FZOI

信息学

数据结构优化DP

简单问题

- 给一个序列$\{a_i\}$求一个子段，使其和最大，子段长度不小于L

简单问题

- 给一个序列$\{a_i\}$求一个子段，使其和最大，子段长度不小于L
- 令 $sum_i = \sum_{k=1}^{i} a_k$

$$ans = \max_{L\le i\le n}\left\{sum_i - \min_{0\le j\le i-L}\{sum_j\}\right\}$$

- 你们会怎么做?
- 发现随着i增加，每次都会增加一个待选状态集合。
- 我们只需要用一个变量，记录所有待选状态的最小值，即可实现$O(1)$的转移

之前问题总结

- 我们的决策待选集合只扩大，不缩小；
- 那么是可以用一个变量维护最值的
- （不断的与新加入集合的元素比较即可）

- 但更多的问题，我们会需要的操作更加复杂，此时就需要使用高级数据结构进行维护

POJ3171 Cleaning Shifts

- 给N个区间(a_i, b_i)及其代价值c_i，问要覆盖$[L, R]$区间至少要花费多少代价；
- $1 \le N \le 25000, 1 \le L \le R \le 10^6$

POJ3171 Cleaning Shifts

西南大学附属中学 High School Affiliated to Southwest University

- 状态：
- $f[x]$表示覆盖$[L, x]$需要的最小代价；
- 显然我们会将所有区间按照b_i进行排序，设前区间$[a_i, b_i]$代价为c_i，
- 状态转移方程:

$$f[b_i] = \min_{a_i-1 \le x < b_i} \{f[x]\} + c_i$$

- 初始值：
- $f[L-1] = 0$，其余为正无穷；
- 目标值：

$$\min_{b_i \ge R}\{f[b_i]\}$$

POJ3171 Cleaning Shifts

$$f[b_i] = \min_{a_i - 1 \le x < b_i} \{f[x]\} + c_i$$

- 暴力做：
- 对于每个区间，暴力扫$[a_i - 1, b_i \;\; - 1]$，时间复杂度：$O(n^2)$
- 思考下每次决策会要做的事情：
- 1、在$[a_i - 1, b_i - 1]$中找到最小值；
- 2、更新b_i的值；
- **线段树！！**

POJ3171 Cleaning Shifts

$$f[b_i] = \min_{a_i-1\le x<b_i}\{f[x]\} + c_i$$

- 数据结构优化：
- 使用线段树维护f的值，每次决策的时候：
- 1、在$[a_i-1, b_i-1]$中找到最小值；
- 2、更新b_i的值
- 总时间复杂度：
- $O(nlogn)$

HDU 5542 The Battle of Chibi

- 给出长度为N的序列$\{A_i\}$，问这个序列中有多少个长度为m的严格单调递增子序列。
- $1 \le M \le N \le 1000, |A_i| \le 10^9$
- 最后输出答案对10^9+7取模即可

HDU 5542 The Battle of Chibi

- 给出长度为N的序列$\{A_i\}$，问这个序列中有多少个长度为m的严格单调递增子序列。
- 状态：
- $F[i,j]$表示$\{A_i\}$中的前j个数构成的以A_j结尾的数列中，长度为i的严格递增子序列个数
- 状态转移方程：

$$F[i,j] = \sum_{k<j \ \&\& \ A_k<A_j} F[i-1,k]$$

HDU 5542 The Battle of Chibi

$$F[i,j] = \sum_{k<j\ \&\&\ A_k<A_j} F[i-1,k]$$

- 如果尝试写下代码：

```
a[0] = -inf
f[0][0] = 1

for (int i = 1; i < m; i++){
    for (int j = 1; j <= n; j++){
        for (int k = 0; k < j; k++){
            if (a[k] < a[i])
                f[i][j] = (f[i][j] + f[i-1][k]) % MOD;
        }
    }
}
```

标准$O(n^3)$代码

因此会TLE

HDU 5542 The Battle of Chibi


```
a[0] = -inf
f[0][0] = 1

for (int i = 1; i < m; i++){
    for (int j = 1; j <= n; j++){
        for (int k = 0; k < j; k++){
            if (a[k] < a[i])
                f[i][j] = (f[i][j] + f[i-1][k]) % MOD;
        }
    }
}
```

- 如果我们只看最内两层循环，我们可以将i当作定值。
- 发现当j增加1时，k的取值范围从$[0, j)$变为$[0, j+1)$，只增加了一个可决策状态

HDU 5542 The Battle of Chibi

$$F[i,j]=\sum_{k<j\ \&\&\ A_k<A_j} F[i-1,k]$$

- 如果我们只看最内两层循环，我们可以将i当作定值。
- 发现当j增加1时，k的取值范围从$[0,j)$变为$[0,j+1)$，只增加了一个可决策状态。
- 那么我们需要一个数据结构：
- 1、每次能动态插入一个元素；
- 2、快速的查询前面插入的所有元素中小于A_j的$F[i-1,k]$的和；
- **树状数组！！**

HDU 5542 The Battle of Chibi

西南大学附属中学 High School Affiliated to Southwest University

$$F[i,j] = \sum_{k<j\ \&\&\ A_k<A_j} F[i-1,k]$$

- 每个决策用二元组(A_k , $F[i-1, k]$)来存储
- j扩大1前，将(A_j , $F[i-1, j]$)加入集合；
- 给定A_j，在树状数组中查询满足$A_k < A_j$的所有的$F[i-1, k]$的和
- 树状数组中：A_k为key值，$F[i-1, k]$为权值，每次查询的时候，即为求$1\sim A_k-1$的前缀和
- 注意：由于A_k会比较大，所以应该提前离散化一下；

总结一下

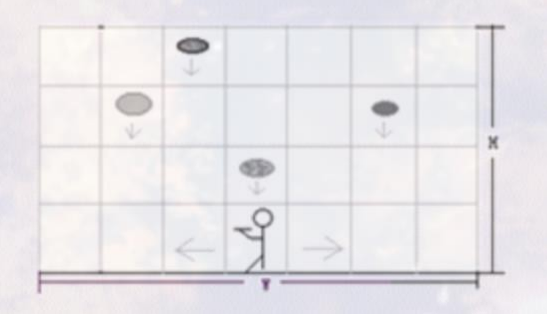
- 求解的过程：
- 1、先像正常的DP问题一样，找状态，找状态转移方程；
- 2、分析每次决策，要对待选集合所做的操作；
- 3、寻找合适的数据结构去维护待选集合；

总结一下

- DP的决策常常是找最大，最小,求和；
- 但是在决策的时候会有诸多限制；
- 我们要做的是：将限制分离出来（分别为一个维度）
- 排序可以解决一个维度
- 在用数据结构维护一个维度，就能胜任很多问题了
- 再多个维度怎么办？
- 分治或者二维数据结构嘛

[BZOJ2131]免费的馅饼

- 题意一个竖直的二维平面里，有宝物不断地从上面掉下来。第i个宝物下降到平面底部的时间为ti，位置为pi，宝物价值为vi。你作为一个玩家，要在这个竖直平面的底部接住足够的宝物来获取最高的价值。你可以在平面底部移动，并且你的速度最大为：2单位/单位时间。初始时间你可以在平面底部的任意一个位置。现在给定上述所有信息，需要求出可获得的最大价值。
- 第一行输入平面底部宽度W（≤1e8）和宝物个数N（≤1e5） 接下来N行每行三个整数ti,pi,vi，输出最大价值



[BZOJ2131]免费的馅饼

- 状态：
- f_i表示考虑前i个物品，且第i个物品要接，能得到的最优解
- 状态转移方程：

$$f_i = \max\{f_j\} + v_i, \ |p_i - p_j| \le 2 * (t_i - t_j)$$

- 暴力做的时间复杂度：
- $O(n^2)$

[BZOJ2131]免费的馅饼

$$f_i = \max\{f_j\} + v_i, \ |p_i - p_j| \le 2 * (t_i - t_j)$$

- 如何优化?
- 如果j的条件是个区间那么就是前面的题目
- 把条件展开:

$$\begin{cases} 2t_j - p_j \le 2t_i - p_i \\ 2t_j + p_j \le 2t_i + p_i \end{cases}$$

- j需要同时满足以上两个条件;
- 可以发现，每个物品相当于有两个值。
- j的两个值都小于i时就可以转移

[BZOJ2131]免费的馅饼

$$f_i = \max\{f_j\} + v_i,$$
$$\begin{cases} 2t_j - p_j \le 2t_i - p_i \\ 2t_j + p_j \le 2t_i + p_i \end{cases}$$

- 可以发现，每个物品相当于有两个值。j的两个值都小于i时就可 以转移;
- 其中一个值，排好序，按顺序进行遍历，另外一个值我们拿线段树（树状数组）进行维护统计即可；
- 总时间复杂度：
- $O(nlogn)$
- 题解来源：

  https://blog.csdn.net/izumi_hanako/article/details/78276844