模型抽象

高睿泉

南京外国语学校

2017 年 8 月 25 日

business

来源：GCJ2015 Round 3 Problem A

题目大意

给定一棵n个点的树，每个点有一个权值c_i。现要求选出若干个点，满足:权值最大的点和权值最小的差不超过D。如果一条边的两个端点都被选中，那么将这两个点用一条边连起来，求得到的新图中最多有多少个点与1号点连通。

数据范围

$1 \leq n, c_i, D \leq 10^6$

Solution

- 这道题可以用lct做到$O(n\log n)$

Solution

- 这道题可以用lct做到$O(n\log n)$
- 但存在一种给为巧妙的线性做法
- 如果有想法可以上来讲一下

Solution

- 考虑如何把被某个点独立开：

Solution

- 考虑如何把被某个点独立开:
- 对于点i,如果他被选中且与1连通，那么他到1号点路径上的点全部都要选

Solution

- 考虑如何把被某个点独立开：
- 对于点i,如果他被选中且与1连通，那么他到1号点路径上的点全部都要选
- 取$O = c_1$。取问题就转化为：现在有若干个形如$[O - l_i, O + r_i](l_i, r_i \geq 0, l_i + r_i \leq D)$的区间，要求选择一个数字x满足区间[O-x,O+D-x] 包含最多的上述区间

Solution

- 考虑如何把被某个点独立开：
- 对于点i,如果他被选中且与1连通，那么他到1号点路径上的点全部都要选
- 取$O = c_1$。取问题就转化为：现在有若干个形如$[O - l_i, O + r_i]$($l_i, r_i \geq 0, l_i + r_i \leq D$)的区间，要求选择一个数字x满足区间[O-x,O+D-x] 包含最多的上述区间
- 可以发现对于一个区间$[O - l_i, O + r_i]$,对$x \in [l_i, D - r_i]$的答案贡献为1

Solution

- 考虑如何把被某个点独立开：
- 对于点i,如果他被选中且与1连通，那么他到1号点路径上的点全部都要选
- 取$O = c_1$。取问题就转化为：现在有若干个形如$[O - l_i, O + r_i](l_i, r_i \geq 0, l_i + r_i \leq D)$的区间，要求选择一个数字x满足区间[O-x,O+D-x] 包含最多的上述区间
- 可以发现对于一个区间$[O - l_i, O + r_i]$,对$x \in [l_i, D - r_i]$的答案贡献为1
- 只需要利用部分和的技巧算出每个x取每个值时的答案即可

Solution

- 考虑如何把被某个点独立开：
- 对于点i,如果他被选中且与1连通，那么他到1号点路径上的点全部都要选
- 取$O = c_1$。取问题就转化为：现在有若干个形如$[O - l_i, O + r_i](l_i, r_i \geq 0, l_i + r_i \leq D)$的区间，要求选择一个数字x满足区间[O-x,O+D-x] 包含最多的上述区间
- 可以发现对于一个区间$[O - l_i, O + r_i]$,对$x \in [l_i, D - r_i]$的答案贡献为1
- 只需要利用部分和的技巧算出每个x取每个值时的答案即可
- 复杂度$O(n + D) - O(n + D)$

二分答案

- 二分答案是一个比较常用的技巧
- 比如NOIP 2015 D2T1

二分答案

- 二分答案是一个比较常用的技巧
- 比如NOIP 2015 D2T1
- 二分答案一般的作用是把问题改为判定问题
- 但必须保证二分的东西具有单调性

二分答案

- 二分答案是一个比较常用的技巧
- 比如NOIP 2015 D2T1
- 二分答案一般的作用是把问题改为判定问题
- 但必须保证二分的东西具有单调性
- 二分导数还可以替代三分求极值

二分答案

- 二分答案是一个比较常用的技巧
- 比如NOIP 2015 D2T1
- 二分答案一般的作用是把问题改为判定问题
- 但必须保证二分的东西具有单调性
- 二分导数还可以替代三分求极值
- 当然二分还可以用于具有特别限定的问题里面，但很多都超过了noip的难度

Decrease

来源：ARC079 Problem E

题目大意

给定一个长度为n的序列a_i，每次操作：

- 任选一个最大值将其减去n
- 把剩下的其他数字加1

求至少进行多少次操作后，最大的数字变得小于n。

数据范围

$1 \le n \le 500 \le a_i \le 10^{16} + 1000$

Solution

- 由于我们无法直接模拟运算的过程，那我们考虑如何验证如何判断经过x次是否每个数字均小于n:

Solution

- 由于我们无法直接模拟运算的过程，那我们考虑如何验证如何判断经过x次是否每个数字均小于n:
- 把操作变为所有数加1，最大的数减$n+1$，那么先给所有数字加x，计算每个数字需要减到n以下次数的总和

Solution

- 由于我们无法直接模拟运算的过程，那我们考虑如何验证如何判断经过x次是否每个数字均小于n:
- 把操作变为所有数加1，最大的数减$n+1$，那么先给所有数字加x，计算每个数字需要减到n以下次数的总和
- 如果总次数超过x次，那显然无法满足条件

Solution

- 由于我们无法直接模拟运算的过程，那我们考虑如何验证如何判断经过x次是否每个数字均小于n:
- 把操作变为所有数加1，最大的数减$n+1$，那么先给所有数字加x，计算每个数字需要减到n以下次数的总和
- 如果总次数超过x次，那显然无法满足条件
- 如果总次数不超过，那么由于在停止操作之前，减$n+1$的数字不会超过n，所以不会出现负数,用反证即可证出这样一定合法

Solution

- 但问题在于这样对于x是不具备单调性的:

Solution

- 但问题在于这样对于x是不具备单调性的:
- 27 0 0 0 0 0 0对于$x = 3$是合法的，但对于$x = 7$却不合法

Solution

- 但问题在于这样对于x是不具备单调性的:
- 27 0 0 0 0 0 0对于$x = 3$是合法的，但对于$x = 7$却不合法
- 考虑计算次数的公式$\sum_{i=1}^{n} \lfloor \frac{a_i + mid + 1}{n+1} \rfloor$

Solution

- 但问题在于这样对于x是不具备单调性的:
- 27 0 0 0 0 0 0对于$x = 3$是合法的，但对于$x = 7$却不合法
- 考虑计算次数的公式$\sum_{i=1}^{n} \lfloor \frac{a_i + mid + 1}{n+1} \rfloor$
- 如果mid合法，那么$mid + n + 1$一定合法

Solution

- 但问题在于这样对于x是不具备单调性的:
- 27 0 0 0 0 0 0对于$x = 3$是合法的，但对于$x = 7$却不合法
- 考虑计算次数的公式$\sum_{i=1}^{n} \lfloor \frac{a_i + mid + 1}{n+1} \rfloor$
- 如果mid合法，那么$mid + n + 1$一定合法
- 所以只需要枚举模n+1的余数，二分一下即可

Solution

- 但问题在于这样对于x是不具备单调性的:
- 27 0 0 0 0 0 0对于$x = 3$是合法的，但对于$x = 7$却不合法
- 考虑计算次数的公式$\sum_{i=1}^{n} \lfloor \frac{a_i+mid+1}{n+1} \rfloor$
- 如果mid合法，那么$mid + n + 1$一定合法
- 所以只需要枚举模n+1的余数，二分一下即可
- 复杂度$O(n^2 log\ a_i) - O(n)$

Complete The Graph

来源：Codeforces Round #372 (Div.1) Problme B

题目大意

给定一张n个点m条边的无向图，有一些边的边权是由你决定的，请在此基础上构造出一个S号点和T号点最短路恰好为L的图。

数据范围

$1 \le n \le 1000, 1 \le m \le 10000$

$1 \le L, w \le 10^9$，修改的边权在$[1, 10^{18}]$之间

Solution

- 将边权设为INF可以理解为没有这条边

Solution

- 将边权设为INF可以理解为没有这条边
- 显然边数越少最短路一定越大

Solution

- 将边权设为INF可以理解为没有这条边
- 显然边数越少最短路一定越大
- 二分前i条未决定边的边权为1(最小值),其他为INF的图的最短路是否小于等于L

Solution

- 将边权设为INF可以理解为没有这条边
- 显然边数越少最短路一定越大
- 二分前i条未决定边的边权为1(最小值),其他为INF的图的最短路是否小于等于L
- 如果i小于等于而i-1大于，那么可以证明第i条边一定在最短路上，且可以补上和L的差值后仍在最短路上

Solution

- 将边权设为INF可以理解为没有这条边
- 显然边数越少最短路一定越大
- 二分前i条未决定边的边权为1(最小值),其他为INF的图的最短路是否小于等于L
- 如果i小于等于而i-1大于，那么可以证明第i条边一定在最短路上，且可以补上和L的差值后仍在最短路上
- 特判一下需要特判不合法的情况

Solution

- 将边权设为INF可以理解为没有这条边
- 显然边数越少最短路一定越大
- 二分前i条未决定边的边权为1(最小值),其他为INF的图的最短路是否小于等于L
- 如果i小于等于而i-1大于，那么可以证明第i条边一定在最短路上，且可以补上和L的差值后仍在最短路上
- 特判一下需要特判不合法的情况
- 复杂度$O((nlog\ n + m)log\ w) - O(m + n)$(使用dijkstra算法)

Solution

- 将边权设为INF可以理解为没有这条边
- 显然边数越少最短路一定越大
- 二分前i条未决定边的边权为1(最小值),其他为INF的图的最短路是否小于等于L
- 如果i小于等于而i-1大于，那么可以证明第i条边一定在最短路上，且可以补上和L的差值后仍在最短路上
- 特判一下需要特判不合法的情况
- 复杂度$O((nlog\ n+m)log\ w)-O(m+n)$(使用dijkstra算法)
- 这题也可以算出所有边去最小值和最大值时的最短路，并围绕最小值时的最短路进行一番讨论，可以把最短路以外的部分压到线性

倍增

- 倍增是用来通过预处理连续2^i的信息，再通过二进制分解来完成询问(一般不能有修改，但可以动态从末尾加入)。
- 可以用于维护区间静态信息，树上lca等很多信息，一般预处理复杂度$O(n\log n)$,单次查询$O(\log n)$

倍增

- 倍增是用来通过预处理连续2^i的信息，再通过二进制分解来完成询问(一般不能有修改，但可以动态从末尾加入)。
- 可以用于维护区间静态信息，树上lca等很多信息，一般预处理复杂度$O(nlog\,n)$,单次查询$O(log\,n)$
- 求序列字区间的最小值：$f[i][j]$表示区间$[i, i+2^j-1]$内的最小值, 则有$f[i][j] = min(f[i][j-1], f[i+2^{j-1}][j-1])$

倍增

- 倍增是用来通过预处理连续2^i的信息，再通过二进制分解来完成询问(一般不能有修改，但可以动态从末尾加入)。
- 可以用于维护区间静态信息，树上lca等很多信息，一般预处理复杂度$O(n\log n)$,单次查询$O(\log n)$
- 求序列字区间的最小值：$f[i][j]$表示区间$[i,i+2^j-1]$内的最小值，则有$f[i][j]=min(f[i][j-1],f[i+2^{j-1}][j-1])$
- 那么对于区间$[l,r]$,取$x=\lfloor \log(r-l+1)\rfloor$,则最小值为$min(f[l][x],f[r-2^x+1][x])$

倍增

- 倍增是用来通过预处理连续2^i的信息，再通过二进制分解来完成询问(一般不能有修改，但可以动态从末尾加入)。
- 可以用于维护区间静态信息，树上lca等很多信息，一般预处理复杂度$O(nlog\, n)$,单次查询$O(log\, n)$
- 求序列字区间的最小值：$f[i][j]$表示区间$[i, i+2^j-1]$内的最小值，则有$f[i][j] = min(f[i][j-1], f[i+2^{j-1}][j-1])$
- 那么对于区间$[l, r]$,取$x = \lfloor log\,(r-l+1)\rfloor$,则最小值为$min(f[l][x], f[r-2^x+1][x])$
- 特殊的，区间最小最大值、gcd等(一个数计算多次和计算一次效果一样)查询可以做到$O(1)$

树上倍增

- $f[i][j]$表示i号点向上2^j的点的标号,则有$f[i][j] = f[f[i][j-1]][j-1]$

树上倍增

- $f[i][j]$表示i号点向上2^j的点的标号,则有$f[i][j]=f[f[i][j-1]][j-1]$
- 求lca的过程就是先将深度大的点向上移动到深度相同的位置，然后类似二分地枚举j看$f[x][j],f[y][j]$是否相同，不相同则更新
- 最后如果x,y相同则lca为x,否则为$f[x][0]$
- 同样倍增也可以类似区间地维护树上路径的权值信息

Best Edge Weight

来源：Codeforces Round #423 (Div. 1, rated, based on VK Cup Finals) Problem D

题目大意

给定一张n个点m条边的无向图,对于每条边：求在不修改其他边权的情况下，这条边最大为多少，使得他出现在任意一棵最小生成树上

数据范围

$1 \le n \le 2 \times 10^5, n-1 \le m \le 2 \times 10^5$

Solution

- 先求出最小生成树,对于最小生成树外的边：答案就是树上对应路径上边权最大值-1，直接用倍增处理即可

Solution

- 先求出最小生成树,对于最小生成树外的边：答案就是树上对应路径上边权最大值-1，直接用倍增处理即可
- 对于最小生成树内的边：答案就是覆盖这条边的路径的最小值-1,考虑一条路径对生成树的影响：将路径上的边对其权值取min

Solution

- 先求出最小生成树,对于最小生成树外的边：答案就是树上对应路径上边权最大值-1，直接用倍增处理即可
- 对于最小生成树内的边：答案就是覆盖这条边的路径的最小值-1,考虑一条路径对生成树的影响：将路径上的边对其权值取min
- 将倍增时的(i, j)考虑成树上i向上长度为2^j的路径，那么就很容易地把路径分成$O(log\ n)$段
- 再对于每个(i, j)维护c[i][j]表示对应路径整体修改的最小值

Solution

- 先求出最小生成树,对于最小生成树外的边：答案就是树上对应路径上边权最大值-1，直接用倍增处理即可
- 对于最小生成树内的边：答案就是覆盖这条边的路径的最小值-1,考虑一条路径对生成树的影响：将路径上的边对其权值取min
- 将倍增时的(i,j)考虑成树上i向上长度为2^j的路径，那么就很容易地把路径分成$O(log\ n)$段
- 再对于每个(i,j)维护c[i][j]表示对应路径整体修改的最小值
- 最后$c[i][j]$往$c[i][j-1], c[f[i][j-1]][j-1]$下传即可,得到的$c[i][0]$就是对应边答案,复杂度$O(nlog\ n)-O(nlog\ n)$

Typesetting

来源：2017 Multi-University Training Contest - Contest 6

题目大意

http://acm.hdu.edu.cn/showproblem.php?pid=6107

Solution

- 对于w，维护一个数组f[i]表示最大的[i,f[i]-1]可以塞进长为w一行内

Solution

- 对于w，维护一个数组f[i]表示最大的[i,f[i]-1]可以塞进长为w一行内
- 对于w,dw,w-pw-pw分别处理出来序列f1,f2,f3

Solution

- 对于w，维护一个数组f[i]表示最大的[i,f[i]-1]可以塞进长为w一行内
- 对于w,dw,w-pw-pw分别处理出来序列f1,f2,f3
- 倍增维护用长度为w的2^j行，从i开始可以最大的可以塞进的区间(f1)

Solution

- 对于w，维护一个数组f[i]表示最大的[i,f[i]-1]可以塞进长为w一行内
- 对于w,dw,w-pw-pw分别处理出来序列f1,f2,f3
- 倍增维护用长度为w的2^j行，从i开始可以最大的可以塞进的区间(f1)
- 同样对于中间有一个图片的情况也维护一个倍增(f2,f3)

Solution

- 对于w，维护一个数组f[i]表示最大的[i,f[i]-1]可以塞进长为w一行内
- 对于w,dw,w-pw-pw分别处理出来序列f1,f2,f3
- 倍增维护用长度为w的2^j行，从i开始可以最大的可以塞进的区间(f1)
- 同样对于中间有一个图片的情况也维护一个倍增(f2,f3)
- 每次询问直接判断文字在哪一段结束，对于结束的那一段，利用倍增二分一下即可

Solution

- 对于w，维护一个数组f[i]表示最大的[i,f[i]-1]可以塞进长为w一行内
- 对于w,dw,w-pw-pw分别处理出来序列f1,f2,f3
- 倍增维护用长度为w的2^j行，从i开始可以最大的可以塞进的区间(f1)
- 同样对于中间有一个图片的情况也维护一个倍增(f2,f3)
- 每次询问直接判断文字在哪一段结束，对于结束的那一段，利用倍增二分一下即可
- 复杂度$O((n+Q)log\ n)-O(nlog\ n)$

Senior Pan

来源：2017 Multi-University Training Contest - Contest 9

题目大意

给定一张n个点m条边的有向图，其中k个点是带标记的，求这些标记点两两最短路的最小值。

数据范围

$1 \le n, m \le 10^5, 1 \le k \le n$

Solution

- 可以把问题转化为k1个白点，k2个黑点(黑点白点不重复),求白点和黑点之间两两最短路的最小值

Solution

- 可以把问题转化为k1个白点，k2个黑点(黑点白点不重复),求白点和黑点之间两两最短路的最小值
- 这样直接建立S向白点连边，黑点向T连边，跑一边最短路即为答案

Solution

- 可以把问题转化为k1个白点，k2个黑点(黑点白点不重复),求白点和黑点之间两两最短路的最小值
- 这样直接建立S向白点连边，黑点向T连边，跑一边最短路即为答案
- 将k个点按照其标号的二进制表示第i位上的01情况可以将其分成白点黑点(要正反做两遍)

Solution

- 可以把问题转化为k1个白点，k2个黑点(黑点白点不重复),求白点和黑点之间两两最短路的最小值
- 这样直接建立S向白点连边，黑点向T连边，跑一边最短路即为答案
- 将k个点按照其标号的二进制表示第i位上的01情况可以将其分成白点黑点(要正反做两遍)
- 将每一位拿出来做一遍，最小值就是答案

Solution

- 可以把问题转化为k1个白点，k2个黑点(黑点白点不重复),求白点和黑点之间两两最短路的最小值
- 这样直接建立S向白点连边，黑点向T连边，跑一边最短路即为答案
- 将k个点按照其标号的二进制表示第i位上的01情况可以将其分成白点黑点(要正反做两遍)
- 将每一位拿出来做一遍，最小值就是答案
- 复杂度$O((n\log n+m)\log w)-O(m+n)$(使用dijkstra算法)

差分约束系统

- 现有n个变量a_i和一些形如$a_x \le a_y + d_{x,y}$的限制条件

差分约束系统

- 现有n个变量a_i和一些形如$a_x \le a_y + d_{x,y}$的限制条件
- 求一组可行解或判断有无解

差分约束系统

- 现有n个变量a_i和一些形如$a_x \leq a_y + d_{x,y}$的限制条件
- 求一组可行解或判断有无解
- 对于限制条件$a_x \leq a_y + d_{x,y}$，建一条$y \rightarrow x$,边权为$d_{x,y}$的边

差分约束系统

- 现有n个变量a_i和一些形如$a_x \leq a_y + d_{x,y}$的限制条件
- 求一组可行解或判断有无解
- 对于限制条件$a_x \leq a_y + d_{x,y}$，建一条$y \rightarrow x$,边权为$d_{x,y}$的边
- 跑一边最短路，所得每个点的最短路即为一组可行解(其实是每个变量的上界)

差分约束系统

- 现有n个变量a_i和一些形如$a_x \leq a_y + d_{x,y}$的限制条件
- 求一组可行解或判断有无解
- 对于限制条件$a_x \leq a_y + d_{x,y}$，建一条$y \to x$,边权为$d_{x,y}$的边
- 跑一边最短路，所得每个点的最短路即为一组可行解(其实是每个变量的上界)
- 无解的情况就是出现了负环，在spfa过程中如果一个点入队n次即代表出现了负环

差分约束系统

- 现有n个变量a_i和一些形如$a_x \le a_y + d_{x,y}$的限制条件
- 求一组可行解或判断有无解
- 对于限制条件$a_x \le a_y + d_{x,y}$，建一条$y \to x$,边权为$d_{x,y}$的边
- 跑一边最短路，所得每个点的最短路即为一组可行解(其实是每个变量的上界)
- 无解的情况就是出现了负环，在spfa过程中如果一个点入队n次即代表出现了负环
- 例题：http://poj.org/problem?id=3169

差分

- 有些变换对于序列直接的影响非常的复杂，很难找到方法维护

差分

- 有些变换对于序列直接的影响非常的复杂，很难找到方法维护
- 将其差分(甚至差分多次)后却能大大简化问题，利于维护

差分

- 有些变换对于序列直接的影响非常的复杂，很难找到方法维护
- 将其差分(甚至差分多次)后却能大大简化问题，利于维护
- 比如区间修改、单点查询的树状数组就是维护差分序列

差分

- 有些变换对于序列直接的影响非常的复杂，很难找到方法维护
- 将其差分(甚至差分多次)后却能大大简化问题，利于维护
- 比如区间修改、单点查询的树状数组就是维护差分序列
- 差分在组合数学中也有很重要的应用

差分

- 有些变换对于序列直接的影响非常的复杂，很难找到方法维护
- 将其差分(甚至差分多次)后却能大大简化问题，利于维护
- 比如区间修改、单点查询的树状数组就是维护差分序列
- 差分在组合数学中也有很重要的应用
- 差分不一定是想相邻项相减，也有时候需要特别的差分

lamp

来源：TCO17 Round 2A 1000pts

题目大意

给定一排n个路灯的初始开关状态a_i，可以进行至多k次开关操作：

- 选择一个位置$x(0 < x < n-2)$，满足位置$x-1$,x,$x+1$,$x+2$上恰好有一个灯开着，或有一个灯关着
- 将区间$[0,x]$或$[x+1,n-1]$内所有开关状态反转

求最后n个路灯能有多少种开关状态。

数据范围

$4 \le n \le 50, 1 \le k \le 10^9$

Solution

- 考虑差分序列$b_i = a_i \, xor \, a_{i+2}, (i < n-2)$

Solution

- 考虑差分序列$b_i = a_i \ xor \ a_{i+2}, (i < n-2)$
- 对于一次操作，问题就变成了每次交换一个相邻1个0和1个0，同时可以选择是否把序列01反转

Solution

- 考虑差分序列$b_i = a_i \; xor \; a_{i+2}, (i < n-2)$
- 对于一次操作，问题就变成了每次交换一个相邻1个0和1个0，同时可以选择是否把序列01反转
- 由于反转对序列相邻01关系不构成影响，可以最后直接乘2(前提是能反转)

Solution

- 考虑差分序列$b_i = a_i \, xor \, a_{i+2}, (i < n-2)$
- 对于一次操作，问题就变成了每次交换一个相邻1个0和1个0，同时可以选择是否把序列01反转
- 由于反转对序列相邻01关系不构成影响，可以最后直接乘2(前提是能反转)
- 我们只用考虑交换相邻01的问题

Solution

- 考虑差分序列$b_i = a_i \, xor \, a_{i+2}, (i < n-2)$
- 对于一次操作，问题就变成了每次交换一个相邻1个0和1个0，同时可以选择是否把序列01反转
- 由于反转对序列相邻01关系不构成影响，可以最后直接乘2(前提是能反转)
- 我们只用考虑交换相邻01的问题
- 考虑两个01序列之间最少操作次数：第一个序列第i个1一定对应第i个1，则代价为对应1位置差的绝对值之和

Solution

- 考虑差分序列$b_i = a_i \ xor \ a_{i+2}, (i < n-2)$
- 对于一次操作，问题就变成了每次交换一个相邻1个0和1个0，同时可以选择是否把序列01反转
- 由于反转对序列相邻01关系不构成影响，可以最后直接乘2(前提是能反转)
- 我们只用考虑交换相邻01的问题
- 考虑两个01序列之间最少操作次数：第一个序列第i个1一定对应第i个1，则代价为对应1位置差的绝对值之和
- 很容易发现这是背包的模型：$dp_{i,j,k}$表示做到第i个1，放在了第j个位置上，现在总代价为k的方案数

Solution

- 可以得到转移:
  $$dp_{i,j,k} = \sum_{j'=0}^{j-1} dp_{i-1,j',k-|j-pos_i|}$$
  其中pos_i表示初始序列第i个1的位置

Solution

- 可以得到转移:
$$dp_{i,j,k} = \sum_{j'=0}^{j-1} dp_{i-1,j',k-|j-pos_i|}$$
其中pos_i表示初始序列第i个1的位置
- 这样状态$O(n^4)$,转移$O(n)$，可能是过不去的

Solution

- 可以得到转移:
  $$dp_{i,j,k} = \sum_{j'=0}^{j-1} dp_{i-1,j',k-|j-pos_i|}$$
  其中pos_i表示初始序列第i个1的位置
- 这样状态$O(n^4)$,转移$O(n)$，可能是过不去的
- 仔细发现我们只需维护一个前缀和$S_{i,j,k} = \sum_{j'=0}^{j} dp_{i,j',k}$即可把转移复杂度压到$O(1)$

Solution

- 可以得到转移:
  $$dp_{i,j,k} = \sum_{j'=0}^{j-1} dp_{i-1,j',k-|j-pos_i|}$$
  其中pos_i表示初始序列第i个1的位置
- 这样状态$O(n^4)$,转移$O(n)$，可能是过不去的
- 仔细发现我们只需维护一个前缀和$S_{i,j,k} = \sum_{j'=0}^{j} dp_{i,j',k}$即可把转移复杂度压到$O(1)$
- 总复杂度$O(n^4) - O(n^3)$

状态变换

- 比如$dp_{i,j}$表示前i个东西用了j个位置，最大收益是多少
- 当这个位置上限非常大，但总收益不大的时候，可以把收益放进状态
- 变成$dp_{i,j}$表示前i个东西收益为j，最少用多少个位置
- 有的时候第二种状态并不容易直接想出，通过第一种想法转化有助于解题

binary

来源：GCJ2015 World Final problem A

题目大意

一个长度n的序列，某个位置被标记了，每次可以检测这个位置是否在$[1, i]$中，代价为c_i。请找出一种二分策略，使得在最坏情况下代价最小。

数据范围

$1 \leq n \leq 10^6, 1 \leq c_i \leq 9$

Solution

- 考虑区间dp，表示现在已知在区间$[l, r]$中所需的最小代价

Solution

- 考虑区间dp，表示现在已知在区间$[l, r]$中所需的最小代价
- 则$dp_{l,r} = min\{max(dp_{l,k}, dp_{k+1,r})\}, k \in [l, r)$

Solution

- 考虑区间dp，表示现在已知在区间$[l, r]$中所需的最小代价
- 则$dp_{l,r} = min\{max(dp_{l,k}, dp_{k+1,r})\}, k \in [l, r)$
- 可以发现得到的答案的上限是$\lfloor 9log\ n \rfloor$，那我们可以把状态变为$dp_{l,C}$,表示从l开始用C代价最大可以判断出的区间的右端点

Solution

- 考虑区间dp，表示现在已知在区间$[l, r]$中所需的最小代价
- 则$dp_{l,r} = min\{max(dp_{l,k}, dp_{k+1,r})\}, k \in [l, r)$
- 可以发现得到的答案的上限是$\lfloor 9log\ n \rfloor$，那我们可以把状态变为$dp_{l,C}$,表示从l开始用C代价最大可以判断出的区间的右端点
- 枚举一分为二使用的代价，可以得到转移:
  $dp_{l,C} = max\{dp_{pre(dp_{l,C-C'},C')+1,C-C'}\}$
  其中$pre(i,j)$,表示j在$c_{[1,i-1]}$中的最后一个位置，需要通过预处理得到

Solution

- 考虑区间dp，表示现在已知在区间$[l,r]$中所需的最小代价
- 则$dp_{l,r} = min\{max(dp_{l,k}, dp_{k+1,r})\}, k \in [l,r)$
- 可以发现得到的答案的上限是$\lfloor 9log\ n \rfloor$，那我们可以把状态变为$dp_{l,C}$,表示从l开始用C代价最大可以判断出的区间的右端点
- 枚举一分为二使用的代价，可以得到转移:
  $dp_{l,C} = max\{dp_{pre(dp_{l,C-C'},C')+1,C-C'}\}$
  其中$pre(i,j)$,表示j在$c_{[1,i-1]}$中的最后一个位置，需要通过预处理得到
- 复杂度$O(9^2 nlog\ n) - O(10n)$

Solution

- 考虑区间dp，表示现在已知在区间$[l, r]$中所需的最小代价
- 则$dp_{l,r} = min\{max(dp_{l,k}, dp_{k+1,r})\}, k \in [l, r)$
- 可以发现得到的答案的上限是$\lfloor 9log\ n \rfloor$，那我们可以把状态变为$dp_{l,C}$,表示从l开始用C代价最大可以判断出的区间的右端点
- 枚举一分为二使用的代价，可以得到转移:
  $dp_{l,C} = max\{dp_{pre(dp_{l,C-C'},C')+1,C-C'}\}$
  其中$pre(i, j)$,表示j在$c_{[1,i-1]}$中的最后一个位置，需要通过预处理得到
- 复杂度$O(9^2 nlog\ n) - O(10n)$
- 利用单调性可以做到$O(9nlog\ n)$

grid

来源：GCJ2015 World Final problem B

题目大意

要求在一个$n \times n$的网格纸上放置数字1、2、3，满足一下条件：

- 每行每列的和均为3
- 每行每列最多只有2个格子上面有数字

求有多少种方案至少包含x个3。

数据范围

$1 \le n, x \le 10^6$

Solution

- 对于这类网格纸问题，可以建立一个二分图：R_i为第i行对应的点，C_j为第j列对应的点

Solution

- 对于这类网格纸问题，可以建立一个二分图：R_i为第i行对应的点，C_j为第j列对应的点
- 对于这题，在二分图上的意义就是每个点要么连一个类型3的边，要么连一条类型1一条类型2

Solution

- 对于这类网格纸问题，可以建立一个二分图：R_i为第i行对应的点，C_j为第j列对应的点
- 对于这题，在二分图上的意义就是每个点要么连一个类型3的边，要么连一条类型1一条类型2
- 枚举类型3的个数，剩下的问题就可以变为左右各m个点的二分图，每个点入度出度均为1，求这样图的个数

Solution

- 对于这类网格纸问题，可以建立一个二分图：R_i为第i行对应的点，C_j为第j列对应的点
- 对于这题，在二分图上的意义就是每个点要么连一个类型3的边，要么连一条类型1一条类型2
- 枚举类型3的个数，剩下的问题就可以变为左右各m个点的二分图，每个点入度出度均为1，求这样图的个数
- 如果打表可以发现这是一个错排

Solution

- 对于这类网格纸问题，可以建立一个二分图：R_i为第i行对应的点，C_j为第j列对应的点
- 对于这题，在二分图上的意义就是每个点要么连一个类型3的边，要么连一条类型1一条类型2
- 枚举类型3的个数，剩下的问题就可以变为左右各m个点的二分图，每个点入度出度均为1，求这样图的个数
- 如果打表可以发现这是一个错排
- 如果从图论角度考虑：把左边每个点连一条边向他原来连向的点连向的点，就成为了m个点形成了若干个有向的大于1的环的计数问题(左边点原来连向的点可以组成任意一个全排列，乘上$m!$即可)

Solution

- 对于有向环，可以理解为一个轮换，也就是一个错排

Solution

- 对于有向环，可以理解为一个轮换，也就是一个错排
- 最后复杂度$O(n) - O(n)$

Solution

- 对于有向环，可以理解为一个轮换，也就是一个错排
- 最后复杂度$O(n) - O(n)$
- 也可以暴力dp:f_i表示左右各i个点的图连有向边的方案数

Solution

- 对于有向环，可以理解为一个轮换，也就是一个错排
- 最后复杂度$O(n)-O(n)$
- 也可以暴力dp:f_i表示左右各i个点的图连有向边的方案数
- $f_i=\sum\limits_{j=0}^{i-2}\binom{i-1}{j}\cdot(i-j-1)!\cdot f_j=(i-1)!\sum\limits_{j=0}^{i-1}\frac{f_j}{j!}$

Solution

- 对于有向环，可以理解为一个轮换，也就是一个错排
- 最后复杂度$O(n) - O(n)$
- 也可以暴力dp:f_i表示左右各i个点的图连有向边的方案数
- $f_i = \sum_{j=0}^{i-2} \binom{i-1}{j} \cdot (i-j-1)! \cdot f_j = (i-1)! \sum_{j=0}^{i-1} \frac{f_j}{j!}$
- 也可以用累和优化到$O(n)$

谢谢大家!