l 延伸的最大连续和,再求出从 $mid+1$ 往 r 延伸的最大连续和,最后合并两个最大连续和即为横跨左右两个部分的最大字段和。即设 $L[i]=L[i+1]+a[i](1\le i\le mid),\ R[i]=R[i-1]+a[i](mid<i\le n)$,当求解横跨左右的询问 $[l\ \sim r]$ 时: $sumL=\max\limits_{i=1}^{mid}\{L[i]\},sumR=\max\limits_{i=mid+1}^{r}\{R[i]\}ans_{l,r}=\max\big(\max(sumL,0)+sumR,sumL+\max(sumR+0)\big)$

  $sumL$ 和 $sumR$ 都可以通过 L 和 R 向左向右取max 得到

注意统计答案时**左右可以不取,但是两侧至少取一个数字**,递归处理,对三种情况取最大值即可,时间复杂度$O(n\log n)$

#1283、最长公共子序列

题目描述

一个给定序列的子序列是在该序列中删去若干元素后得到的序列

确切地说,若给定序列 $X = x_1, x_2, \ldots, x_m$,则另一序列 $Z = z_1, z_2, \ldots, z_k$ 是 X 的子序列是指存在一个严格递增的下标序列 $i_1, i_2, \ldots, i_k$,使得对于所有 $j = 1, 2, \ldots, k$ 有:

$$X_{i_j} = Z_j$$

例如,序列 $Z =$ `BCDB` 是序列 $X =$ `ABCBDAB` 的子序列,相应的递增下标序列为 $2, 3, 5, 7$

给定两个序列 X 和 Y ,当另一序列 Z 既是 X 的子序列又是 Y 的子序列时,称 Z 是序列 X 和 Y 的公共子序列

例如,若 $X = A, B, C, B, D, A, B$ 和 $Y = B, D, C, A, B, A$,则序列 B, C, A 是 X 和 Y 的一个公共子序列,序列 B, C, B, A 也是 X 和 Y 的一个公共子序列,而且后者是 X 和 Y 的一个最长公共子序列,因为 X 和 Y 没有长度大于 4 的公共子序列

给定两个序列 $X = x_1, x_2, \ldots, x_m$ 和 $Y = y_1, y_2, \ldots, y_n$

要求找出 X 和 Y 的一个最长公共子序列

输入格式

共有两行

每行为一个由大写字母构成的长度不超过 1000 的字符串,表示序列 X 和 Y

输出格式

第一行为一个非负整数。表示所求得的最长公共子序列的长度。若不存在公共子序列．则输出文件仅有一行输出一个整数 0

#1283、最长公共子序列

设$dp[i][j]$表示考虑到X前i个字符,Y前j个字符的最长公共子序列长度

若$X[i] = Y[j]$,则可将它接在公共子序列的结尾,即$dp[i-1][j-1]+1$

若$X[i] \neq Y[j]$,可以考虑舍弃$X[i]$,也可以舍弃$Y[j]$,取两者最大值

状态转移方程

$$dp[i][j] = \begin{cases} dp[i-1][j-1]+1, & X[i] = Y[j] \\ \max(dp[i-1][j], dp[i][j-1]), & X[i] \neq Y[j] \end{cases}$$

枚举所有的i, j,答案为$dp[|X|][|Y|]$

时间复杂度$O(|X| \cdot |Y|)$

可参考*LCS* 交互网页更好地理解实现过程

若X和Y是一个排列,是否有更高效的实现方式?如果没有特殊性质,是否有更高效的实现方式?

#1038、最长上升子序列

题目描述

设有整数序列 $b_1, b_2, b_3, \ldots, b_m$

若存在

$$i_1 < i_2 < i_3 < \ldots < i_n$$

且

$$b_{i_1} < b_{i_2} < \cdots < b_{i_n}$$

则称 $b_1, b_2, b_3, \ldots, b_m$ 中有长度为 n 的不下降序列

$$b_{i_1}, b_{i_2}, \ldots, b_{i_n}$$

求序列 $b_1, b_2, b_3, \ldots, b_m$ 中所有长度(n)最大不下降子序列

输入格式

第一行一个整数 M （$M \leq 10000$）
接下来输入 M 个用空格隔开的整数 (≤ 20000) 序列;

输出格式

一行一个整数，表示最大长度 n

#1038、最长上升子序列

设$dp[i]$表示以$a[i]$结尾的最长上升子序列长度

显然$dp[i]$初始值为1

尝试找到一个j满足$j < i$且$a[j] < a[i]$,那么$a[i]$可以接在$a[j]$之后,用$dp[j] + 1$更新$dp[i]$

状态转移方程

$$dp[i] = \max_{1 \le j < i \wedge a[j] < a[i]} \{ dp[j] + 1 \}$$

答案为

$$\max_{1 \le i \le n} \{ dp[i] \}$$

时间复杂度$O(n^2)$

若需要求最长不下降子序列仅需找到j满足$j < i$且$a[j] \le a[i]$转移即可

#1283、最长公共子序列

若$n = 10^5$,$O(n^2)$算法无法通过

对于一个相同长度的上升子序列,末尾元素越小越容易使得后续序列变长

设 len 为当前最长上升子序列的长度,$d_i(1 \le i \le len)$ 表示所有长度为 i 的不下降子序列里,末尾元素最小能是多少

初始时 $d_1 = a_1, len = 1$

依次放入 $a_2 \dots a_n$,每次加入 a_i 后都尝试维护好新的 len 和 d_i,显然始终满足 $d_i < d_{i+1}(1 \le i < len)$ 分情况如下:

- 若 $a_i > d_{len}$,说明 a_i 比现在最长的子序列的末尾元素还大,即 $d_{len+1} = a_i$ 且 $len \mathrel{+}= 1$
- 否则,找到**尽量靠右**的 k 满足 $d_k \ge a_i$。考察所有长度 $= k$ 的上升序列,本来它最小能以 d_k 收尾,但是现在可以把 d_k 换成 a_i 使得结束位置的数字更小,即 $d'_k = a_i$。但是对于 $j = 1 \dots k-1$ 或 $j = k+1 \dots len$, a_i 都不会影响这些 d_j

上述寻找操作可以用二分来完成,所以总时间复杂度为 $O(n \log n)$

也可以用线段树/树状数组优化求区间(前缀)最值,同样做到 $O(n \log n)$ 求最长上升序列的长度

#1448、最大上升子序列和

题目描述

一个数的序列 b_i ,当 $b_1 < b_2 < \ldots < b_S$ 的时候,我们称这个序列是上升的

对于给定的一个序列 $(a_1, a_2, \ldots, a_N)$,我们可以得到一些上升的子序列

$$(a_{i1}, a_{i2}, \ldots, a_{iK})$$

其中 $1 \leq i_1 < i_2 < \ldots < i_K \leq N$

比如,对于序列 $(1, 7, 3, 5, 9, 4, 8)$,有它的一些上升子序列,如 $(1, 7), (3, 4, 8)$ 等等。这些子序列中和最大为 18 ,为子序列 $(1, 3, 5, 9)$ 的和

你的任务,就是对于给定的序列,求出最大上升子序列和

注意,最长的上升子序列的和不一定是最大的,比如序列 $(100, 1, 2, 3)$ 的最大上升子序列和为 100 ,而最长上升子序列为 $(1, 2, 3)$

输入格式

输入的第一行是序列的长度 $N(1 \leq N \leq 1000)$

第二行给出序列中的 N 个整数,这些整数的取值范围都在 0 到 10000 (可能重复)

输入格式

最大上升子序列和

输入样例

```
7
1 7 3 5 9 4 8
```

输出样例

```
18
```

设$dp[i]$表示以$a[i]$结尾的最长上升子序列和

初始值$dp[i] = a[i]$

尝试找到一个j满足$j < i$且$a[j] < a[i]$

那么$a[i]$可以接在$a[j]$之后,用$dp[j] + a[i]$更新$dp[i]$

状态转移方程

$$dp[i] = \max_{1 \leq j < i \wedge a[j] < a[i]} \{ dp[j] + a[i] \}$$

答案为

$$\max_{1 \leq i \leq n} \{ dp[i] \}$$

时间复杂度$O(n^2)$

#1040、合唱队形

题目描述

N 位同学站成一排,音乐老师要请其中的($N-K$)位同学出列,使得剩下的 K 位同学排成合唱队形

合唱队形是指这样的一种队形:设 K 位同学从左到右依次编号为 $1,2,\ldots,K$,他们的身高分别为 $T_1,T_2,\ldots,T_K$,则他们的身高满足

$$T_1<T_2<\cdots T_i>T_{i+1}>\cdots>T_k(1\leq i\leq k)$$

你的任务是,已知所有 N 位同学的身高,计算最少需要几位同学出列,可以使得剩下的同学排成合唱队形

输入格式

输入的第一行是一个整数 N ,表示同学的总数

第二行有 n 个整数第 i 个整数 T_i 是第 i 位同学的身高(厘米)

输出格式

输出一个整数,最少需要几位同学出列

数据规模

对于 50% 的数据,保证有 $2\leq n\leq 20$

对于全部的数据,保证有 $2\leq n\leq 100,130\leq T_i\leq 230$

输入样例

```
8
186 186 150 200 160 130 197 220
```

输出样例

```
4
```

设$dp1[i]$表示以$T[i]$结尾的最长上升子序列长度

设$dp2[i]$表示以$T[i]$开头的最长下降子序列长度

对于$dp2[i]$可以逆序遍历数组求LIS

答案为

$$\max_{1\leq i\leq n}\{dp1[i]+dp2[i]-1\}$$

时间复杂度$O(n^2)$

#2254、购买股票

题目描述

如果你能知道接下来 n 天的每日股价,那您一定可以很快赚到十个亿然后捐给 SYC

给定一个长度为 n 数组 p ,它的第 i 个元素 p_i 表示一支给定股票第 i 天的价格

你只能选择 **某一天** 买入这只股票,并选择在 **未来的某一个不同的日子** 卖出该股票。输出你可以从这笔交易中获取的最大利润。如果你不能获取任何利润,输出 `0`

输入格式

第一行输入一个正整数 n

第二行输入 n 个使用空格分隔的正整数 p_i

输出格式

输出你可以从这笔交易中获取的最大利润。如果你不能获取任何利润,输出 `0`

数据规模

对于全部的数据 $1 \leq n \leq 10^5, 0 \leq p_i \leq 10^4$

输入样例1

```
6
7 1 5 3 6 4
```

输出样例1

```
5
```

样例解释1

解释：在第 2 天（股票价格 = 1 ）的时候买入,在第 5 天（股票价格 = 6 ）的时候卖出,最大利润 = $6-1=5$

注意利润不能是 $7-1=6$,因为卖出价格需要大于买入价格；同时,你不能在买入前卖出股票

#2254、购买股票

设 $dp[i][0]$ 表示考虑到第 i 天且未持有股票的最大收益, $dp[i][1]$表示考虑到第i天且持有股票的最大收益

显然 $dp[1][0] = 0, dp[1][1] = -p[1]$

若第i天未持有

- 第$i-1$天未持有,第i天依然观望,即$dp[i-1][0]$
- 第$i-1$天持有在第i天卖出赚得$p[i]$收益,即$dp[i-1][1]+p[i]$

$$dp[i][0] = \max(dp[i-1][0], dp[i-1][1]+p[i])$$

若第i天持有

- 第$i-1$天持有第i天未卖出,即$dp[i-1][1]$
- 第$i-1$天未持有在第i天买入,即$-p[i]$

$$dp[i][1] = \max(dp[i-1][1], -p[i])$$

枚举所有的i,同时递推$dp[i][0/1]$,答案为$dp[n][0]$,时间复杂度$O(n)$

#2908、又是股票购买

题目描述

如果你能知道接下来 n 天的每日股价,那您一定可以很快赚到十个亿然后捐给 SYC

给定一个长度为 n 数组 p ,它的第 i 个元素 p_i 表示一支给定股票第 i 天的价格

设计一个算法来计算你所能获取的最大利润,你可以尽可能地完成更多的交易(多次买卖一支股票)

注意：你不能同时参与多笔交易(你必须在再次购买前出售掉之前的股票)

输出你可以从这笔交易中获取的最大利润。如果你不能获取任何利润,输出 `0`

输入格式

第一行输入一个正整数 n

第二行输入 n 个使用空格分隔的正整数 p_i

输出格式

输出你可以从这笔交易中获取的最大利润。如果你不能获取任何利润,输出 `0`

数据规模

对于全部的数据 $1 \leq n \leq 10^5, 0 \leq p_i \leq 10^6$

输入样例1

```
6
7 1 5 3 6 4
```

输出样例1

```
7
```

样例解释1

在第 2 天(股票价格 = 1)时买入,在第 3 天(股票价格 = 5)的时候卖出, 这笔交易所能获得利润 = $5-1=4$
随后,在第 4 天(股票价格 = 3)的时候买入,在第 5 天(股票价格 = 6)的时候卖出, 这笔交易所能获得利润 = $6-3=3$
共得利润 $4+3=7$

#2908、又是股票购买

设$dp[i][0]$表示考虑到第i天且未持有股票的最大收益, $dp[i][1]$表示考虑到第i天且持有股票的最大收益

显然 $dp[1][0] = 0, dp[1][1] = -p[1]$

若第i天未持有

- 第$i-1$天未持有,第i天依然观望,即$dp[i-1][0]$
- 第$i-1$天持有在第i天卖出赚得$p[i]$收益,即$dp[i-1][1] + p[i]$

$$dp[i][0] = \max(dp[i-1][0], dp[i-1][1] + p[i])$$

若第i天持有

- 第$i-1$天持有第i天未卖出,即$dp[i-1][1]$
- 第$i-1$天未持有在第i天买入,即$dp[i-1][0] - p[i]$

$$dp[i][1] = \max(dp[i-1][1], dp[i-1][0] - p[i])$$

枚举所有的i,同时递推$dp[i][0/1]$,答案为$dp[n][0]$,时间复杂度$O(n)$

#2256、还是股票购买

题目描述

如果你能知道接下来 n 天的每日股价,那您一定可以很快赚到十个亿然后捐给 SYC 。

给定一个长度为 n 数组 p ,它的第 i 个元素 p_i 表示一支给定股票第 i 天的价格

你可以无限次地完成交易,但是你每笔交易都需要付手续费 f 。如果你已经购买了一个股票,在卖出它之前你就不能再继续购买股票了

> *注意：这里的一笔交易指买入持有并卖出股票的整个过程,每笔交易你只需要为支付一次手续费*

输出你可以从这笔交易中获取的最大利润。如果你不能获取任何利润,输出 `0`

输入格式

第一行输入一个正整数 n, f

第二行输入 n 个使用空格分隔的正整数 p_i

输出格式

输出你可以从这笔交易中获取的最大利润。如果你不能获取任何利润,输出 `0`

数据规模

对于全部的数据 $1 \leq n \leq 10^5, 0 \leq p_i \leq 10^6$

输入样例1

```
6 2
1 3 2 8 4 9
```

输出样例1

```
8
```

样例解释

在此处买入 $p_1 = 1$
在此处卖出 $p_4 = 8$
在此处买入 $p_5 = 4$
在此处卖出 $p_6 = 9$
总利润: $((8 - 1) - 2) + ((9 - 4) - 2) = 8$

#2256、还是股票购买

设$dp[i][0]$表示考虑到第i天且未持有股票的最大收益, $dp[i][1]$表示考虑到第i天且持有股票的最大收益

显然 $dp[1][0] = 0, dp[1][1] = -p[1]$

若第i天未持有

- 第$i-1$天未持有,第i天依然观望,即$dp[i-1][0]$
- 第$i-1$天持有在第i天卖出赚得$p[i]-f$收益,即$dp[i-1][1]+p[i]-f$

$$dp[i][0] = \max(dp[i-1][0], dp[i-1][1]+p[i]-f)$$

若第i天持有

- 第$i-1$天持有第i天未卖出,即$dp[i-1][1]$
- 第$i-1$天未持有在第i天买入,即$dp[i-1][1]-p[i]$

$$dp[i][1] = \max(dp[i-1][1], dp[i-1][1]-p[i])$$

枚举所有的i,同时递推$dp[i][0/1]$,答案为$dp[n][0]$,时间复杂度$O(n)$

谢谢观看