LAB
实验舱

动态规划

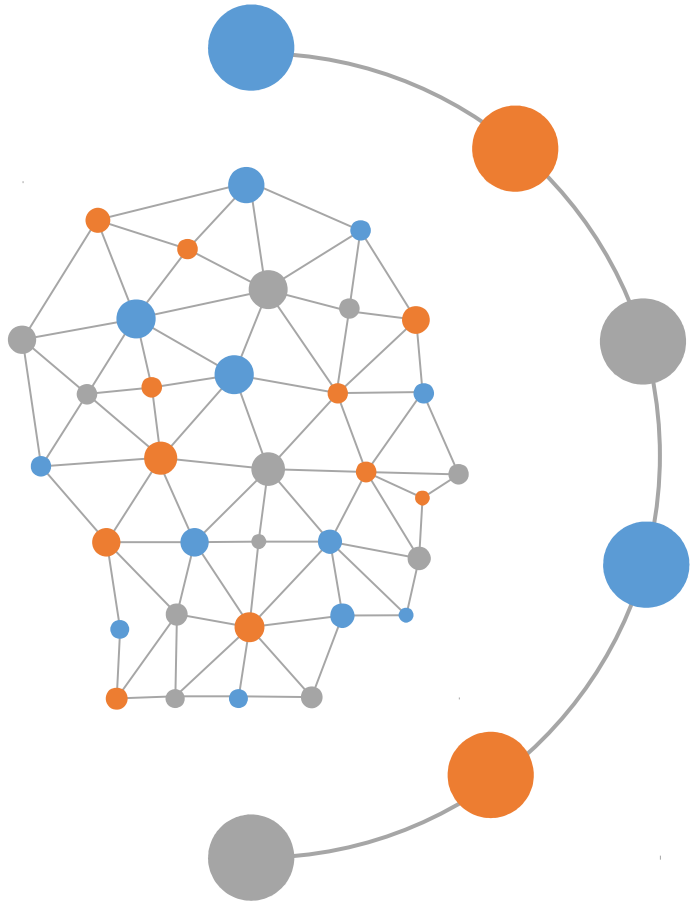
samjia2000

LAB
实验舱

DP 的类型

DP 的类型

- 数位 Dp
- 概率 Dp
- 树形 Dp
- 状态压缩 DP
- Dp 套 Dp



数位 DP

常见的数位 Dp 一般是用来统计或查找一个区间满足条件的数，然后按数位顺序 Dp ，一般需要仔细分情况讨论，常见处理如将区间拆为 [1~R]-[1~L-1] ，记忆化，预处理等。

当然也有一些题目不是给出 [L,R] 然后求其中满足条件的一些数（在难题分享里面会涉及这样的题目）

数位 DP

数位 DP- 例题 1

给出 l,r ，求出所有 l<=x<=r 的数中，满足 x 是 11 的倍数且所有数字和加起来也是 11 的倍数的 x 的个数。

相信一个熟练的 noip 选手能够很快的知道这是一道十分简单的题目。

数位 DP

数位 DP- 例题 2

给出 l,r,k,p ，其中 p 是一个小质数，求 $\sum_{i=l}^{r} \binom{k}{i} \bmod p$

使用卢卡斯定理即可

数位 DP

数位 DP- 例题 3

给出 l,r ，定义 f(i)=i 的各位数字的乘积，记 c(i) 表示 l 到 r 中满足 f(x)=i 的 x 的个数，求 c(i) 的前 k 大的和。

l,r<=10^12

由于是各位数字的乘积，那么其质因子一定在 2,3,5,7 中，搜出所有可能的数字，记 f[i][j] 为前 i 位对应的数字是 j 的方案数，即可求出所有非零的 c(i)

概率 DP

概率 Dp 是一类求事件概率或期望的 Dp 的总称，使用到的方法包括但不限于：

- 期望的线性性
- 补集转换
- 转化为计数问题

概率 DP

概率 Dp- 例题 1 noip2016 换教室

有 v 节点和 e 条带权无向边，牛牛有 n 个目的地。第 i 个目的地是 c[i]，接下来他会选择至多 m 天 {j1,j2...}，然后尝试将第 jk 个目的地更改为 d[jk]，成功的概率是 p[jk]。

从一个目的地到另外一个目的地需要花费与走过的边的边权相同的代价。

问最小的期望需要花费的代价是多少。

v<=300,n,m<=2000

期望的线性性

概率 DP

概率 Dp- 例题 2 CF838F

一个袋子里面有 n 个球，有 p_i 的概率袋子中有 i 个红球，随机取出一个球需要 c 的代价，取出一个红球的收益为 1 ，黑球的收益为 0 。

一开始不知道袋子中有多少个红球。

问最优策略下的期望收益最大是多少。

n<= 10000

概率 DP

每次取完一个球，可以知道的信息是：现在已经拿出了i个红球，j个黑球。

记$g_{i,j}$表示有i个红球，j个黑球的情况下，所有可能情况的概率和，$f_{i,j}$表示当前局面的最大期望收益，$p_{i,j}$表示当前局面下取球能得到红球的概率。

那么有：$f_{i,j} = \max\left(0, p_{i,j}(f_{i+1,j}+1) + (1-p_{i,j})f(i,j+1) - c\right)$

$$
\begin{aligned}
g_{i,j} &= \sum_{k=i}^{n} p_k \cdot \frac{\binom{i+j}{i}\binom{n-i-j}{k-i}}{\binom{n}{k}} \\
&= \sum_{k=i}^{n} p_k \cdot \frac{\binom{i+j}{i}\left(\binom{n-i-j-1}{k-i-1} + \binom{n-i-j-1}{k-i}\right)}{\binom{n}{k}} \\
&= \frac{i+1}{i+j+1} \cdot g_{i+1,j} + \frac{j+1}{i+j+1} \cdot g_{i,j+1}
\end{aligned}
$$

概率 DP

对于$p_{i,j}$有：

$$
\begin{aligned}
p_{i,j} &= \sum_{k=i}^{n} p_k \cdot \frac{\binom{i+j}{i}\binom{n-i-j}{k-i}}{\binom{n}{k}} \cdot \frac{k-i}{n-i-j} \\
&= \sum_{k=i}^{n} p_k \cdot \frac{\binom{i+j}{i}\binom{n-i-j-1}{k-i-1}}{\binom{n}{k}} \\
&= \frac{i+1}{i+j+1} g_{i+1,j}
\end{aligned}
$$

那么，根据$i+j$从大到小转移，就可以得到一个$O(n^2)$的做法了。

树形 DP

顾名思义，树形 dp 是基于树形结构进行的动态规划，往往会利用树的一些性质，如 dfs 序，树的直径等。
在 OI 中，树形 dp 出现的频率很高。（因为中国 OIer 动不动就往树上爬）
在一些树形 dp 的题目中，会利用重链剖分、长链剖分、树形依赖等思想方法解决问题，也常常会使用数据结构来优化 dp 。

树形 DP

树形 dp- 例题 1

有一棵 n 个点的有根树，要求选出 k 个点，使得每个节点周围都有至少一个节点是被选中的。

n<=100000,k<=min(n,100)

树形 DP

容易想到，设 f[i][j][0/1][0/1] 表示 i 的子树中选择了 j 个节点 /i 是否被选择 /i 是否被覆盖，然后进行转移。

转移是这样的：

```
for y in son(x):
    for i = 0 --> siz[y]:
        for j = 0 --> siz*[x]:
            f[x][i] (+) f[y][j] --> f[x'][i+j]
```

直接转移的复杂度是 $O(n^2)$ 的。

树形 DP

实际上，可以将 i,j 枚举到的上界改为 min(siz[],k)
这样改动之后，时间复杂度是 O(nk) 的。
why?
分成三种情况讨论：

① siz[x] 和 siz[y] 都大于 k，由于这样的合并次数不会超过 n/k 次，那么 n/k*k^2=nk

② siz[x] 和 siz[y] 中有一个不超过 k，然后另外一个大于 k，这种情况只会发生在 siz[x]>k 而 siz[y]<=k 的时候，可以看成每个 y 子树中的节点有一个 k 的贡献，故这种情况下总的次数不会超过 nk

③ siz[x] 和 siz[y] 都不超过 k 的情况，这种情况下，可以将所有极大的大小不超过 k 的子树的大小视为 b[1],b[2]...b[m]，那么产生的次数是 b[1]^2+b[2]^2+...+b[m]^2, 而 b[i]<=k，故不会超过 nk

所以总的时间复杂度是 O(nk)

树形 DP

树形 dp- 例题 2

有一棵 n 个节点的树，每个节点上面有一个非负整数，现在求选出一个包含节点 1 的联通子图，使得这个子图的点权和不超过 m 的方案数。

n,m <= 5000

树形 DP

求出 dfs 序 dfn[x] ，记 R[x] 为节点 x 的子树中 dfs 序最大的。
记 dp[i][j] 表示 dfn[x]<=i 的节点都已经决定了是否被选择了，并且总的权值和为 j 的方案数 .
然后考虑转移，对于 dfn[z]=i+1 的节点 z ，考虑是否选择 z ，如果不选择 z ，那么 z 的子树都不可以选，即 dp[i][j] --> dp[R[z]+1][j]
否则，如果选择 z ，那么 dp[i][j] --> dp[i+1][j+a[z]]
时间复杂度为 O(nm)
上述利用 dfs 序的方法被称为树形依赖。

状压 DP

基于状态压缩的 dp 是为了避免过多无用状态导致的时空复杂度的浪费，而采用压缩状态来进行 dp 。
使用状态压缩的 dp 包括但不限于：插头 dp ，斯坦纳树等
在一些特别的题目中，当看起来状态很多然而实际上只有很少一部分状态有用的时候，会采用搜索出所有的状态然后仅对这些状态进行转移的方法，以此提高程序的效率。

状压 DP

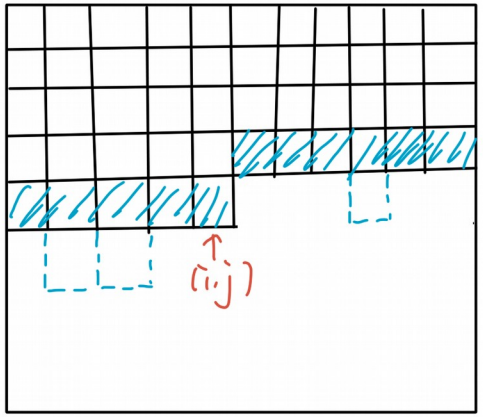
状压 DP- 例题 1

有一个 n*m 的格子图，每个格子上要么有障碍，要么是空的，现在要求用 1*2 的骨牌覆盖所有空的格子，要求每个格子被恰好一个骨牌覆盖，骨牌不可以覆盖有障碍的格子，求方案数模 1e9+7

n,m<=18

状压 DP

记 f(i,j,S) 表示现在处理到 (i,j) 这个格子， S 是一个表示 (i-1,j+1)..(i-1,m),(i,1)..(i,j) 的格子往下是否有凸出来的骨牌的状态。
如图：



于是对于每一个新的位，考虑这个位是否放一个骨牌，以及这个骨牌的方向。
时间复杂度 $O(nm2^m)$

状压 DP

状压 DP- 例题 2

有一个 n*m 的格子图，有一个格子 (x,y) ，现在要求选择一些格子，使得这些格子是一个四连通块且围住了格子 (x,y) （ (x,y) 不可以被选择）

n,m<=10

状压 DP

首先考虑判断 (x,y) 是不是被围住，类似于判断点是否在多边形内的方法，只需要求 (1,y),(2,y),(3,y)...(x-1,y) 这些格子中被选择的格子的数量的奇偶性，如果是奇数那么说明 (x,y) 被围起来了，否则说明 (x,y) 没有被围起来。
然后类似于上一题的 DP ，这里的状态不一样的是要记录这些格子之间的联通状态， 10 的时候有差不多 100000 种状态，转移的时候需要考虑联通块的合并。

状压 DP

状压 DP- 例题 3

有一个 n 个点 m 条边的无向图，每个点有点权，保证任意两个节点之间不存在一条长度超过 10 的简单路径。

现在要求选择一些节点作为关键点，使得对于任意一个节点，要么它是关键点，要么它周围有至少一个点是关键点。

求最小花费。

n<=2000,m<=25000

状压 DP

求出以 1 为根的 dfs 树，那么每个节点的深度都不超过 10 。
记 f(x,S) 表示已经确定了树上 1-x 的路径上的所有点是否被选择，以及每个没被选择的点是否被满足的最小花费。
使用树形依赖 dp ，按照 dfs 序转移，每次看这个新的点是否被选择并且当一个节点与后面的状态没有影响的时候需要判断合法性。
时间复杂度 $O(n*3^{10})$
注意：这里不使用树形依赖 dp 会得到复杂度较差的做法。

状压 DP

状压 dp- 例题 4
给定一个 n 个点 m 条边的带边权无向图。有 p 个特殊点，每个点还有一个特定的频道。
你需要选择边权和最小的一些边，使相同频道的特殊点两两联通。

$n \le 1000$ $m \le 3000$ $p \le 10$

状压 DP

斯坦纳树。
设 f(x,S) 表示选择了一些边，使得 S 中的点能够到达节点 x 的最小花费。

f(x,S) 的转移如下：

---

f(x,U)+f(x,V) --> f(x,U or V)
f(x,U) --> f(y,U) (x,y)\in E

然后对于每个频道的集合，可以找到让这些频道联通的最小花费，最后做一个子集 DP 即可求出答案。

时间复杂度 $O(2^p(n+m)\log n)$

DP 套 DP

DP 套 DP 的问题中，一般形式为， dp 对应的状态实际上维护了一个 dp 的状态。
具体的从题目来看。

DP 套 DP

DP 套 DP- 例题 1

给你一个只由 AGCT 组成的字符串 S ，对于每个 0<=i<=|S|, 问有多少个只由 AGCT 组成的长度为 m 的字符串 T, 使得 LCS(S,T)=i

|S|<=15,m<=1000

DP 套 DP

首先要解决的第一个 dp 问题是，计算两个串的 lcs ，那么一种做法是，对于 s 串，在枚举 t 串的前缀 t[1..x] 的过程中，维护 f[1..|s|] 其中 f[i] 表示 s[1..i] 跟 t[1..x] 的 lcs 。
容易发现 f 是不下降的，且 0<=f[i]-f[i-1]<=1

然后第二个 dp 问题就是原问题。
由于有第一个 dp 的解法，可以用一个状态 S 表示 f ，在加入 t 的新字符的时候改变对应的状态即可。
时间复杂度 $O(2^{|s|}m)$

其他类型

实际上，还有很多不同类型的 DP 。

其他类型

其他类型 - 例题 1

给出 n，随机一个 n 的排列 $\{P_n\}$，求 $\text{sum}(|P_i-P_{i-1}|)$ 的期望，模 1e9+7

n<=100

其他类型

笛卡尔树 DP

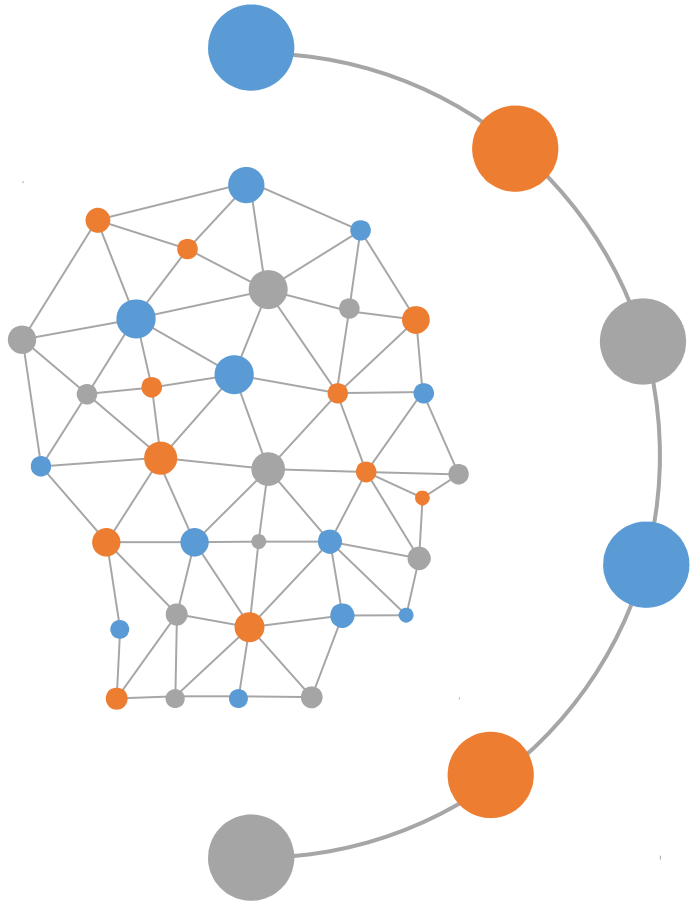
考虑最后 $\{P_n\}$ 的笛卡尔树，按照 P 从大到小的顺序加入每个点，那么实际上是一个合并左右儿子的过程，每次需要根据这个节点的左右儿子是否为空以及它是否是开头或结尾来进行转移。

dp 状态为 f[i][j][0/1][0/1] 表示现在已经确定了 P 值为 i~n 的节点，此时共有 j 棵子树，开头结尾是否确定时的期望值

DP 的优化

DP 的优化

- 决策单调性优化
- 斜率优化
- 凸优化
- Others



斜率优化

一般形式：

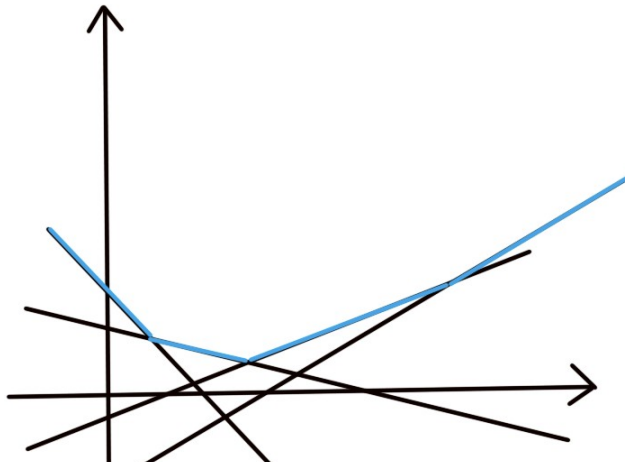
f[i]=max{k[i]*x[j]+y[j]} (k[i] 单调，下面假设 k[i] 单调递增)

对于决策点 j1 和 j2(x[j1]<x[j2]) ，当 j2 优于 j1 时：

$$k_i \cdot x_{j_1} + y_{j_1} < k_i \cdot x_{j_2} + y_{j_2}$$

$$k_i \cdot (x_{j_2} - x_{j_1}) > y_{j_1} - y_{j_2}$$

$$k_i > \frac{y_{j_1} - y_{j_2}}{x_{j_2} - x_{j_1}}$$



实际上，这是一个凸壳的结构，只需要维护用单调栈这个凸壳即可。

斜率优化

斜率优化 - 例题 1

给出一个长度为 n 的正整数序列，要求进行 k 次分割操作，每次操作的得分为分割后的两部分和的乘积。
最大化得分

n<=100000,k<=200

斜率优化

实际上是要最小化每段内部的乘积的和。

设 f[i][j] 表示前 i 个数字被分成 j 段的每段最小乘积和， pre[i]=a[1]+a[2]+..+a[i]

则 f[i][j]=max(f[i'][j-1]+(pre[i]-pre[i'])^2)

f[i'][j-1]+(pre[i]-pre[i'])^2=f[i'][j-1]+pre[i']^2-2*pre[i]*pre[i']+pre[i]^2

将 2*pre[i] 视为 k[i] ， pre[i'] 视为 x[i'],f[i'][j-1]+pre[i']^2 视为 y[i'] ，就可以直接斜率优化了。

决策单调性

一维决策单调性：
$f[i]=\max\{f[j]+w[j+1][i]|j<i\}$
记 $s[i]=\max\{j|f[j]+w[j+1][i]=f[i]$ 且 $j<i\}$
如果 s 单调不下降，那么称 f 满足一维决策单调性

二维决策单调性：
$f[k][i]=\max\{f[k-1][j]+w[k][j+1][i]|j<i\}$
$s[k][i]=\max\{j|f[k-1][j]+w[k][j+1][i]=f[k][i]$ 且 $j<i\}$
如果 s[k] 单调不下降，那么称 f 满足二维决策单调性

决策单调性

常见的算法有两种。

1. 单调栈 + 二分
2. 分治

决策单调性

单调栈 + 二分

记 $[l_i,r_i]$ 表示 i 作为最优决策点的区间，维护一个单调栈，单调栈中存储当前有用的决策。

新增一个决策 k 的时候，假设栈顶为 j ，那么如此操作：

如果在 l_j 处 k 比 j 优，那么将 j 弹出；否则在 $[l_j,r_j]$ 中二分出最小的点 t ，使得在 t 点 k 比 j 优，然后将 k 的区间设为 [t,n]

时间复杂度 O(n log n)

决策单调性

分治

设过程 solve(l,r,wl,wr) 表示已知对于 i=l..r ， i 的决策点在 [wl,wr] ，要求求出 f[l..r]

记 mid=(l+r)/2

那么求出 f[mid] 的决策点 p ，继续执行 solve(l,mid-1,wl,p) 和 solve(mid+1,r,p,wr)

时间复杂度 O(n log n)

分治做法实现简单，也更为常见，可以很好的用在二维决策单调性问题上。

决策单调性

决策单调性 - 例题 1

一个数轴上面有 n 对点，第 i 对点分别位于 a_i 与 b_i ，现在要求在数轴上面任意位置选择 k 个位置，然后 2n 个点每个点需要从 k 个点中选择一个点，然后移动到这个点，每一对点需要选择同一个点，要求最小化所有点移动的距离和。

k<=n<=100000,n*k<=100000

决策单调性

考虑任意一对点 a_i 和 b_i，假设 $a_i<b_i$，那么如果 $[a_i,b_i]$ 中有一个被选择的点，那么这个点对的花费为 b_i-a_i，否则假设左边第一个点为 L，右边第一个点为 R，那么花费为 $\min\{a_i+b_i-2L,2R-a_i-b_i\}$

那么假设所有点对至少花费 b_i-a_i

可以发现选择的点一定是给出的 2n 个点中的。

将这些点从小到大排序，记为 x[1]..x[2n]

记 f[t][i] 表示已经选择了 t 个点，最后一个是 x[i] 的最小花费。

f 的转移为：

$f[t][i]=\max\{f[t-1][j]+w(i,j)|j<i\}$

其中 $w(i,j)=\text{sum}(\min\{a_u+b_u-2x_i,2x_j-a_u-b_u\}|x_i<a_u<=b_u<x_j)$

f[t] 的决策点是单调的，而 w(i,j) 可以使用数据结构维护

使用分治计算 f，时间复杂度为 $O(nk\log^2 n)$

凸优化

凸优化 - 例题 1

有 n 个物品，现在要将这些物品分成连续的 k 段，如果 i,j 被分到了同一段，那么会产生 w(i,j) 的代价，要求最小化代价。

n<=4000,1<=k<=min(n,800)

凸优化

较为直接的思路是记 f[t][i] 表示将前 i 个物品分成了 t 段的最小花费
利用分治算法，可以做到 $O(n^2+nk \log n)$
f[k][1..i] 满足 f[k][i+1]-f[k][i]<f[k][i+2]-f[k][i+1]
可以看成一个斜率单调不降的函数
尝试选择一个 c ，设 g(c,k)=f[k][n]+kc
那么 g(c,k)<g(c,k+1) 当 c>f[k][n]-f[k+1][n]
对于一个 c ，可以将 f 的转移改写成 f[i]=min{f[j]+c+cost(j+1,i)|j<i} ，那么记 k(c) 为 f[n] 对应的切成的段的数量。
当 c 增大的时候， k(c) 随之单调下降。
于是可以二分 c ，每次计算 f 的过程中记录下此刻的段数，根据段数来判断 c 是否过大或过小。
于是时间复杂度变成 $O(n^2+n \log n \log W)$

凸优化

上面的例题不过是一个小小的例子，从前面的例子可以看出，当 dp 的答案满足一个凸的性质的时候，可以通过二分答案点附近的斜率，然后来去掉其中一维的限制，从而可以加速 dp 的计算。
这就是“凸优化”

其他的优化方法

在一些树上的 dp 问题中，长链剖分与重链剖分也是用来优化 dp 的重要方法。
重链剖分的构造方法?
长链剖分的构造方法?

在国内的各种重大比赛以及一些网络赛中，有许多 dp 题都是与数据结构相结合的。


END