动态规划优化

Dynamic Programming

方泓杰

目录
CONTENTS

PART 0

热身

Hdu 6076 Security check 弱化版

有两个长度为n的队列过安检，每个人有一个特征值。如果两个队列中的目前的第一个人的特征值之差小于等于k，那么一次只能检查其中一个人，否则一次可以检查两个人。每次检查花费1的时间。问最后检查完所有人之后所需要的最少时间。

数据范围$1 \leq n \leq 6000$.
TL：1s，ML：256Mb

Hdu 6076 Security check 弱化版

设$f_{i,j}$表示第一列到第i个人，第二列到第j个人的最短时间。

$$f_{i,j}=\begin{cases}\min\{f_{i-1,j},f_{i,j-1}\} & (|a_i-b_j|\le k)\\ f_{i-1,j-1}+1 & (|a_i-b_j|>k)\end{cases}$$

时间复杂度$O(n^2)$

Hdu 5707 combine string

给你三个字符串S1，S2和S3；判断S3是否恰好由S1、S2组成，S1为S3的子序列，S2也为S3的子序列，不能重叠。

数据范围$1 \leq |S_1, S_2, S_3| \leq 5000$
TL：1s，ML：256Mb

Hdu 5707 combine string

设$f_{i,j}$表示S1串到了第i个位置，S2串到了第j个位置，S3串到了第i+j个位置是否可行。
每次判断是否可以从$f_{i-1,j}$和$f_{i,j-1}$转移而来即可。

PART 01
一些基本优化

前缀和优化

利用前缀和可以实现$O(n)$预处理后$O(1)$查询一段的和。

二维前缀和……？一模一样。

ST表优化 BIT/线段树优化

可能有些人有听过，st表利用倍增的思想，支持$O(nlogn)$预处理，$O(1)$查询最值。写起来有点麻烦。

如果不是特别卡，建议用线段树，
毕竟好写，也就多个log吧?

树状数组/线段树 带修改区间求和/最值等等

```
namespace BIT {
    const int M = 6e5 + 10;
    int c[M], n;
    # define lb(x) (x&(-x))
    inline void init(int _n) {
        n = _n;
        memset(c, 0, sizeof c);
    }
    inline void edt(int x, int d) {
        for (; x<=n; x+=lb(x)) c[x] += d;
    }
    inline int sum(int x) {
        int ret = 0;
        for (; x; x-=lb(x)) ret += c[x];
        return ret;
    }
    inline int sum(int x, int y) {
        return sum(y) - sum(x-1);
    }
    # undef lb
}
```

优化状态：二取方格数

• 有一个 N×N 的方格图，在某些方格中填入正整数，其它方格中填入数字 0。
• 某人从 A(1,1) 出发，可以向下、向右行走，到达 B(N,N)。在走过的路上，他可以取走方格中的数，取走后的方格变为 0。
• 此人从 A 点到 B 点共走了两次，试找出两条这样的路径， 使得取得的数字和为最大。

数据范围$1 \leq n \leq 100$
TL：1s，ML：256Mb

二取方格数

$f_{a,b,c,d}$表示第一个人走到了(a,b)，第二个人走到了(c,d)的方案数。

优化？观察相同点。
两个人走的总步数相同。
$f_{i,a,b}$表示两个人都走了i步，其中第一个人向右走了a步，第二个人向右走了b步的方案数。可以由i，a，b推出坐标表示。
状态$O(n^4) \rightarrow O(n^3)$

Noip2010 乌龟棋

乌龟棋的棋盘是一行 N 个格子，每个格子上一个分数（非负整数）。游戏要求玩家控制一个乌龟棋子，从起点第 1 格 出发走到终点第 N 格。

乌龟棋中有 M 张卡片，卡片有四种花色，分别对应 1,2,3,4 四个数字。每次使用一张卡片，棋子就可以向前移动这张卡片所对应数字的格数。比如用一张第三种花色的卡片，乌龟棋就向前移动三格。每张卡片在一次游戏中只能使用一次。玩家在本次游戏中的得分，就是移动乌龟棋从第一格到最后 一格的过程中，经过的所有的格子上的分值的和。

很明显，用不同的卡片使用顺序会使得最终游戏的得分不同，你的任务是要找到一种卡片使用顺序使得最终游戏得分最多。

数据保证到达终点时刚好用光 M 张爬行卡片。

数据范围$1 \leq N \leq 350, 1 \leq M \leq 120$, 每种卡片张数 ≤ 40

TL：1s，ML：256Mb

Noip2010 乌龟棋

$f_{i,a,b,c,d}$表示玩到第i格，用了a/b/c/d张1/2/3/4的最大得分。
从$f_{i-1,a-1,b,c,d}, f_{i-2,a,b-1,c,d}, f_{i-3,a,b,c-1,d}, f_{i-4,a,b,c,d-1}$推得。

观察到$i = a + 2b + 3c + 4d + 1$，第一维无需枚举。
状态$O(n^5) \rightarrow O(n^4)$

数据结构优化：和谐序列

给定一个 n 个元素的序列a_i，定义“和谐序列”为序列的任何两个相邻元素相差不超过K，求a_i的子序列中“和谐序列”的个数。

数据范围$1 \leq n \leq 10^5$
TL：1s，ML：256Mb

和谐序列

f_i表示以第i个数为结尾的和谐序列的个数。

$$f_i = 1 + \sum_{j=1}^{i-1} f_j[|a_j - a_i| \le K]$$

拆开绝对值，即$a_i - K \le a_j \le a_i + K$
那么每次更新完f_i，将f_i插入到BIT上a_i对应位置。
查询的时候只需要查找区间和即可。

如果a_i很大，需要排序离散，并在查询时候二分。

Bzoj2259 新型计算机

新型计算机的输入很独特 假设输入序列中有一些数字 (都是自然数，包括 0) 计算机先读取第一个数字s_1，然后顺序向后读入s_1个数字； 接着再读一个数字s_2，顺序向后读入s_2个数字……依此类推 不过只有计算机正好将输入序列中的数字读完，它才能正确处理数据，否则计算机就会进行自毁性操作!

Tim 现在有一串输入序列，但可能不是合法的，也就是可能会对计算机造成破坏。 于是他想对序列中的每一个数字做一些更改，加上一个数或者减 去一个数，当然，仍然保持其为原来的数。 使得更改后的序列为一个新型计算机可以接受的合法序列。 不过 Tim 还希望更改的总代价最小，所谓总代价，就是对序列 中每一个数改变的绝对值之和。

数据范围$1 \leq n \leq 10^6$

TL：1s，ML：256Mb

Bzoj2259 新型计算机

从前往后不好做，考虑从后往前。

f_i表示$i \to n$合法的最小代价。

$$f_i = \min\{f_j + |a_i - (j - i - 1)|\}$$

当$a_i > j - i - 1$，也就是$a_i + i + 1 > j$，那么有

$$f_i = \min\{f_j - j\} + a_i + i + 1$$

当$a_i < j - i - 1$，也就是$a_i + i + 1 < j$，那么有

$$f_i = \min\{f_j + j\} - (a_i + i + 1)$$

那么以j为下标，用BIT/线段树维护$f_j - j$和$f_j + j$最小值即可。

每次分情况查找。

该题还有其他做法，在这里不一一叙述。

Bzoj3688 折线统计

二维平面上有 n 个点 (x_i, y_i)，现在这些点中取若干点构成一个集合 S，对它们按照 x 坐标排序，顺次连接，将会构成一些连续上升、下降的折线，设其数量为 $f(S)$。现给定 k，求满足 $f(S) = k$ 的 S 集合个数。

数据范围$1 \leq n \leq 50000, k \leq 10, 1 \leq xi, yi \leq 10^5$
TL：1s，ML：256Mb

Bzoj3688 折线统计

令$f_{i,j,0/1}$表示到第i个数，选了j段，当前上升/下降。

$$f_{i,j,0} = \sum_{k=1}^{i-1} f_{k,j,0} + f_{k,j-1,1}(if\ Y_k > Y_i)$$

$$f_{i,j,1} = \sum_{k=1}^{i-1} f_{k,j,1} + f_{k,j-1,0}(if\ Y_k < Y_i)$$

开$2k$棵BIT每次求出f的时候在对应位置插入即可。
然后每次查询求和。复杂度$O(nk\log n)$

PART 02

矩阵乘法

矩阵乘法

矩阵$A = n * m$，矩阵$B = m * k$，那么A与B才可以进行矩阵乘法。
得到的矩阵$C = n * k$。

$$C_{i,j} = \sum A_{i,k} \times B_{k,j}$$

矩阵乘法实现是$O(n^3)$的。
矩阵乘法不满足交换律，满足结合律。
所以可以进行快速幂，从而快速得到答案。

斐波那契数列

$f_i = f_{i-1} + f_{i-2}, f_1 = 1, f_2 = 1$，求$f_n$。取模。
数据范围$1 \le n \le 10^{18}$。

构造矩阵$\begin{bmatrix} f_i \\ f_{i-1} \end{bmatrix}$，考虑如何转移到$\begin{bmatrix} f_{i+1} \\ f_i \end{bmatrix}$

根据矩阵乘法定义：$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} f_i \\ f_{i-1} \end{bmatrix} = \begin{bmatrix} f_{i+1} \\ f_i \end{bmatrix}$

所以$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$就是转移矩阵，对转移矩阵进行矩阵乘法快速幂，最后乘初始矩阵即可。复杂度$O(\log n)$

练习

$f_i = 2f_{i-1} + 3f_{i-2} - f_{i-3}, f_1 = 1, f_2 = 1, f_3 = 2$，求$f_n$。取模。
数据范围$1 \leq n \leq 10^{18}$。

Bzoj4870 组合数问题

$1 \leq n \leq 10^9,$
$0 \leq r < k \leq 50,$
$2 \leq p \leq 2\text{^}30 - 1$

TL：1s，ML：512Mb

组合数 C_n^m 表示的是从 n 个互不相同的物品中选出 m 个物品的方案数。举个例子，从 $(1,2,3)$ 三个物品中选择两个物品可以有 $(1,2),(1,3),(2,3)$ 这三种选择方法。根据组合数的定义，我们可以给出计算组合数 C_n^m 的一般公式：

$$C_n^m = \frac{n!}{m!(n-m)!}$$

其中 $n! = 1 \times 2 \times \cdots \times n$。（特别的，当 $n = 0$ 时，$n! = 1$，当 $m > n$ 时，$C_n^m = 0$）

小葱在 NOIP 的时候学习了 C_i^j 和 k 的倍数关系，现在他想更进一步，研究更多关于组合数的性质。小葱发现，C_i^j 是否是 k 的倍数，取决于 $C_i^j \bmod k$ 是否等于 0，这个神奇的性质引发了小葱对 mod 运算（取余数）的兴趣。现在小葱选择了是四个整数 n, p, k, r，小葱现在希望知道

$$\left(\sum_{i=0}^{\infty} C_{nk}^{ik+r}\right) \bmod p$$

即

$$\left(C_{nk}^{r} + C_{nk}^{k+r} + C_{nk}^{2k+r} + \cdots + C_{nk}^{(n-1)k+r} + C_{nk}^{nk+r} + \cdots\right) \bmod p$$

的值。

Bzoj4870 组合数问题

题目可以转化为从nk个物品中选出模k余r个物品的方案数。
$f_{i,j}$表示前i个物品 选出(模k余j个)物品的方案数。
那么每次转移的时候都是把当前j这一维度向后移动1位。
转移方式相同，即可使用矩阵乘法优化。
时间复杂度$O(k^3 \log n)$

Bzoj2510 弱题 弱化版

有m个球，一开始每个球均有一个初始标号，标号范围为$1\sim n$且为整数，标号为i的球有a_i个，并保证$\sum a_i = m$。每次操作等概率取出一个球（即取出每个球的概率均为$\frac{1}{m}$），若这个球标号为$p(p < n)$，则将它重新标号为$p+1$；若这个球标号为n，则将其重标号为1。（取出球后并不将其丢弃）现在你需要求出，经过K次这样的操作后，每个标号的球的期望个数。

数据范围$1 \le n \le 100, 1 \le m \le 10^8, 1 \le k \le 2^{31}-1$
TL：1s，ML：256Mb

Bzoj2510 弱题

考虑令$f_{t,i}$表示t轮过后，标号i的期望个数。

$$f_{t+1,i} = f_{t,i} - f_{t,i} \times \frac{1}{m} + f_{t,i-1} \times \frac{1}{m}$$

那么就可以构造矩阵了。以n=3为例。$\begin{bmatrix} f_{t,1} \\ f_{t,2} \\ f_{t,3} \end{bmatrix}$，需要达成的是$\begin{bmatrix} f_{t+1,1} \\ f_{t+1,2} \\ f_{t+1,3} \end{bmatrix}$，

于是有$\begin{bmatrix} 1-\frac{1}{m} & 0 & \frac{1}{m} \\ \frac{1}{m} & 1-\frac{1}{m} & 0 \\ 0 & \frac{1}{m} & 1-\frac{1}{m} \end{bmatrix} \begin{bmatrix} f_{t,1} \\ f_{t,2} \\ f_{t,3} \end{bmatrix} = \begin{bmatrix} f_{t+1,1} \\ f_{t+1,2} \\ f_{t+1,3} \end{bmatrix}$

矩阵乘法即可，复杂度$O(n^3 \log k)$。挑战：继续优化？

天使的分裂

$1 \leq n \leq 10^{18}$
TL：1s，ML：256Mb

赤目天使是天使发动分裂技能($harmonics$)的产物，保留了天使分裂时候的意识，因此充满攻击性。赤目天使进行着疯狂地分裂，企图消灭 SSS。

仲村由理要评估赤目天使对 SSS 的威胁。她发现由于某些原因，赤目天使的个数随天数的变化是一个斐波那契数列。准确来说，令发动技能那天为第0天，第i天的赤目天使个数设为f_i，那么f满足$f_0 = 1$，$f_1 = 1$，$f_i = f_{i-1} + f_{i-2}(i > 1)$。

由理定义了一个评估函数F_n，表示第n天的威胁估价。这个函数很奇怪，它是赤目天使个数的卷积，即：

$$F_n = \sum_{i=0}^{n} (f_i f_{n-i})$$

由理想要知道第0天到第n天期间所有评估函数的值的和，即$\sum_{i=0}^{n} F_i$，但是这个值可能很大，于是她要你将答案对$998244353(7 \times 17 \times 2^{23} + 1)$取模。

天使的分裂

观察后可以发现$F_n = F_{n-1} + F_{n-2} + f_n$
考虑构造矩阵$[F_n \quad F_{n-1} \quad f_{n+1} \quad f_n \quad sum]$，自行试着推导转移矩阵。

时间复杂度$O(5^3 \log n)$

PART 03

斜率优化、四边形不等式

从一道老题说起：BZOJ1597 土地购买

FJ准备扩大他的农场,他正在考虑 n 块长方形的土地. 每块土地有长宽。每块土地的价格是它的面积,但FJ可以同时购买多快土地. 这些土地的价格是它们最大的长乘以它们最大的宽, 但是土地的长宽不能交换. 如果FJ买一块3x5的地和一块5x3的地,则他需要付5x5=25. FJ希望买下所有的土地,但是他发现分组来买这些土地可以节省经费. 他需要你帮助他找到最小的经费.

数据范围$1 \leq n \leq 50000, 1 \leq W, L \leq 10^6$
TL：1s，ML：256Mb
Oj#474

BZOJ1597 土地购买

筛选出一定可以和其他打包买的土地并删去，剩下的一定是一些x递增，y递减的矩形。

令f_i表示前i块土地最小费用。

$$f_i = \min\{f_j + y_{j+1} \times x_i\}$$

如何优化?

Bzoj1597 土地购买

1. 证明决策单调
考虑状态i的时候，状态j比状态k优。证明对于i后所有状态p，状态j都比状态k优。$(j > k)$
· 简单来说，就是：现在比你厉害，将来一定比你厉害。
2. 求斜率方程
· 简单来说，就是：化简化简再化简，直到形成斜率形式。
3. 维护队列

BZOJ1010 玩具装箱

P教授有n件玩具，第i件玩具长度为C_i。他要把**所有玩具**运到北京。P教授要求在一维容器中的玩具编号是连续的。同时如果一个一维容器中有多个玩具，那么两件玩具之间要加入一个单位长度的填充物。也就是说，如果将第i个玩具到第j个玩具放到容器中，容器的长度将为

$$x = j - i + \sum_{k=i}^{j} C_k$$

容器长度为x,其制作费用为$(x - L)^2$。其中L是一个常量。求最小费用。

$1 \leq n \leq 50000, 1 \leq L, C_i \leq 10^7$

TL：0.1s，ML：162Mb
Oj#473

BZOJ1010 玩具装箱

显然有一个$\Theta(n^2)$的dp状态
设f_i表示选到第i个玩具之后的最小花费，显然有：

$$f_i = \min\{f_j + w_{i,j}\}$$

其中$w_{i,j}$表示把i到j当成一段的价值。

然后……？
斜率优化！

BZOJ1010 玩具装箱

1. 证明决策单调性

· 考虑状态i的时候，状态j比状态k优。证明对于i后所有状态p，状态j都比状态k优。$(j > k)$

为了描述方便，令$s_i = i + \sum_{i=1}^{i} c_i, t = 1 + l$

$$f_j + (s_i - s_j - t)^2 \leq f_k + (s_i - s_k - t)^2$$

状态i变为状态p，实际上只要将s_i加个常数v即可变为s_p，故即证：

$$f_j + (s_i + v - s_j - t)^2 \leq f_k + (s_i + v - s_k - t)^2$$

化简后只需证明$2v(s_i - s_j - t) \leq 2v(s_i - s_k - t)$

由于s非负，显然成立。

BZOJ1010 玩具装箱

2. 求斜率方程

$$f_j + (s_i - s_j - t)^2 \leq f_k + (s_i - s_k - t)^2$$

化简后有

$$\frac{(f_j + s_j^2) - (f_k + s_k^2)}{s_j - s_k} \leq 2(s_i - t)$$

看成平面上的点对$(s, f + s^2)$的斜率即可。
右边单调递增，所以维护一个决策队列（下凸壳）即可。
注意两边同时除的时候是否要变号！

BZOJ1010 玩具装箱

3. 维护队列
维护一个单调的决策队列，支持：
· 每次给出x（x单调），找出大于x的最小斜率所在决策点。
· 每次加入一个点

那么直接维护一个下凸壳即可。
问题解决，复杂度$\Theta(n)$

```cpp
long long gs(int i, int j) {
    long long x = j - i + p[j] - p[i-1];
    return 1ll * (x - l) * (x - l);
}

double slop(int j, int k) {
    return 1.0 * (F[j] - F[k]) / (s[j] - s[k]);
}

int main() {
    scanf("%d%d", &n, &l);
    for (int i=1; i<=n; ++i) scanf("%d", &c[i]), s[i] = s[i-1] + c[i];
    for (int i=1; i<=n; ++i) p[i] = s[i], s[i] += i;

    f[0] = 0; F[0] = 0; hd = 1, tl = 0; q[++tl] = 0;
    for (int i=1; i<=n; ++i) {
        while(hd < tl && slop(q[hd+1], q[hd]) <= 2.0 * (s[i] - l - 1)) ++hd;
        f[i] = f[q[hd]] + gs(q[hd]+1, i);
        F[i] = 1ll * s[i] * s[i] + f[i];
        while(hd < tl && slop(i, q[tl]) <= slop(q[tl], q[tl-1])) --tl;
        q[++tl] = i;
    }

    cout << f[n];
    return 0;
}
```

从一道更老的题说起：合并石子

有n堆石子，每堆有a_i个。按一定顺序合并，每次只能移动相邻两堆石子，合并费用为新一堆石子的数量。求合并成一堆最小花费。

$1 \leq n \leq 3000$
TL: 1s，ML: 512Mb

合并石子

显然有一个$\Theta(n^3)$区间dp的做法
令$f_{i,j}$表示$[i,j]$区间石子合并后的最小代价。
转移显然是枚举i和j之间的分界点k进行合并，分解成$f_{i,k}$和$f_{k+1,j}$。
于是就有转移方程

$$f_{i,j} = \min\{f_{i,k} + f_{k+1,j} + cost(i,j)\}$$

这就是典型的四边形不等式优化题目啦！
关于四边形不等式，由于证明较为复杂，在这里就不赘述了，大家只要懂得套用条件即可。

四边形不等式

$f_{i,j} = \min\{f_{i,k} + f_{k+1,j}\} + w(i,j)$ 或$f_{i,j} = \min\{f_{i-1,k} + w_{k+1,j}\}$

给出两个定义

①区间包含性：对于$a \le b < c \le d$，有$w(a,d) \ge w(b,c)$。

②四边形不等式：对于$a \le b < c \le d$，有$w(a,c) + w(b,d) \le w(a,d) + w(b,c)$。

再给出两个定理：

①如果上述的w函数同时满足区间包含单调性和四边形不等式性质，那么函数f也满足四边形不等式性质。

②我们再定义$s_{i,j}$表示$f_{i,j}$取得最优值时对应的下标（即$i \le k \le j$时，k处的f值最大，则$s_{i,j} = k$）。此时有假如$f_{i,j}$满足四边形不等式，那么$s_{i,j}$单调，也就是$s_{i,j} \le s_{i,j+1} \le s_{i+1,j+1}$。

四边形不等式

如果函数w满足上述性质，那么就可以用四边形不等式啦！
首先，根据定理2有$s_{i,j-1} \le s_{i,j} \le s_{i+1,j}$。
转移方程可改为$f_{i,j} = \min\{f_{i,k} + f_{k+1,j}\} + w(i,j)\ (s_{i,j-1} \le k \le s_{i+1,j})$
由于这个状态转移方程枚举的是区间长度L，而s(i,j-1)和s(i+1,j)的长度为L-1，是之间已经计算过的，可以直接调用。不仅如此，区间的长度最多有n个，对于固定的长度L，不同的状态也有n个，故时间复杂度为$\Theta(n^2)$
证明可以参考论文，此处省略。
Q: 那我怎么知道满足四边形不等式?
A: 打表验证下啊。

```
for (int i=1; i<=n; ++i) s[i][i] = i;
for (int len=2; len<=n; ++len) {
    for (int i=1, j, t, r; i+len-1<=n; ++i) {
        j = i+len-1; t = 2e9;
        for (int k=s[i][j-1]; k<=s[i+1][j]; ++k)
            if(f[i][k] + f[k+1][j] < t) {
                t = f[i][k] + f[k+1][j];
                r = k;
            }
        f[i][j] = t + gs[j] - gs[i-1];
        s[i][j] = r;
    }
}
```

FJOI2004 达尔文芯片问题

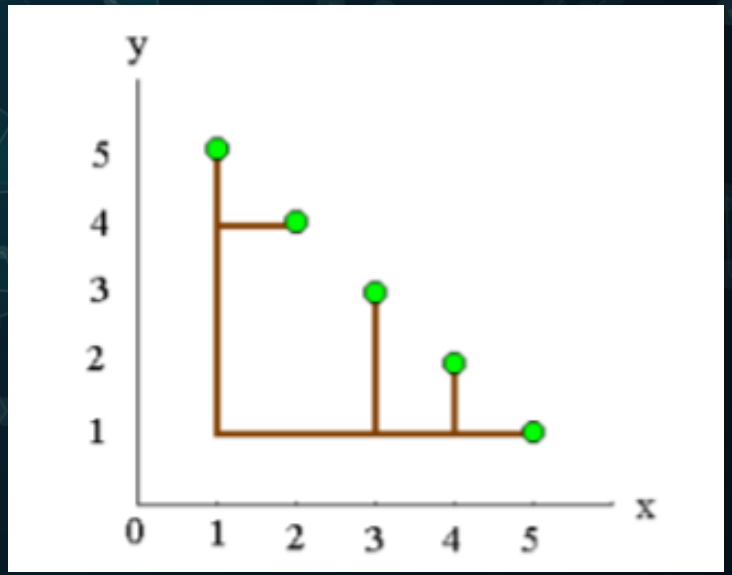
科学家们发现，将若干关键逻辑元按照电路板平面坐标系 2 维降序排列，经过演化，这些逻辑元自动按照 x,y 坐标方向联接成一棵树，树的每条边都平行于坐标轴。关键逻辑元构成这棵树的全 部叶结点。这类树称为正交树。

有趣的是，达尔文芯片自动产生的正交树的总边长是所有这种正交树中总边长最小的。例如，5 个关键逻辑元在电路板 xOy 坐标 系中的正交树如下图所示，它的总边长为 12。

求总边长最小的正交树的总边长值。

50%: $1 \leq n \leq 600$;

100% : $1 \leq n \leq 3000$



FJOI2004 达尔文芯片问题

$f_{i,j}$表示$[i,j]$全部连起来所用最小代价，那么有

$$f_{i,j}=\min\{f_{i,k}+f_{k+1,j}+y_k-y_j+x_{k+1}-x_i\}$$

然后呢？试试四边形不等式！

打个表看看后面那坨是否满足四边形不等式。

然后……它真的满足

那就直接做咯！

复杂度$O(n^2)$

决策单调性

这样一类问题：$f_i = \min\{f_j + w_{j,i}\}$
如果w函数满足四边形不等式，f就满足决策单调性。

如何判断满足决策单调性？多试几组数据，观察决策点变化。

有了决策单调性，设上一次决策点为k_{i-1}，容易想到从k_{i-1}枚举决策。
可惜这不是一个复杂度上的优化。

考虑换一种想法，不是找决策点，而是利用前面已经做出的决策更新后面。

决策单调性

决策1出来时，也就是f_1已经计算得到，那么很显然后面的所有决策都可以被他更新，所以这时候最优决策表是:

1111111111111111111111111111111111111111

我们继续算f_2，也计算得到了结果，更新决策表，由于决策单调性，只会出现:

11111111111111111112222222222222222222222

而不会有

11111111122222211111111111111111111111111

决策单调性

接下来有了决策3，也就是f_3，就有了两种情况

1111111111111111111122222222222333333333333

或者

1111111111111111111133333333333333333333333

所以我们就有一个想法：

考察决策2种的起始点是否没有决策3优，如果没有，就是下面那种情况，直接把决策2删除，增进来决策3即可。

否则，在决策2中二分决策3的起点。

决策单调性

所以，我们就可以用一个栈来维护决策序列。
每次遇到新决策时，扫描栈顶，如果新决策啥都比老决策优秀，那么就抛弃老决策；继续这个循环。否则，二分出起点，把新决策压入栈中。
由于每个决策只会一次进栈，所以均摊复杂度$O(n \log n)$

怎么还是这题啊：BZOJ1597 土地购买

FJ准备扩大他的农场,他正在考虑 n 块长方形的土地. 每块土地有长宽。每块土地的价格是它的面积,但FJ可以同时购买多快土地. 这些土地的价格是它们最大的长乘以它们最大的宽, 但是土地的长宽不能交换. 如果FJ买一块3x5的地和一块5x3的地,则他需要付5x5=25. FJ希望买下所有的土地,但是他发现分组来买这些土地可以节省经费. 他需要你帮助他找到最小的经费.

数据范围$1 \leq n \leq 50000, 1 \leq W, L \leq 10^6$
TL：1s，ML：256Mb
Oj#474

BZOJ1597 土地购买

如果不删去多余的土地的话，是不满足决策单调性的。
删去之后满足了决策单调性，就可以在$O(n\log n)$时间内完成啦！
虽然比斜率优化的$O(n)$慢，可以作为一个备用，当无法应用斜率优化，就可以尝试这个了！

Hdu3480 Division

将含有n个元素的一个集合分成M个子集，使得每个子集的最大值与最小值差的平方和最小。
数据范围$1 \leq n \leq 10000, 1 \leq m \leq 5000$
TL：5s，ML：256Mb

2种方法?

Hdu3480 Division

先排序，接下来肯定是选取连续的一段，这有点像？
玩具装箱！
考虑dp状态：
$f_{i,j}$表示前i个元素分为j个子集的最小值。
那么转移就枚举上一个分割的地方k。$f_{k,j-1} + (a_i - a_{k+1})^2$
为了方便将两维度调换顺序，
$f_{i,j}$表示前j个元素分为i个子集的最小值。
那么考虑对于某一个确定的i，j的转移就可以用斜率优化了。

Hdu3480 Division

1. 证明决策单调
考虑状态i的时候，状态j比状态k优。证明对于i后所有状态p，状态j都比状态k优。$(j > k)$
2. 求斜率方程
3. 维护队列

Hdu3480 Division

$$f_{i,j} = \min\{f_{k,j-1} + (a_i - a_{k+1})^2\}$$

非常像四边形不等式优化。
稍稍打表验证后发现满足四边形不等式。
为了方便将两维度调换顺序，
$f_{i,j}$表示前j个元素分为i个子集的最小值。
然后直接四边形不等式优化即可。

PART 04

动态规划相关复习 & 综合题

Codeforces 988F Rain and Umbrellas

一个长度为n的数轴，小A在原点，要走到$x = a$。
有些地方在下雨，必须要有伞才能过去。
在某些地方有伞，伞有重量p_i，小A可以选择在这些地方捡伞。
小A每走一步消耗能量是携带的所有伞总重量。
注意伞是可以携带多把的。
小A可以选择在任何地方将身上的任意把伞丢弃。
求小A到达终点所消耗最小体力值。
数据范围$1 \leq a, n \leq 2000$
TL：1s，ML：256Mb

Codeforces 988F Rain and Umbrellas

首先有一个结论：在每个地方最多只留一把伞。
由于每个地方可能有多把伞，预处理出每个地方最轻的伞w_i。
设f_i表示走到x=i的最小花费。
那么枚举上一次拿伞的地方j进行转移：

$$f_i = \min\{f_j + (i-j) \times w_j\}$$

时间复杂度$O(n^2)$

Bzoj1076 奖励关

有n种宝物，k关游戏，每关游戏随机给出一种宝物，可捡可不捡。每种宝物有一个价值（有负数）。每个宝物有前提宝物列表，必须在前面的关卡取得列表宝物才能捡起这个宝物，求期望收益。

数据范围$1 \leq k \leq 100, 1 \leq n \leq 15.$
TL：1s，ML：256Mb

Bzoj1076 奖励关

观察宝物数量很少，意味着我们可以将宝物拥有数量的状态进行压缩。
用15位的二进制数来表示宝物拥有状态，我们很容易设计出dp状态。
$f_{i,s}$表示前i关，已有宝物状态为S的期望收益。
这里我们采取倒推，因为倒推的时候我们知道所有状态都是有效的，不会出现无效变成有效的情况。

$$f_{i,s}=\frac{1}{n}\sum_{j=1}^{n}\max\left(f_{i+1,s+\{j\}}+p,f_{i+1,s}\right)$$

最后$f_{1,\emptyset}$就是答案。复杂度$\Theta(nk2^n)$

引号序列 弱化版

一个合法的引号序列是空串；如果引号序列合法，那么在两边加上同一个引号也合法；或是把两个合法的引号序列拼起来也是合法的。
求长度为2n，字符集大小为k的合法引号序列个数，对大质数取模。

注：这里是英文的引号，也就是不分左引号右引号，注意和括号的差别。

数据范围$1 \leq n \leq 1000$
TL：1s，ML：256Mb

引号序列 弱化版

设$f_{i,j}$表示到了第i个位置，前面还有j个左引号没有匹配的方案数。
每次有1种方案匹配前面的某一个引号，有$k-1$种方案开启新的引号。
也就是说$j \geq 1$时候，有$f_{i+1,j+1} \mathrel{+}= (k-1)f_{i,j}, f_{i+1,j-1} \mathrel{+}= f_{i,j}$。
特别的，当$j = 0$时候，有$f_{i+1,j+1} \mathrel{+}= kf_{i,j}$
复杂度$O(n^2)$

bzoj4760: Hoof, Paper, Scissors

现在FJ想要和他的最机智的奶牛Bessie玩剪刀石头布，一共进行了N轮Bessie，作为一个奶牛，非常的怠惰，无论她出什么，都喜欢连续的出，最多变化K次。现在FJ已经给出了他出的东西的序列，Bessie想知道她最多能赢多少次。

数据范围$1 \leq n \leq 10^5, 0 \leq k \leq 20$
TL：1s，ML：128Mb

bzoj4760: Hoof, Paper, Scissors

$f_{i,j,0/1/2}$ 表示前i个比赛，用了j次改变，最后一次出k的最多能赢次数。
枚举上一次出什么，进行转移即可。
用滚动数组压压空间。时间复杂度$O(nk)$

eighty-seven

有n张带有正整数的卡牌，m次询问，每次询问抽掉1~3张卡牌，问剩下的卡牌上的数字任意组合，能否组成87。T组数据。

数据范围$1 \le T \le 5, 1 \le n \le 50, 1 \le m \le 100000$
TL：1s，ML：256Mb

eighty-seven Analysis

预处理出抽出任意1~3张牌的答案即可。
考虑枚举三张牌后，就是一个bool型背包，用类似背包的方法即可。
时间复杂度$\Theta(87 \times Tn^4)$
有点可怕……
考虑优化？优化啥啊，bitset!
　　因为状态全为bool类型，可以把一个0/1变为一个数位，可以利用位运算进行操作，这样就使得复杂度除以了一个常数。
C++直接调用<bitset>头文件中的bitset即可。
简单来说，bitset就是一个可以做位运算的bool数组。

复杂度$\Theta(87 \times Tn^4 \div 32)$

Bzoj3687 简单题

求正整数集合所有子集算术和的异或和。
数据范围$1 \leq n \leq 200, 1 \leq \sum a_i \leq 2 \times 10^6$。
TL：1s，ML：512Mb

Bzoj3687 简单题 Analysis

令f_i表示子集和是否能为i。
那么每次加入一个数，就在所有的f_i为1的基础上，让f_{i+x}为1。
bitset!
复杂度$\Theta(n \times 2000000 \div 32)$。

Loj515 贪心只能过样例

一共有n个数，第i个数x_i可以取$[L_i, R_i]$之间任意正整数，设$S = \sum x_i^2$，求S种类数。

数据范围$1 \leq L_i, R_i \leq 100$。

TL：1s，ML：256Mb

Loj515 贪心只能过样例

发现最大的S不超过10^6， $f_{i,j}$表示到第i个数，S是否可以为j。
那么这个转移只要枚举即可，复杂度$O(n \times |R-L| \times 10^6)$
观察到是bool数组，用bitset转移即可，复杂度

$$O(n \times |R-L| \times 10^6 \div 32)$$

Codeforces 396B On Sum of Fractions

定义$a(n)$表示不比n大的最大的质数。
定义$b(n)$表示比n大的最小的质数。
求

$$\sum_{i=2}^{n} \frac{1}{a(i)b(i)}$$

数据范围$1 \leq n \leq 10^9$
TL：1s，ML：256Mb

Codeforces 396B On Sum of Fractions

考虑写出前几项：

$$\frac{1}{2\times 3}+\frac{1}{3\times 5}+\frac{1}{3\times 5}+\frac{1}{5\times 7}+\frac{1}{5\times 7}+\cdots$$

容易得到

$$\frac{3-2}{2\times 3}+\frac{5-3}{3\times 5}+\frac{7-5}{5\times 7}+\cdots$$

裂项相消：$\frac{1}{2}-\frac{1}{3}+\frac{1}{3}-\frac{1}{5}+\frac{1}{5}-\frac{1}{7}+\cdots$

只需要暴力求出n的u和v即可。

Codeforces 999F Cards and Joy

有nk张牌，每张牌上写有一个数字（这些数字可以相同），有n个人，每个人有一个最喜欢的数字。现在分发这些牌，每个人拿到k张牌，每个人拿到t张他们最喜欢的数字的卡片，获得的快乐值是h_t，求所有人获得的快乐值的和。保证$\{h_i\}$递增。

数据范围$1 \leq n \leq 500, 1 \leq k \leq 10$
TL：1s，ML：256Mb

Codeforces 999F Cards and Joy Analysis

我们可对每个数字分别处理，令$f_{x,y}$表示x个人（他们有相同的最喜欢的卡片），y张卡（均为他们最喜欢的卡片）最大可能的总快乐值。
那么我们枚举这个人分了多少卡片即可。

$$f_{x,y} = \max(f_{x,y}, f_{x-1,y-u})$$

最后对于每个数字分开统计求和即可。时间复杂度$O(n^2k^2)$

参考

Hzwer，n+e《动态规划优化》

Thanks