Cai0715 解题表格

目录：

数据结构：

题目来源	题目名称	题目大意
URAL 1147	Shaping Regions	A*B 的矩阵，按上下顺序给出 N 个小矩形，每个小矩形都有一种颜色，求最后所有颜色的覆盖面积。
算法讨论	因为是计算面积，且面积必然存在于两条边之间。所以，我们按 x 坐标划分区间，即 x(i)与 x(i+1)为一个区间，我们求这个区间内的颜色分布，将所有的区间组合起来就是最后结果。那么怎么求这个区间内的颜色分布呢，我们利用线段树。只要横跨 x(i)这条线段的矩形，我们就对 y 轴进行染色。最后，若 y 轴的某跳线段有色，则将(x(i+1)-x(i))*线段长度累加进答案数据即可。	
其它	时间复杂度为 O（N^2*LogN）。	
PKU 2155	Mobile phones	标准的树套树练习题。
算法讨论	如果这个题是一维的染色，显然是用线段树。而这里变成了二维，那我们就可以使用二维线段树，也就是树套树。关于树套树的讲解，可以去看 2004 年集训队薛矛的论文。这个题，可以说是比较标准的二维线段树模型，直接做就行了。此题用二维树状数组实现会更优。	
其它	时间复杂度为 O（M*LogN^2）。	
PKU 3788	Crazy Thairs	N 个数（N<=50000），求 5 个互相不逆序的数的组合有多少个。
算法讨论	N 的范围是 50000，而数的大小却在 10^9 内，很明显需要进行离散化。离散之后，我们开辟 4 个树状数组，第 1 个树状数组用来存第 I 个数在组合中为位置 1 时的情况数，直接累加 1 即可；第 2 个树状数组用来存第 I 个数在组合中为位置 2 时的情况数，这里就不能直接累加 1 了，而是要看第一个树状数组中比它小的数，位于第 1 个位置上的数有多少个，然后累加。依此类推。最后，由于 C（50000,4）可以用 qword 存下，所以树状数组操作时，不用高精度。不过，C（50000,5）却无法存下，所以仅仅在统计结果时需要使用高精度加法。	
其它	时间复杂度为 O（NlogM*K）（K 为高精度时产生的常数）。	
PKU 2104	K-th Number	N 个数，M 次询问，每次询问 L..R 区间内第 K 大的数是多少，输出。
算法讨论	看了众多的讲解，大体思路是将数组进行归并排序，同时在归并排序中建立线段树。然后二分答案，在线段树中查找区间内有几个比它小的树，这里的查找同样使用二分。具体的看代码吧。另外，线段树套平衡树也可以用在这里，但是无疑会加大编程复杂度和时空复杂度。	
其它	时间复杂度为 O（N(LogN)^2+M(LogN)^3）。	

PKU 2774	Long Long Message	两个串，长度小于等于 10^5，求它们的任意子串的最长公共前缀。
算法讨论	将两串并为一串，两串中间添加特殊字符，先构造后缀数组，然后计算出 height 数组，到底怎么计算参见 WC2009 罗穗骞的论文。然后，利用后缀数组的性质，最长的公共前缀一定出现在后缀数组中第 I-1 个位置和第 I 个位置，即 height[I]，而且 sa[i-1]和 sa[i]分别处于第一个字符串和第二个字符串，这样取最大值即可。	
其它	时间复杂度为 O（NLogN）。	
PKU 1743	Musical Theme	一个串，长度小于等于 20000，求其最长的不可重叠的重复子串。
算法讨论	这道题要求 A 中两子串，每个子串相邻之差为定值，所以我们构造每相邻两个数相减所得数列 B，问题就转换成要求 B 中子串相等的最大长度。由于不可重叠，我们则二分枚举最大长度 x。然后用 x 将 height 数组分组，求每组中 sa[I]的 max 和 min，若相减大于等于 x，则说明没有重叠，返回 true。	
其它	时间复杂度为 O（NLogN）。	
PKU 3261	Milk Patterns	一个串，长度小于等于 20000，求其可重叠、至少重复 K 次的子串。
算法讨论	和 PKU 1743 很思想大概是一样的，只不过这次分组后，需要查看是否有一组不小于 K 个元素。还有，此题需要先进行离散化。	
其它	时间复杂度为 O（NLogN）。	
Uscao 1.3.1	Calfflac	一个串，长度小于等于 20000，求其最长的回文串长度。
算法讨论	什么方法都可以过，这里用了后缀数组+RMQ。思想是，先取出所有英文字母，然后将串倒序加倒原串后面。所以求最长回文子串就转换成了求后缀的最长前缀。具体的细节部分，看程序吧。	
其它	时间复杂度为 O（NLogN）。	
PKU 1986	Distance Queries	求两点的最近公共祖先。
算法讨论	由于题目给的是无向树，我们先用边表存边，然后转换成有根树。接着就是很求最近公共祖先了。可以用 Tarjan 算法，但是我嫌太麻烦，所以将这个问题转换成了 RMQ 问题。接着就是朴素的 ST 预处理，然后查询即可。	
其它	时间复杂度为 O（NLogN）。	
PKU 2985	The k-th Largest Group	有 N 个组，M 个操作。操作 1：将任意两组 A 和 B 合并；操作 2：查询第 K 大组中有几个元素。

<table>
<tr><td>算
法
讨
论</td><td colspan="2">使用相对独立的并查集和平衡树，之所以说相对独立，因为他们之间的联系仅仅依靠某组的元素个数。对于操作 1，我们可以使用并查集，同时在平衡树里查找值等于 Son[A]和等于 Son[B]的结点删掉，然后再添加值等于 Son[A]+Son[B]的新结点。对于操作 2，我们利用平衡树的 Find 过程，轻松实现。</td></tr>
<tr><td>其它</td><td colspan="2">时间复杂度为 O（NLogN）。</td></tr>
<tr><td>SPOJ 726</td><td>Promotion</td><td>有 N 组数，每次选取所有前 N 组数中最大值和最小值，将它们的差累加进 ans，然后删除掉这两个数，问最后 ans 的值是多少。</td></tr>
<tr><td>算
法
讨
论</td><td colspan="2">取最大值与最小值，可以用双端堆来实现。不过，双端堆完全可以被平衡树所代替。所以，这个题我们还是使用 SBT。每次利用查找过程寻找最大与最小，删除即可。其实，这个题还有另一种更加高效的方法，即分段哈希。因为值域在 Abs(10^6)内，所以我们可以开一个 1..2*10^6 的数组来储存这些值，将它们分成 Sqrt（2*10^6）段，定义 D[X]代表 X 出现的次数，C[X]代表 X 所在段内元素的个数，每次加入一个值则累加 D[X]和 C[X]，查找的时候若 C[X]有元素，则继续查找 D[X]。</td></tr>
<tr><td>其它</td><td colspan="2">时间复杂度为 平衡树 O（NM*LogNM）、分段 Hash（NM）。</td></tr>
<tr><td>PKU 3277</td><td>City Horizon</td><td>给你个 N 个矩形的 l，r，h，求这 N 个矩形所占的面积。</td></tr>
<tr><td>算
法
讨
论</td><td colspan="2">先进行离散化，分别对 X 轴和 Y 轴进行离散。然后建立线段树，线段树有 2 个主域，cover 和 last，cover 代表这条线段被覆盖了多少次，last 代表这条线段被谁最后一次覆盖的。经过离散化后，我们可以处理出 2*N 条垂直于 X 轴的线段 P，若 P 是某矩形左边的线段，则 data[p]=1，否则 data[p]=-1。然后按升序插入这 2*N 条线段，每次用 data 修改 cover，在修改之前若 cover>0，则计算面积。具体的看代码，思想和 picture 的思想差不多。</td></tr>
<tr><td>其它</td><td colspan="2">时间复杂度为 O（NLogN）。</td></tr>
<tr><td>Vijos 1183</td><td>Fish && kitty</td><td>有 N 个数，和一个数 K。第一问，这 N 个数组成的序列中，任意连续两个数之差不超过 K，问有多少种方案。第二问，最长的序列长度是多少。</td></tr>
<tr><td>算
法
讨
论</td><td colspan="2">很明显的可以得到一个递推方程，F[I]:=Sum{F[J] | A[I]-K<=A[J]<=A[I]+K}，初始 F[I]=1，则 Ans=Sum{F[I]}-N。由于 N<=10^6，所以必须使用高效的数据结构，即线段树。对于第一问，我们将原方程中的 A[I]-K<=A[J]<=A[I]+K 就变成一条（A[I]-K,A[I]+K）的线段，我们每次查询这条线段的 data 域累加到 F[I]上即可，最后我们再将 F[I]作为一个点插入线段树，包含这个点的线段的 data 域累加上 F[I]。对于第二问，我们定义 Max[I]代表以第 I 个数为结尾的最长序列长度，所以 Max[I]可以再求 F[I]的过程中得到，最后我们将 Max[I]随着 F[I]一起插入回线段树中即可。</td></tr>
<tr><td>其它</td><td colspan="2">时间复杂度为 O（NLogN）。</td></tr>
<tr><td>HDOJ 1823</td><td>Luck and Love</td><td>让你插入一些点，每个点拥有 A,B,T 三个值。然后让你查询一些点，满足 A 在给定的 A1,A2 范围内,B 在给定的 B1,B2 范围内，求最大的 T 值。</td></tr>
</table>

算法讨论	标准的二维线段树，由于只有一位小数，我们乘以个 10 就行了。。。然后树套树，一个控制 A，一个控制 B 即可。另外，这个题目的数据非常的阴险，即出现 A1>A2 或 B1>B2 的情况，将它们反过来就行了。。。WA 了 N 次，就是 WA 在这里了。。。	
其它	时间复杂度为 O（NLogN^2）。	
ZJU 2112	Dynamic Rankings	有 N 个数，两种操作。Q：查询 L 到 R 区间内第 K 小的数；C：修改第 T 个数为 X。对于每个查询，输出。
算法讨论	这个题的 N 高达 50000，M 高达 10000，所以必须使用高效的数据结构。对于区间的查询，显然是使用线段树，在区间内查询第 K 小，则在线段树里套一个平衡树即可，然后二分枚举答案。思想不是太复杂，不过很多细节部分要处理。还有一个问题就是如何解决空间问题，由于线段树的深度至多是 LogN，每一层深度的节点占据的位置至多是 1..N，所以根据每个线段树节点的深度和区间左范围，我们可以确定套在其身上的平衡树所占的数组位置，所以空间优化到了 NlogN。	
其它	时间复杂度为 O（NLogN^3）。	
PKU 3294	Life Forms	给你 N 个字符串，输出若干个子串，满足大于 N/2 个字符串含有它。
算法讨论	先将 N 个字符串并成一个字符串，中间用不同的符号分割开。然后利用后缀数组求出 Height 数组，二分枚举答案 X，将 Height 分为若干组（如何分组？若 Height[I]大于等于 X，则属于上一组，否则属于新的一组），若某个组里面的后缀的来源是大于 N/2 个字符串的，则可行，继续放大 X，否则缩小 X。若最后 X=0 则 No Answer，否则再次分组，输出满足条件的组的前缀。	
其它	时间复杂度为 O（NLogN）。	
SPOJ 220	Relevant Phrases of Annihilation	有 N 个字符串，求这个 N 个字符串中（至少重复两次）（不可重叠）的子串的（最大长度）。例如：ababa 和 cbacba 中，解是 2，即 ba。
算法讨论	和 PKU 3294 这个题目大同小异，也是二分枚举，分组。但是这次在每个组里检查的是，（至少重复两次）（不可重叠），第一个可以用一个计数数组累加实现，第二个我们记录每个字符串的后缀的起点的最大最小值，若差不小于二分的答案 X，则满足。	
其它	时间复杂度为 O（NlogN*M）。	
PKU 1470	Closest Common Ancestors	有一颗树，求给定两节点的公共祖先，最后输出每个节点当做祖先的次数。
算法讨论	经典的 Trajan 算法，关于 Trajan 在算法艺术里有讲解。	
其它	时间复杂度为 O（N+M）。	

<table>
<tr><td>PKU 3368</td><td>Frequent values</td><td>给你 N 个数，升序排列，问 L,R 区间内出现频率最多的是谁。</td></tr>
<tr><td>算法讨论一</td><td colspan="2">RMQ。先对数列进行变换，变换后数列中的 A[I]代表与其相同的左面的节点数，例如，(-1 -1 1 1 1 1 3 10 10 10）==>（1 2 1 2 3 4 1 1 2 3)，然后定义 Sum[I]代表与它相同的总共有多少个，即（2 2 4 4 4 4 1 3 3 3)。然后，我们对 A 数组进行 RMQ 预处理。对于每一个询问，Ans=Max{该区间最左面相同的数的个数，该区间最右面相同的数的个数，该区间中间相同的数的个数}=Max{ Sum[L]-A[L]+1 ， A[R] ， Rmq[中间区间]}</td></tr>
<tr><td>算法讨论二</td><td colspan="2">线段树。也是分段思想，在线段树的结点内设 5 个变量 l、r、mx、lf、rf，[l,r]表示该结点的区间范围，lf 和 rf 分别表示元素 a[l]和 a[r]在区间内的出现频率，mx 表示区间内的最高出现频率。假设区间[x,y]和[y+1,z]均被询问[i,j]覆盖，则可以分情况讨论区间[x,z]的 mx 值：若 a[y]==a[y+1]，则 mx[x,y]=max{mx[x,y],mx[y+1,z],rf[x,y]+lf[y+1,z]}，否则 mx[x,y]=max{mx[x,y],mx[y+1,z]}。</td></tr>
<tr><td>其它</td><td colspan="2">时间复杂度为 O（N+M）。</td></tr>
<tr><td>PKU 2823</td><td>Sliding Window</td><td>有 N 个数，每次从左至右选 M 个数，即[1..M]，[2..M+1]，...，[M-N+1..N]，选取每个区间的最大值和最小值。</td></tr>
<tr><td>算法讨论</td><td colspan="2">建立两个单调队列，一个维护最大值，一个维护最小值，下面以最小值的单调队列来写做法。
先初始化单调队列为 A[1]..A[M]，从小到大排序，之后每次将区间向后移动一格。
当移动到 J 位置时，从单调队列中二分查找到第一个比不小于 A[J]的数 A[I]，由于（I=A[J])，显然 A[I]此时已经没有用了，所以我们用 A[J]覆盖它。若没有不小于 A[I]的数，则将其作为新元素加入队尾。
取这个区间的最小值，从单调队列的队首取出第一个元素，若其在区间范围内则它就是最小值，否则将队首踢出队列，重复此操作。
所以整体时间复杂度为 O（NLogN+2N）。</td></tr>
<tr><td>其它</td><td colspan="2">后来发现，其实，本题完全没必要二分单调队列。
假设维护的是最小值的队列，当移动到 J 位置时，从队尾往前扫，如果队尾某元素大于 A[J]，则出队。
这样，每个元素至多出队一次，入队一次，整体复杂度为 O（N）。</td></tr>
<tr><td>SGU 142</td><td>Keyword</td><td>给出一个长为 N（N<=500000）的字符串 s，仅由字符 a 和 b 组成。要求一个字符串 u，u 也是仅由 a 或 b 构成，但并不是 s 的一个连续子串，同时要求 u 的长度最短。</td></tr>
<tr><td>算法讨论</td><td colspan="2">2^20 已经超过了 N 的范围，所以一个长为 N 的串至多包含 20 个所有本质不同的子串。
所以，我们二分枚举长度，再枚举这个串的具体组成即可。
接下来的问题就是，已枚举出一个子串，如何判定其是否出现在原串中。
我们可以在每次二分出长度 Len 后，把原串中所有长为 Len 的子串加入 Trie 树中。
这样，就可以在 O（1）的时间内判定枚举出的子串是否出现在原串中了。</td></tr>
</table>

<table>
<tr><td>其它</td><td colspan="2">建立 Trie 树的复杂度是 O（NL），即 O（NLogN）。
枚举串的复杂度是 O（2^L），即 O（N）。
二分的复杂度是 O（LogL），忽略不计。
所以整体时间复杂度是 O（NLogN）。</td></tr>
<tr><td>SGU 148</td><td>B-Station</td><td>在一个地下室内有 N（N<=15000）层：第 1 层最靠近地面，而第 N 层就是最底层。每一层都可以蓄水：第 i 层最多可以蓄水 Li 个单位。一开始每一层都有水量为 Wi 个单位。如果某一层的存水量大于其最大蓄水量，这些水（包括原来的水）就会都流入下一层。除此之外，对每一层，还可以手动的将某一层中的水都放入下一层：但这需要 Pi 个单位的钱。</td></tr>
<tr><td>算
法
讨
论</td><td colspan="2">我们枚举需要手动放水的最高层 I，这样可以在 O（N）时间内求出，以 I..N 的花费。
由于总共要枚举 N 次，所以总复杂度为 O（N^2）。

下面进行优化。
设需要手动放水的最高层为 I，Sum[I]为前 I 层总水量，则花费为
P[I]+Sum{P[J]} （J>I，Sum[J]-Sum[I-1]<=L[J]）。
下面进行参数分离，我们将 Sum[J]-Sum[I-1]<=L[J]变形为
Sum[J]-L[J]<=Sum[I-1]，即 C[J]=Sum[J]-L[J]，C[J]<=Sum[I-1]。
我们可以发现，C[J]为定值，所以我们维护一颗平衡树，从后向前推。
当推到第 I 层时，从平衡树中找出比 Sum[I-1]小的元素的总花费，然后加上 P[I]就是第 I 层的花费，更新 Ans，把 C[I]加入平衡树，继续向前推。
这样，总共需要推 N 次，每次需要利用平衡树进行插入和查找，总复杂度为 O（NLogN）。</td></tr>
<tr><td>其它</td><td colspan="2">参数分离是个很强大的东西。</td></tr>
<tr><td>SGU 155</td><td>Cartesian Tree</td><td>给出一列 N 个二元组(ai, ki)，要求为这些二元组建立一颗笛卡儿树，所谓笛卡尔树，就是如果关注树上每一个节点的 ai 元素，那么这是一个严格的最小二叉堆：根节点的 ai 值最小；如果关注 ki 元素，那么这是一颗严格的排序二叉树：左孩子较小，右孩子较大。</td></tr>
<tr><td>算
法
讨
论</td><td colspan="2">此题要求建立一个笛卡尔树，实际上可以理解为建立一个 Treap，只不过不用随机堆的关键字。
但是，此题若用 Treap 写，可能会退化成 O（N^2），导致 TLE。（或许打乱顺序进行 Treap 可以过）
所以我们寻找一种新的算法。
由于 K 元素已知，我们对其进行 Qsort，之后 O（N）的建树，这样可以确保仅用排序的 O（NLogN）的复杂度，我们就可以建立一颗非常平衡的平衡树。
之后，我们在这颗平衡树上利用左旋右旋进行调整，使其满足堆性质，由于一个点至多下降 LogN 层，所以复杂度也是 O（NLogN）。
所以，整体复杂度也是 O（NLogN）。</td></tr>
<tr><td>其它</td><td colspan="2">另外，此题还有 O（N）的算法，见楼天成的 SGU 表格。</td></tr>
</table>

<table>
<tr><td>PKU 2482</td><td>Stars in Your Window</td><td>在一个灰常大的坐标系里，有 N（N<=10000）个点，每个点均有一个权值，问你用 W*H（W,H<=10^9）覆盖这些点，最多能覆盖的权值为多少。</td></tr>
<tr><td>算法讨论</td><td colspan="2">先对 N 个点的坐标进行离散化，横向扫描，对于纵向建立线段树进行维护。
有如下两种维护方式：
1）把每个点 Y[I]，看做从 Y[I]开始向前数 H 个单位的总权值，所以我们只用插入（Y[I],Y[I]+H-1）这样一条线段进行维护即可。
2）把每个点拆做两个点，Y[I]带着权值 D[I]，Y[I]+H 带着权值-D[I]，每次向线段树中插入这两个点，这样题目就变成了求前缀和最大。
两种方法都可以 AC 这个题，其本质都是一样的。但是第二个更容易的可以用树状数组实现，时间上会更优。</td></tr>
<tr><td>其它</td><td colspan="2">时间复杂度为 O（NLogN）。</td></tr>
<tr><td>PKU 2892</td><td>Tunnel Warfare</td><td>给出直线上一系列的村庄，如果相邻村庄都没有被破坏，则两村庄是连接的，题目给出一系列的破坏操作，对指定号码的村庄进行破坏，还有一系列的询问操作，询问与指定号码的村庄直接相连或间接相连的村庄有几个，还有一个修复操作，是对最后破坏的村庄进行修复。</td></tr>
<tr><td>算法讨论</td><td colspan="2">对于破坏操作，则在平衡树中插入关键字为 X 的节点。
对于查询操作，则在平衡树种查找第一个不小于 X 的节点和第一个不大于 X 的节点。
对于修复操作，则在平衡树种删除关键字为 X 的节点。
当然，此题也可以用线段树来做。</td></tr>
<tr><td>其它</td><td colspan="2">时间复杂度为 O（NLogN）。</td></tr>
<tr><td>PKU 2763</td><td>Housewife Wind</td><td>N 个点的无根树，有 Q 次询问。0 号询问，问从上一次到达的点到现在给定的点需要走的距离。1 号询问，修改树上某条边的权值。对于每个 0 号询问进行输出。</td></tr>
<tr><td>算法讨论</td><td colspan="2">假设没有 1 号询问，则是一个经典问题，求树上（X，Y）两点间的位置。
定义 F［I］代表 I 节点到根节点的距离，所以 Dis[X，Y]＝F［X］+F［Y］－2*Lca［X，Y］。
由于 N 高达 10^5，所以只能将 LCA 转成 RMQ 来求解。（Lca 如何转 Rmq？）
这样，我们就可以用 O（NLogN）进行预处理，O（QLogN）处理所有询问。
现在我们加入 1 号询问，更改某条边的权值。
设边是（X，Y），且 X 是 Y 的父亲，则更改该边只会影响以 Y 为根子树上的 F 数组中的值。
在刚才遍历的的数组中，Y 第一个出现的位置到最后一个出现的位置之间，正好是以 Y 为根的子树。所以，我们可以用线段树完成这个操作。</td></tr>
<tr><td>其它</td><td colspan="2">时间复杂度为 O（QLogN+NLogN）。</td></tr>
</table>

图论相关：

<table>
<tr><th>题目来源</th><th>题目名称</th><th>题目大意</th></tr>
<tr><td>PKU 3164</td><td>Command Network</td><td>最小树形图经典题。</td></tr>
<tr><td>算
法
讨
论</td><td colspan="2">很标准的最小树形图模型，此算法分 4 个步骤进行。
1）求每个点入弧的最小值。
2）若有环则跳到第三步。
3）将环缩成一个点，更新所有点到这个点的权值，更新这个点到其他所有点的权。
4）输出答案。
具体的参考《图论的算法与程序设计》和
http://hi.baidu.com/bin183/blog/item/45c37950ece4475f1138c273.html。</td></tr>
<tr><td>其它</td><td colspan="2">时间复杂度为 O（N^3）。
一个扩展问题，求不指定根的最小树形图。其实也很简单，我们设立一个新结点当根，根到所有点的距离设为相同的权，这个权要比所有权的和还大，然后就求这个图的最小树形图就可以了。</td></tr>
<tr><td>PKU 1144</td><td>Network</td><td>求一个图的割点。</td></tr>
<tr><td>算
法
讨
论</td><td colspan="2">若该点是割点，则满足：
1）U 不是根节点，且 U 的任一后代 S 不存在到 U 祖先之间的后向边；
2）U 是根节点，且 U 的有效儿子不不止 1 个，什么叫做有效儿子，需要自己慢慢的体会。
根据以上思想，在 DFS 树我们引入标号函数 Low[U]，代表 U 节点可追溯到的最早节点。当 Low[U]<=Low[S]时，若 U 不为根则该点为割点，若 U 为根则累加有效儿子数。</td></tr>
<tr><td>其它</td><td colspan="2">时间复杂度为 O（N）。</td></tr>
<tr><td>PKU 2186</td><td>Popular Cows</td><td>有 N 头牛，若 a 仰慕 b 且 b 仰慕 c，则 a 也仰慕 c。求有多少牛，被所有牛（除自己）都仰慕。</td></tr>
<tr><td>算
法
讨
论</td><td colspan="2">这个题，咋看像是传递闭包，但是 N 的范围太大，无法承受。所以，我们进一步抽象，转换成求强连通分量。将每个强连通分量求出来之后，缩成点，若该点的出度为 0，且只有一个出度为 0 的强连通分量，则强连通分量里点的数量就是答案，反之就无解，输出 0。</td></tr>
<tr><td>其它</td><td colspan="2">时间复杂度为 O（N）。</td></tr>
<tr><td>PKU 1236</td><td>Network of Schools</td><td>N 个高校之间有一些单向的网络链接（N<100），当发布一个软件时，学校 i 收到软件时，它可以将软件发送给所有它链接到的学校。现在要求发布一款软件，最少需要发给多少个学校，使得所有学校都可以收到软件（问题 A）。最少需要添加多少条单向</td></tr>
</table>

		网络链接，可以使得将软件任意发给一个学校，使得所有学校都可以收到（问题 B）。
算法讨论	第一问，明显是求最小点基的个数，关于最小点基的定义参考各个算法书吧。第二问，我们挖掘实质可以发现，将原图的强连通分量缩点后，设入度为 0 有 t1 个，出度为 0 的点有 t2 个，则取 t1 和 t2 的最大值即可，具体的证明应该很显而易见。还有，其实，t1 就是最小点基的个数，这也可以证明。	
其它	时间复杂度为 O（N）。	
URAL 1129	Door painting	幼儿园里有许多房间，之间是走廊和门。一个装修计划即将执行。 这些门允许涂上明快的颜色：绿色和黄色。园长希望满足以下条件：任意一扇门的各个面必须不同颜色。每一个房间绿色门的数量，与黄色门的数量之差最多为 1。给定园长的计划，请提出你的安排。
算法讨论	先构图，每个屋子和相连的屋子连一条边。如果每个点的度为偶数，则我们可以按欧拉回路的路径走，从屋子走到走廊里标为 G，从走廊走到屋子里标记为 Y。这样，就一定不会相同，且在数量上 G=Y。若图的某些点的度不是偶数，由于图的特殊性，这些点一定是偶数个，所以我们随意选两个度为奇数的点连一条边，直到没有度为奇数的点。这样，就构成了欧拉回路，且 G 和 Y 的数量相差不会超过 1。	
其它	时间复杂度为 O（N+M）。	
PKU 3281	Dining	有 n 个牛，有 f 种食物和 d 种饮料，每个牛喜欢一个或多个食物和饮料，但是所有的食物和饮料每种都只有一个，问最多可以满足多少头牛的需要。
算法讨论	主要是构图，吃的 X、喝的 Y、人 P，三种元素，分别拆点，之间连一条容量为 1 的弧。然后 S 向 X 连容量为 1 的弧，X'向 P 连容量为 1 的弧，P'向 Y 连容量为 1 的弧，Y'向 T 连容量为 1 的弧。求最大流即可。	
其它	时间复杂度为 O（N^3）。	
PKU 1149	PIGS	有 M 个猪圈（M ≤ 1000），每个猪圈里初始时有若干头猪。 一开始所有猪圈都是关闭的。 依次来了 N 个顾客（N ≤ 100），每个顾客分别会打开指定的几个猪圈，从中买若干头猪。 每个顾客分别都有他能够买的数量的上限。 每个顾客走后，他打开的那些猪圈中的猪，都可以被任意地调换到其它开着的猪圈里，然后所有猪圈重新关上。 问总共最多能卖出多少头猪。
算法讨论	关键在于构图。先建立一个虚拟源点 s，和虚拟汇点 t。假设某个猪圈有 n 个人有钥匙，依次为 p1,p2,...,pn，则 s 和 p1 之间连一条容量为猪圈猪数量的弧，p1 和 p2 之间连一条容量为 maxlongint 的弧，...，pn-1 和 pn 之间连一条容量为 maxlongint 的弧，之后每个 pi 和 t 连一条容量为要买的猪的数量的弧。这	

	样，整个图就至多有 102 个结点。随便找一个最大流算法都能 A 掉。当然，我用的是邻接矩阵式的 Hlpp，因为方便边的合并。更加详细的讲解：http://imlazy.ycool.com/post.2059102.html	
其它	时间复杂度为 O（N^3）。	
ZJU 2676	Network Wars	给出一个带权无向图，每条边 e 有一个权。求将点和点 t 分开的一个边割集，使得该割集的平均边权最小。
算法讨论	根据分数规划，类似 P=Wx/Cx 的形式（W 代表边权，C 为 1，x 为由 0 或 1 的 M 维向量，若 xi 为 1 则代表取第 I 条边，反之不取），我们可以写成，0=(W-PC)x，设 G=(W-P'C)x，G=min{(W-P')*x}，所以我们将原图中的每条边的权改为 W-P'，二分枚举 P'，则该问题转换成了判定性问题，直到 G 为 0 时，P=P'，此时我们就可以找出最小割来了。具体的看胡伯涛 2007 年的论文。	
其它	时间复杂度为 O（N^3*LogAns）。	
PKU 2125	Destroying The Graph	N 个点 M 条边的有向图，给出如下两种操作。 删除点 I 的所有出边，代价是 AI。 删除点 J 的所有入边，代价是 BI。 求最后删除图中所有的边的最小代价。
算法讨论	我们将两种操作各看成一个点，x 和 y，一条有向边则看成连接两个操作的边，这样问题就转换成了最小点权覆盖问题。然后，起点 s 向 x 连接一条弧，容量为删除出弧的权，y 向终点 t 连接一条弧，容量为删除入弧的权值。然后 x 和 y 若有关系，则连一条容量为 maxlongint 的弧。这样，我们就将原问题进一步转化了成了最小割切模型。然后利用最大流求出最小割即可。	
其它	时间复杂度为 O（N^3）。	
PKU 2396	Budget	给定一个矩阵每行，每列的和，和各个元素的限制条件(>, =, <)，求出一个满足这各种限制的矩阵。
算法讨论	我们将两种操作各看成一个点，x 和 y，一条有向边则看成连接两个操作的边，这样问题就转换成了最小点权覆盖问题。然后，起点 s 向 x 连接一条弧，容量为删除出弧的权，y 向终点 t 连接一条弧，容量为删除入弧的权值。然后 x 和 y 若有关系，则连一条容量为 maxlongint 的弧。这样，我们就将原问题进一步转化了成了最小割切模型。然后利用最大流求出最小割即可。	
其它	时间复杂度为 O（N^3）。	
SGU 242	Student's Morning	容量有上下界的可行流。
算法讨论	这个题除了构图不一样和输出不一样，其余的和 PKU2396 是一模一样。PKU2396 的题解是本篇文章的上一篇，详细的题解在那里，这里只把构图说一下。第一次构图，新建源 s 汇 t，将 s 向人连一条上界容量为 1 的弧，人向学校连一条上界容量为 1 的弧，学校向汇连一条下界为 2 上界无限的弧。然后根据此图构建附加网络图即可。	
其它	时间复杂度为 O（N^3）。	

PKU 2516	Minimum Cost	有 N 个顾客，M 个小贩，K 种商品，给出 N 个顾客对不同商品的需求量，M 个小贩对不同商品的供给量，以及不同顾客从不同小贩处买不同商品的价格是多少，求最小费用。
算法讨论	此题的构图，我想了半天，后来才发现很简单。由于有 K 种商品，但是每一种商品的图其实是独立的。所以，我们可以构 K 次二分图。每次，新增源点 s 和汇点 t，s 向提供者连弧，容量为给定值，费用为 0；提供者向购买者连弧，容量为 maxlongint，费用为给定值；购买者向 t 连弧，容量为给定值，费用为 0。显然，这个图我们求最小费用最大流即可，当然也可以用 KM 求最佳匹配。由于要求 K 次最小费用最大流，所以时间复杂度要乘以个 k。	
其它	此题的关键在于，把部分脱离出整体，逐个求解。	
HDOJ 1914	The Stable Marriage Problem	稳定婚姻系统模版。
算法讨论	赤裸裸的稳定婚姻系统，直接套用模板即可。关于稳定婚姻系统的讲解，见这里 http://hi.baidu.com/leokan/blog/item/4f9b04f719993025730eecef.html。	
其它	实用性不是很大。	
PKU 1364	King	有一个长度为 N 的数列，有 M 个约束条件，每个约束条件表示从 Ai-Ai+Bi 这一段的数字之和大于（或小于）Ci。求出是否存在这样的串满足所有的约束条件的数列。
算法讨论	此题一看就是差分约束题，但是给的是大于和小于号，由于是整数，所以加 1 和减 1 后就转换成了大于等于和小于等于。然后套用经典的差分约束解决即可。差分约束的讲解见这里，http://hi.baidu.com/lxc_0601/blog/item/29d27201b4c7c70b7aec2c5d.html。	
其它	时间复杂度为 O（NM）。	
SGU 176	Flow constructio	容量有上下界的最小流。
算法讨论	SGU242 求解的是容量有上下界的可行流，还是套用那个模版。不过，这里求最小流时，我们需要二分答案 Ans，在附加网络中将 Ed 到 St 的流量从固定的 Maxlongint 更改为 Ans，如果满流则缩小流量，否则增大流量。	
其它	时间复杂度为 O（N^2*M*LogAns）。	
PKU 2513	Colored Sticks	给定一系列的颜色，问是否存在一条欧拉路。
算法讨论	这道题很明显就是判断无向图中是否存在欧拉路，判断条件是：1）有且只有两个度为奇数的点 2）图是连通的。 由于点是字符串，我们可以利用字母树将其转换成一个颜色序号。这样，对颜色序号操作即可。判断图是否连通，我们可以利用并查集。若最后所有点在一个集合，则图是连通的。	

其它	X	
PKU 3565	Ants	平面上给你 N 个 A 点和 N 个 B 点，要求 A 和 B 一一匹配且不能出现相交的情况。。
算法讨论一	二分图匹配。在原有的匈牙利算法基础上，每一次判断是否和已匹配的边相交，若相交则无法到达该点。然后，按原来的匈牙利匹配即可。时间复杂度 O(N^2*M)，跑了 2s 整。	
算法讨论二	随机调整法。初始时随机匹配，然后查看是否有 AB 和 CD 相交，若存在则交换顶点，即变成 AC 和 BD 两条边。这样不断的调整，直到不再存在相交的边为止。时间复杂度 O(N^4)，但是远远小于 O(N^4)，跑了 32ms。	
其它	随机调整是个很强大的算法，不过考试时在传统题上慎用。	
PKU 1062	昂贵的聘礼	N 个点，向起点连一条权值为 T 的弧，然后根据描述将图连出来，求起点到点 1 的最短路。不过，每个点有一个等级限制，这条最短路上的最大等级和最小等级差不能超过 M。
算法讨论	我们先枚举每个点，以它的等级作为最大等级 Max。然后每次扩展最短路时，最短路上的点的等级必须小于 Max 且相差不能超过 M。接下来就是 SPFA 求最短路了。	
其它	一般这类有限制的图论题，都需要枚举个什么东西取消限制。	
UVA 10246	Asterix and Obelix	N 个点，M 条路，每个点有一个权值，每条路有一个权值，A 到 B 的距离定义为，A 到 B 的路径长度+路径中点权最大值。输出 A 到 B 的距离最小值。
算法讨论	由于询问很多，我们先进行预处理。枚举中间节点 X 为最大权值点，将所有权值比它大的删掉，求 X 到所有剩余点的最短路。对于每询问 A,B，我们枚举中间节点 X，结果就是 Min{Dis[X,A]+Dis[X,B]+Cos[X]}。	
其它	时间复杂度为 O（N^2*LogN）。	
PKU 3635	Full Tank?	有 N 个点，M 条边，一辆车的汽油容量为 C，每个点都可以加油，费用为 Ci，每走一单位距离消耗一单位的油，求从起点到终最少需要花多少钱。共有 100 组询问。
算法讨论	不考虑汽油，则是一个最短路问题。现在，我们来考虑汽油，先来寻找一种状态，F[I,J]代表在 I 节点，拥有 J 个汽油的最小价值，决策就是，要么加油，要么走去下一个节点。换句话来说，我们就是将 N 个点拆成了 N*（C+1）个点，已知道 F[St,0]=0，求 F[Ed,0]。很显然，是一个变形的 Dijkstra_Heap。	
其它	时间复杂度为 O（Q*NLogN）。	

SPOJ 2272	Crossing the Desert	有 N 个点，起点为 1，终点为 N，一个人穿越沙漠，可以携带水和食物，总和不能超过 Limit 单位，每走一单位路程消耗 1 单位水和食物，水在任何节点都免费供应，而食物只在 1 号节点供应，不过其他节点可以储存食物，问从起点走到终点最少需要多少食物。
算法讨论	这个题要用到最短路的永久性标号思想。假设已经知道 J 号节点所储存的食物，设为 D[J]，再设 I 走到 J 的距离为 Dist，我们来从 J 倒推 I。为了再 J 号节点储存 D[J]单位的食物，我们需要在 I、J 之间往返。从 I 走到 J，至多可以携带 Limit-Dist 单位的食物，走过去消耗 Dist，走回来又消耗 Dist，所以一次往返所能储存的食物最大值为 Limit-3*Dist。但是，最后一次是不用再走回去的，所以最后一次行走所能储存的食物最大值是 Limit-2*Dist。设真正的往返次数为 M，所以，Limit-2*Dist+M*（Limit-3*Dist）=D[j]，所以我们可以求得往返的次数 M。这样，我们就可以确定 D[I]=Min{D[J]+（2*M+1）*Dist}。回过头来看，我们已知的是 D[终点]=0，求 D[起点]，而 D[I]的方程，是个典型的最短路方程，所以问题就转换成了求最短路问题。当然这中间要处理特殊情况：1）从 I 无法走到 J，即 Limit<2*Dist。2）无法在 I 到 J 间往返，也就是说 I 到 J 间往返无法产生多余的食物，即 Limit-3*Dist<=0 时。3）从 I 走到 J 不用往返，即 D[j]-Limit-2*Dist<=0。	
其它	时间复杂度为 O（N^2）。	
PKU 3463	Sightseeing	N 个点，M 条边。求 St 到 Ed 的最短路和最短路+1 的路径条数。
算法讨论	定义 F[I,0]代表到达 I 节点的最短路，F[I,1]代表到达 I 节点的次短路。其实，可以把这两个状态当做两个节点，分别加入到堆中。在用 Dijkstra 求最短路的时候，先用 Dis 更新 F[I,0]：1）Dis=F[I,0]，累加 count。2）Dis < F[I,0]，更新次短路为最短路，更新最短路为当前最新值。3）Dis > F[I,0]或者 F[I,0]已经不在堆中，更新 F[I,1]。这样，最后 F[Ed,1]-1=F[Ed,0]则输出两者的 Count 和，否则输出最短路的 Count。	
其它	时间复杂度为 O（NLogN）。	
URAL 1162	Currency Exchange	有 N 种货币，M 种兑换关系，问经过兑换后是否可以增加原资本。
算法讨论	这里之所以加引号，是因为并非真正意义上的正权回路。所以，我们可以利用 SPFA 来求解，跟求负权回路一样，进行松弛之后，若某个点入队超过 N 次，则必定产生了“正权回路”。	
其它	时间复杂度为 O（N^2）。	
PKU 1751	Highways	给你 N 个坐标系中的点，求一颗最小生成树，有的边已经给出。

<table>
<tr><td>算法讨论</td><td colspan="2">若不管已经给出的边，则是标准的生成树。但是，若必须包含给出的边，怎样求？其实也很简单，我们设给定边的两点距离为 0，则在求最小生成树的时候，必定会选这条边。然后控制输出即可。</td></tr>
<tr><td>其它</td><td colspan="2">时间复杂度为 O（N^2）。</td></tr>
<tr><td>PKU 1679</td><td>The Unique</td><td>N 个点，M 条边，求最小生成树，若最小生成树不唯一，则输出 No。</td></tr>
<tr><td>算法讨论</td><td colspan="2">这个题用 Prim 来做是很简单的。每次拿出一个距离生成树最近的点加入生成树，如果该点的距离不是由一个点更新来的，则必然由多解。否则只有一个解。</td></tr>
<tr><td>其它</td><td colspan="2">时间复杂度为 O（NLogN）。</td></tr>
<tr><td>PKU 3522</td><td>Slim Span</td><td>N 个点，M 条边，求一颗生成树，使得生成树上最大权值的边减最小权值的边最小。</td></tr>
<tr><td>算法讨论</td><td colspan="2">枚举一条边，使其作为生成树上最小的边。然后，对整个图比它大的边进行 Kruskal，最后取一个 Min{Max-Min}即可。</td></tr>
<tr><td>其它</td><td colspan="2">时间复杂度为 O（M^2*LogM），实际上远远比这个要小。</td></tr>
<tr><td>PKU 1639</td><td>Picnic Planning</td><td>N 个点，M 条边，求一颗最小生成树，但是给定一个点 T 和一个限制 K，在生成树中 T 的度不得超过 K。</td></tr>
<tr><td>算法讨论</td><td colspan="2">设共有 N+1 个点，指定点为 V0，限制为 K。

1）抛开 V0，求出 V1..Vn 的最小生成树。由于 V1..Vn 可能不连通，所以这里的最小生成树指的是各个连通块的最小生成树。
2）可以证明，当得出最小生成树后，未在最小生成树内的边不会再被用到，删除即可。但是删边的复杂度太高，所以，不如将新边建立成一个新的集合。
3）下面加入 V0，为了保证图是连通的，对于每个连通块连一条边权最小的边。设 H[I]代表 V0 的度为 I 时的最优值，假设共有 P 个连通块，显然我们这一步中求出了 H[P]。
4）由 H[I]推 H[I+1]，枚举未直接连向 V0 的所有点，使其向 V0 连边，设边权为 A。此时，生成树中会出现环，所以我们在环中找出一个具有最大权值的边，设其权值为 B。则 H[I+1]=H[I]+Min{A-B}，若 H[I+1]不比 H[I]更优，则没有必要再推 H[I+2]，直接 Break。
5）上述算法的瓶颈在于第 4 步，如果快速找出 B。如果枚举着找，则复杂度会增到 O（N^3），所以我们需要进行优化。设 Max[I]代表，V0 到 I 这条路径上最大权值，由于是生成树，所以路径唯一。这个可以在第三步的时候用 DFS 遍历预处理出来，之后每次调用 Max[I]即可。当得到 Min{A-B}之后，我们需要删边，则我们再从取得 B 值的节点进行 DFS 遍历即可。至此，复杂度降低到了 O（N^2）。</td></tr>
</table>

其它	时间复杂度为 O（N^2），题目中还有一个有用的知识点就是动态维护固定根的最小生成树。	
PKU 2728	Desert King （Un AC）	N 个点，M 条边，每条边有两个权值 D 和 H，求得一颗生成树，使得 Sum{H}/Sum{D}最小。
算法讨论	借鉴 ZJU 2676 这个题，根据 01 分数规划思想，P=Hx/Dx，即 0=（H-PD）x，设 G=（H-P'D）x，二分枚举 P'，将每条边的权值设为 H-P'D，然后二分答案，根据单调性缩小区间求解。	
其它	时间复杂度为 O（N^2*LogAns）。 一样的代码，C++可以过，Pascal 就 TLE。	
PKU 2553	The Bottom of a Graph	N 个点，M 条边，求一些点，满足这个点可到达的点均能到达该点。
算法讨论	由于满足互相均能到达，显然是求一个强连通分量，且该强连通分量不能有边连向不属于该强连通分量的点。套用经典模块即可。	
其它	时间复杂度为 O（N）。	
PKU 3694	Network	N 个点形成一个无向图，我们将对其进行 K 个操作，每个操作的时候连一次 AB 的边，并且对于每次操作，输出剩下的割边数有多少条。
算法讨论	引用某牛的一句话，“联想下，之前有一个 poj3177 中我们可以知道一个道理，就是树中的边全是割边，如果连接了树的某两个点的话那么形成的这个环内的边就全部都不是割边了。” 所以，我们得到了一个算法，将图缩点构造成一颗树，每次为 A,B 添边，则将 A..Lca(A,B)和 B..Lca(A,B)路径上的点压缩成一个点，或者说合并成一个集合，利用并查集处理。由于需要记录路径，所以，貌似只能用朴素的 LCA。另外，时间复杂度很大程度的取决于所选的根是谁。	
其它	时间复杂度为 O（N+Q*K*LogN），K 为求 LCA 的时间复杂度，LogN 为并查集的复杂度。	
PKU 2762	Going from u to v or from v to u?	N 个点 M 条边的有向图，若对于其中任意两点 U,V，存在 U 到 V 得路径，或存在 V 到 U 得路径，则输出 Yes，反之 No。
算法讨论	先简化问题，将所有强连通分量缩成一个点，则此时问题就是判断 DAG 图是否满足题目中所给定的性质。假设缩点后，剩下 T 个点。显然，若存在一条最长链使得该链上包含了所有 T 个点则是满足性质的，否则不满足。 证明应该很容易。 所以，必然有一条最长链包含所有 T 个节点，且最长链的起点一定是缩点后入度为 0 的点。	
其它	时间复杂度为 O（N+M）。	
PKU 3687	Labeling Balls	N 个点，M 个关系，每个关系 A,B 代表 A 的重量比 B 轻，且每个点重量都不一样。求最后字典序最小的重量排列，这里的重量排列指的是 1 的重量，2

		的重量...N 的重量。
算法讨论	很明显的拓扑排序，但是若 A 连边向 B，每次取度为 0 且最小的点当做最轻的，则是一个错误的贪心算法。 反例很明显，例如 N=4，M=2，(4,1)，(3,2)，按照上述贪心得出的解是(4,2,1,3)，而正确答案是(2,4,3,1)。 正确思想是，反向建图，即 B 连边向 A，每次取度为 0 且最大的点当做最重的。 证明见：http://imlazy.ycool.com/post.2144071.html。（个人的粗略证明：因为求的重量序列是并不是第 1 重到第 N 重进行排列，而是 1 重多少到 N 重多少的排列，所以正向贪心就是错误的。还有，正向贪心保证的是越往前的越小，反向贪心，保证了越往后的越大。但根据字典序的比较方式，显然反向贪心比正向贪心更优。）	
其它	时间复杂度为 O（N^2）。	
PKU 2488	A Knight's Journey	有一个 N*M 的棋盘，从（1，1）出发，按国际象棋马的走法，走完整个棋盘且不能重复走，求一个字典序最小的方案。
算法讨论	范围是 26*26 的，如果数据真的这么给，基本上 PKU 上 99%会 TLE。由于数据很弱，随便写一个 DFS 就能过了。	
其它	时间复杂度不好估计。	
SGU 101	Domino	N 个骨牌，每个骨牌的有两个数字，求一种连接方式，使得相邻的两个骨牌，数字相同。
算法讨论	经典的求欧拉路问题，套用经典模版。 思想流程： 1、在图中任意找一个回路； 2、将图中属于回路的边删除； 3、在残留图的各极大连通子图中分别寻找欧拉回路； 4、将各极大连通子图的欧拉回路合并到中得到图的欧拉回路。 解法流程： 1）读入构建边表、预处理。 2）无解情况判断，利用并查集和度的关系。 3）若有度为奇数的则从它开始寻找欧拉路，否则任意找一个度不为 0 的点。 4）设置 sum=边数。对于每一个点的所连向的边，直接走。 5）走完之后，回溯时递减一个 sum，把当前边加入输出集合。具体的思想，见程序，细细体会。	
其它	时间复杂度为 O（M）。	

PKU 1386	Play on Words	有 N 个盘子，每个盘子上写着一个仅由小写字母组成的英文单词。你需要给这些盘子安排一个合适的顺序，使得相邻两个盘子中，前一个盘子上面单词的末字母等于后一个盘子上面单词的首字母。请你编写一个程序，判断是否能达到这一要求。
算法讨论	经典问题，判断图中是否存在欧拉路或欧拉回路。	
其它	时间复杂度为 O（N)。	
PKU 2404	Jogging Trails	N 个点 M 条边的无向图，任意点都可以作为起点，要求访问完所有的边然后回到起点，边可以重复走，求最小值。
算法讨论	这是个经典问题，中国邮路问题 1 号。 问题可以抽象成，增加一些边，构成欧拉回路图，使得增加的边的总代价最小。 先排除无解的情况（本题没有)。 然后，将度为奇数的点找出，必然为 2*T 个。在他们之间连 T 条边，则可以构成欧拉回路。 所以，我们可以构建新的图，以这 T 个点作为图中的顶点，任意两点之间连一条边，权值为原图中的两点间的最短路径。 所以，问题进一步转化成了求无向完全图的最小权匹配。 据说有很快的算法可以算出最小权值匹配，又据说可以用 KM 算法直接 N^3 求出。 但是，或许是我构图有问题吧，用最佳匹配最后一组数据一直 WA。。。 所以，我枚举了两边的点，然后进行最佳匹配。 当然还有一个比较简单的算法，即状态压缩动态规划。	
其它	时间复杂度为 O（N^3*C(N,N/2))。	
PKU 3177	Redundant Paths	N 个点，M 条边，问最少添加几条边使得图中任意两点互相到达且至少有两条完全不一样的路径。
算法讨论	很显然，若题目中存在桥，则不可能有两条完全不一样的路径，反之则一定有。所以，我们的任务就是添加边使得图中不存在桥。先将双连通分量缩成一个点，则连接任意两个双连通分量的边就是桥。设度为 1 的点有 T 个，则添加（T+1） Div 2 条边就可以使得图中不存在桥。 有人说至少在树上添加(leaf+1)/2 条边，就能使树达到边二连通，所以结果就是(leaf+1)/2。（来自 Byvoid 的 Blog）	
其它	时间复杂度为 O（N)。	
PKU 3249	Test for Job	N 个点，M 条边，每个点有一个权值，求从任意入度为 0 的点走到任意出度为 0 的点权值最大。

算法讨论	按照拓扑关系进行动态规划。F[I]代表在 I 节点取得的最大值，动态转移方程 F[I]:=Max{F[J]}+A[I]。	
其它	时间复杂度为 O（N）。	
PKU 3592	Instantaneous Transference	一个 N*M 的矩阵，0..9 代表这个格子的价值，*代表这个格子是传送门（当然也可以不传送，价值为 0），#代表这个格子不可达。每个格子，只能向其右面或下面走一个格子。求从左上角开始，走到任意位置时获得的最大价值。
算法讨论	不考虑传送门，则是一个超级简单的动态规划，或者说记忆化搜索吧，和 PKU 3249 类似。现在，加入传送门，所以这个图中可能会出现强连通分量，就破坏了动态规划的无后效性。但是，我们深入研究可以发现，由于强连通分量中任意两点均可达，所以我们可以将它们缩成一个点，价值就是全部点的价值和。所以，就又转换成了一开始的简单动态规划问题。	
其它	时间复杂度为 O（N）。	
PKU 1201	Intervals	有 N 个闭区间，给你每个闭区间的左边界，右边界，和区间内含有的数。求出整个数列至少有多少个数。例如，[1,3]=2，[2,4]=2，则整个区间只要含有 2、3，则满足。
算法讨论	我们大胆假设，设 S[I]代表前 I 个数中含有的数的个数。 则我们可以将区间关系转换成不等式， 1：Sb-Sa-1>=C，即 Sa-1<=Sb-C。 对于自然数 x，有 0 < = S(I) - S(I-1) <= 1，即　2：Si-Si-1>=0，即 Si-1<=Si-0。 3: Si-1-Si>=-1，即 Si<=Si-1+1。 根据差分约束，当满足 D[V]<=D[U]+W[U,V]时，我们向 U 到 V 连一条 W[U,V]的边，求出最短路，就是满足约束限制的答案。	
其它	时间复杂度为 O（SPFA）。	
PKU 1275	Cashier Employment	一个商店，给出你 24 小时里每个小时商店至少需要的员工数。有 N 个应聘者，从应聘时间开始可以工作 8 小时。问满足约束的情况下，最少需要几名应聘者。
算法讨论	定义 Num[I]代表在 I 时刻至少需要的员工数，Sum[I]代表从 I 时刻能够开始工作的人，得出下列不等式： 1：0<=S[I]-S[I-1]<=Sum[I]，即 S[I]-S[I-1]>=0，S[I-1]-S[I]>=-Sum[I]。 2：S[I]-S[I-8]>=Num[I] 3：S[24]+S[I]-S[I+16]>=Num[I]，即 S[I]-S[I+16]>=Num[I]-S[24]。 与原来不同的是，3 号不等式里出现了三个未知数，但是通过观察可以发现，S[24]是一个固定的值，所以我们可以通过枚举来取消 S[24]这个未知数。 引用别人一句话：如果在一个未知数定死的情况下，要求其它所有未知数的最小值怎么办？只要反过来求最长路径就可以了。最长路径中的三角不等式与最短路径中相反： D[v]>=D[u]+W[U,V]，即 D[v]-D[u]>=W[U,V]。	

其它	时间复杂度为 O（N*SPFA）。	
PKU 1486	Sorting Slides	有 N 个矩形和 N 个点，若某矩形包含这个点，则连一条匹配边。问哪些匹配是必须的。
算法讨论	先求出最大匹配。显然若某条边是必须边，则删除这条边，重新从该点的一个端点进行匹配是无法重新匹配上的，否则就不是必须边。	
其它	时间复杂度为 O（N^2*M^2）。	
PKU 2226	Muddy Fields	N 行 M 列的地，*代表泥地，.代表草地。你必须用宽为 1，长度不限的木板将泥地覆盖住，且不能覆盖草地。问最少需要几块木板。
算法讨论	如果是普通的行列覆盖，则我们将行作为元素，将列作为元素，进行二分图匹配，即可求出行列覆盖。 但是，若不能覆盖到草地。我们则需要重新构图。 原来的构图是将整行整列作为元素，那么加入限制后，我们可以将每一行连通的块作为元素，先进行标号，再按原来的方式进行构图即可。	
其它	时间复杂度为 O（N^2*M^2）。	
PKU 3686	The Windy's	有 N 个工件要在 M 个机器上加工，有一个 N*M 的矩阵描述其加工时间。同一时间内每个机器只能加工一个工件，问加工完所有工件后，使得平均加工时间最小。
算法讨论	将工件作为二分图中 X 部的点，总共 N 个。 将每个机器拆成 N 个点作为二分图中 Y 部的点，总共 N*M 个。第 J 个机器的第 P 个点代表，使用机器 J 进行倒数第 P 次加工。 假设我们按顺序在 J 机器上工件 I1,I2,I3..IK 个工件，则总共需要花费 I1*K+I2*(K-1)+I3*(K-3)+...+IK。 所以我们对于 X 中每个点 I，Y 中每个点（J,P），连接一条 A[I,J]*P 权值的边。 接下来进行二分图最佳匹配或费用流即可。	
其它	X	
PKU 1904	King's Quest	有 N 个男生，N 个女生，给出喜欢关系，则他们可以结婚。问每个男生至多可以和几个女生结婚，且其它男生和女生都可以结婚。给定一个初始的匹配。
算法讨论	直观的思路，枚举着进行二分图匹配，经过测试，当 N=2000,M=100000 时就会 TLE。 所以，我们必须寻找更好的方法。 我们先进行构图，若 I 喜欢 J，则 I 向 J 连一条有向边。 在题目给定的一种匹配方案中，J 向 I 连一条边。 这样，我们可以发现，图中必定存在强连通分量。 而这正好和匹配的思想类似，通过不断的走正向反向边，找到了一条增广路强连通分量。 所以，强连通分量里内部是可以互换匹配的，即互相喜欢的 I 和 J 是可以结婚的。	

其它	时间复杂度为 O（NM）。	
PKU 2060	Taxi Cab Scheme	有 N 个订单，分别给出每个订单的开始时间，起点和目的地。问，最少需要几辆出租车？
算法讨论	把每个订单看做一个顶点，对于两个订单 A 和 B，若 A 完成之后，能在 B 开始之前赶到 B 的起点处，则在 A 到 B 之间连一条边。 显然，我们是要求这个图的最小路径覆盖。（什么是最小路径覆盖？） 所以，我们转换成求二分图最大匹配，结果就是 N-匹配数。	
其它	时间复杂度为 O（N^2*K）。	
PKU 2594	Treasure Exploration	给你一个无环有向图，问可相交的最小路径覆盖。
算法讨论	显然是一个最小路径覆盖，不过可以经过同一个点。 既然可以经过同一个点，我们完全可以把连通性进行传递。 这样就又转换成了普通的最小路径覆盖问题。	
其它	时间复杂度为 O（N^3）。	
PKU 1719	Shooting Contest	给你一个 N*M 的矩阵，某个格子[I,J]是白色，也就是说 I 行和 J 列可以匹配。题目要求的就是，I 和 J 进行最大匹配，必须匹配完所有的 I 行，剩下的 J-I 列可以随意和 N 行进行再次匹配。
算法讨论	进一步简化题意，二分图匹配，行可以重复匹配，但是必须保证所有行至少被匹配一次，求方案。 算法大概就是，进行若干次最大匹配。 由于列只能被匹配一次，所以每次匹配完后将列从图中删除再次匹配，直到某一次的匹配为 0。 然后就是判断是否有解了，若有则输出匹配方案即可。	
其它	时间复杂度为 O（NM*K）。	
PKU 3189	Steady Cow Assignment	有 N 头奶牛，M 个牛棚，每个奶牛对每个牛棚均有一个喜爱值，依次递增。将 N 个奶牛放进 M 个牛棚里，使得所有奶牛中最大喜爱值-最小喜爱值+1 最小。
算法讨论	设置 head，tail 两个指针，用来限制喜爱值，初始值均为 1。 然后进行最大流，若流量为 N 则更新最优值，缩小区间，即 inc(head)，否则扩大区间，即 inc(tail)。 这样，最后输出最优值即可。	
其它	利用双指针进行维护，是很重要的思想。	
SGU 185	Two shortest	N 个点 M 条边的无向图，求两条没有重边的从 1 到 N 的最短路。

算法讨论	1、对图进行最短路算法。 2、若 Dis[J]=Dis[I]+A[I,J]，则 I 向 J 连一条容量为 1 的边。 3、对新图进行最大流算法。 4、若找到两条最短路则根据网络图输出，否则无解。	
其它	时间复杂度 O（N^2*M）。	
PKU 2112	Optimal Milking	N 个挤奶机，M 头牛，每个挤奶机可以容纳 K 头牛。牛和牛，牛和挤奶机，挤奶机和挤奶机之间均有距离，求所有牛都可以喝到奶的情况下，所走最大距离的最小值。
算法讨论	先用 Floyd 求出任意两点间的最短路。 这样，我们二分枚举答案。 下面进行构图，若牛和机器之间的距离不超过答案，则连一条容量为 1 的边。 新增源点和汇点，源点向牛连一条容量为 1 的边，机器向汇点连一条容量为 K 的边。 求最大流，若流量为 N 则缩小答案，否则增大答案。	
其它	时间复杂度 O（N^2*M*LogAns）。	
PKU 1637	Sightseeing tour	N 个点 M 条边，边可能是有向边或无向边，求一条欧拉回路。
算法讨论	有向图的欧拉回路有一个性质，即入度=出度，这个和网络流非常的类似。 所以，我们可以转换成网络流问题求解。 先把无向边任意定向，得到新图，计算每个点的入度 R[I]和出度 C[I]。 设 T[I]=R[I]-C[I]，若 T[I]>0 则代表 I 号点多余了 T[I]的入度，若 T[I]<0 则代表 I 号点多余了-T[I]的出度。 所以需要更改 T[I]/2 条边来满足入度=出度，若 T[I]为奇数，显然无解。 所以设置源点 S 和汇点 T，若 T[I]>0 则 I 向 T 连一条容量为-T[I]的边，否则 S 向 I 连一条容量为 T[I]的边。 然后将图中定向后的无向边加入，容量为 1。 求最大流，若最后 S 的所有出弧满载，则有解，否则无解。 若要求方案，则根据图的残留网络，若某条边满载则反向，之后就是一个欧拉回路图。	
其它	时间复杂度为 O（N^2*M）。	
PKU 1815	Friendship	给你 N 个点、源点和汇点，然后有一个 N*N 的矩阵，若为 A[I,J]=1，则代表 I 向 J 连一条边，求一个起点到终点的最小点割集。
算法讨论	很明显的是求最小点割集，利用经典的构图方式，转换成求最小边割集。 将每个点 I 拆成 I 和 I'两个点，I 向 I'连一条容量为 1 的边。 然后读入图中原始连边，若 I 向 J 连一条边，则在新图中，I'向 J 连一条边，容量为正无穷。 这样，我们就转换成了最小边割集，属于最小边割集合的边上的两个端点属于最小点割集。 由于要求的是字典序最小的方案，所以不能使用 DFS。 只能进行枚举。	

	枚举删边后，若流量减小，则该点必定属于最小点割集，然后真正的删除它，更新当前流量。	
其它	时间复杂度为 O（N^3*M）。	
PKU 1966	Cable TV Network（Un AC）	N 个点 M 条边的无向图，问最少去掉几个点可以使得图不连通。
算法讨论	若我们知道源点和汇点，则此题就可以转换成求最小点割问题。（如何求最小点割？） 所以，我们枚举源点和汇点，取其中的最小值即可。	
其它	在 POJ 未 AC，但是官方数据能过。	
PKU 2987	Firing	有 N 个点，每个点有一个权值。有 M 个关系（A,B），代表 A 是 B 的上司。从这个图里面选择一些点，使得权值最大，若选择了 A 且 A 是 B 的上司，则必须选择 B。
算法讨论	很典型的最大权闭合图问题。（什么是最大权闭合图？） 第二问就是总盈利-总亏损，即 Sum{正权值}-MaxFlow。 下面讨论第一问。 由于最小割一定只会出现在（S,I）和（I,T）中，且最小割一定是一个可行方案。 所以，我们只需要求出 X 集中的点，只要选择这些点，一定可以达到最小割，一定可以得到可行方案。	
其它	时间复杂度为 O（N^2*M）。	
PKU 3204	Ikki's Story I - Road Reconstruction	N 个点，M 条边的有向图，给出每条边的起点，终点和容量。仅可以修改一条边的容量，若修改后可以增加网络图中的最大流量，则称为有效边。最后输出这个图中有多少条有效边。
算法讨论	设源点为 St，汇点为 Ed。若某条边（X，Y）是有效边，则从 St 开始走非饱和边必然可以到达 X，从 Y 开始走非饱和边必然可以到达 Ed。 否则，St-X 这段路径的流量或 Y-Ed 这段路径的流量是固定的，所以无法通过修改（X，Y）的流量来增加网络图的最大流量。 接下来，我们从源点正向 DFS，汇点反向 DFS 求出其可以到达的点，枚举每一条边，判断其两个端点是否满足条件即可。	
其它	时间复杂度为 O（N^2*M）。	
PKU 3469	Dual Core CPU	有 N 个点，两个集合。设 A[I]代表 I 在 A 集合需要的代价，B[I]代表 I 在 B 集合的代价。然后给出 M 个关系，（X,Y,D），代表若 X 和 Y 在不同的集合需要 D 的代价。求最后代价最小值。

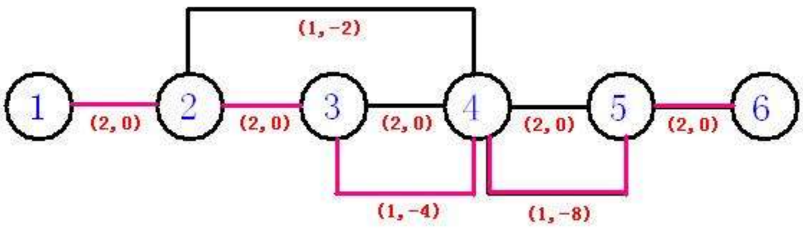
<table>
<tr><td>算
法
讨
论</td><td colspan="2">设 St 为源点，Ed 为汇点。
对于每个点 I，从 St 向 I 连一条容量为 A[I]的弧，从 I 向 Ed 连一条容量为 B[I]的弧。
对于每个关系（X,Y,D），从 X 向 Y 连一条容量为 D 的弧，从 Y 向 X 连一条容量为 D 的弧。
仔细观察后，我们可以发现。
最小代价就等于图中的最小割。</td></tr>
<tr><td>其它</td><td colspan="2">时间复杂度为 O（N^2*M）。</td></tr>
<tr><td>PKU 3084</td><td>Panic Room</td><td>针对输入数据来说题目意思吧。
7 2 （N 个房间，要保护 2 号房间）
NI 0 {0 号房间，NI 代表没有入侵者}
I 3 0 4 5 {1 号房间，I 代表该房间内有入侵者，该房间有 3 个门，分别连向 0，4，5 号房间，且控制端在 2 号房间}
NI 2 1 6
NI 2 1 2
NI 0
NI 0
NI 0
门一开始都是可以打开的，若要关闭或再次打开某扇门必须在控制端进行。
问使得所有入侵者不进入要保护的房间最少需要关闭几扇门，如果入侵者必然会进入要保护的房间则输出 PANIC ROOM BREACH。</td></tr>
<tr><td>算
法
讨
论</td><td colspan="2">设源点为 St，汇点为 Ed。若某条边（X，Y）是有效边，则从 St 开始走非饱和边必然可以到达 X，从 Y 开始走非饱和边必然可以到达 Ed。
否则，St-X 这段路径的流量或 Y-Ed 这段路径的流量是固定的，所以无法通过修改（X，Y）的流量来增加网络图的最大流量。
题目中已给出汇点，即要保护的房间，设为 T，新增源点 S。
对于有入侵者的房间 I，连一条从 S 向 I 容量为 MaxLongint 的弧。
对于每扇门连接的（I,J），若控制端在 I，则从 I 向 J 连一条容量为 Max 的弧，从 J 向 I 连一条容量为 1 的弧。
这样，想使得 S 和 T 不连通，图中的最小割就对应了一种最小的关门方案，如果割中存在容量为 Max 的弧，显然是无解的。</td></tr>
<tr><td>其它</td><td colspan="2">时间复杂度为 O（N^2*M）。</td></tr>
<tr><td>ZJU 2587</td><td>Unique Attack</td><td>N 个点 M 条边的无向图，给定源点和汇点，判断最小割是否唯一。</td></tr>
<tr><td>算
法
讨
论</td><td colspan="2">首先是求最小割集的算法：求出最大流，然后在残留网络中从 s 点开始 DFS，访问到的点与未访问的点就组成了一组最小割集[S1,T1](对于 S1,T1,两者之间必须原来有边)(反证法可以直接证明去掉这些割集以后一定不连通)。
由这个算法也可以得到割边一定满流.
无向图直接把原来的弄成两条边放图里就好(可以证明这两条边至多只有一条有流)</td></tr>
</table>

<table>
<tr><td></td><td colspan="2">然后是唯一性问题，有如下结论：按照上面这个算法求出的最小割集[S1,T1]，对于任意一组最小割集[S,T]，都一定满足 S1 属于 S。
同理，我们在原图中，从 t 向 s 求最大流，然后在残留网络中，从 t 开始 DFS(T->S 的时候,DFS 的条件是目标点到当前点有剩余)，未访问的点与访问到的点组成一组最小割集[S2,T2]。则对于任意一组最小割集[S,T]，都一定满足 S 属于 S2。
这样，只需要求两次最大流，即可判断最小割集的唯一性。(若有两不同最小割,则 DFS 的时候一定出来的是不同的割集)
（转自 http://www.gnocuil.cn/，略有改动）</td></tr>
<tr><td>其它</td><td colspan="2">时间复杂度为 O（N^2*M）。</td></tr>
<tr><td>SGU 326</td><td>Perspective</td><td>有 N 个球队在同一个赛区，已知他们胜利的场数，还剩下的在赛区内的比赛数和跨赛区的比赛数的和，和在赛区内的比赛对阵矩阵。问，1 号球队是否可以不小于其余球队胜利场数的最大值。</td></tr>
<tr><td>算法讨论</td><td colspan="2">先贪心一下，让 1 号球队赢得所有比赛，其余球队输掉所有跨赛区的比赛。如果此时有球队比 1 号球队胜利场次多，显然直接输出 NO。
否则，对于在同一赛区的比赛，我们这样构图。
新增源点 S，汇点 T。
对于每个点 I，从 S 向 I 连一条容量为和 1 号球队胜利场数之差的弧。
对于赛区内的每一个进行 K 次的对阵（I,J），新增一个点 P，从 I 向 P 连一条容量为 K 的弧，从 J 向 P 连一条容量为 K 的弧，从 P 向 T 连一条容量为 K 的弧。
这样，去一遍最大流，若所有 P 指向 T 的弧都满流，则输出 YES，否则输出 NO。
为什么说所有 P 指向 T 的弧都满流就是 YES 的？
假设某条 P 指向 T 的弧不满流，则 S 指向 I 和 S 指向 J 的弧已经满流，但是还比赛还没有打完，不管剩下的比赛谁赢都将比 1 号球队的胜利场次多。所以若所有 P 指向 T 的弧都满流，则输出 YES，否则输出 NO。</td></tr>
<tr><td>其它</td><td colspan="2">时间复杂度为 O（MaxFlow）。</td></tr>
<tr><td>PKU 3422</td><td>Kaka's Matrix Travels</td><td>N*N 的棋盘，每个格子有一个权值 A[I,J]。一个人在（1，1）的位置，要走到（N，N）的位置，每次可以向下走一格或向右走一格，没走到一个格子获得权值 A[I,J]且更改 A[I,J]为 0，问走 K 次后，最多可以获得多少权值是多少。</td></tr>
<tr><td>算法讨论</td><td colspan="2">先将点权变为边权值，拆点变为[I,J]和[I,J]'。
从[I,J]向[I,J]’连一条容量为 1，费用为 A[I,J]的弧。
从[I,J]向[I,J]‘连一条容量为 Max，费用为 0 的弧。
从[I,J]‘向[I,J+1]连一条容量为 Max，费用为 0 的弧。
从[I,J]’向[I+1,J]连一条容量为 Max，费用为 0 的弧。
汇点为[N,N]'，新增源点为 S。
从 S 向[1,1]连一条容量为 K，费用为 0 的弧。
由于要求"最大费用"，所以将权 T 变为-T，求一遍最小费用最大流即可。(因为权是负数，所以用 ZKW 神牛的费用流算法时，要先用 SPFA 初始化除最</td></tr>
</table>

	短路）	
其它	时间复杂度为 O（MinCost-MaxFlow）。	
SPOJ 371	Boxes	N 个盒子围成一圈，每个盒子里有 A[I]个球，每向相邻的盒子移动一个球需要花费 1 的代价。求，最后所有盒子至少有一个球时花费的最小代价。
算法讨论	新增源点 S，汇点 T，把每个盒子看成点 I。 S 向 I 连一条容量为 A[I]，费用为 0 的弧； I 向 T 连一条容量为 1，费用为 0 的弧 I 向 I+1 连一条容量为 Max，费用为 1 的弧； I+1 向 I 连一条容量为 Max，费用为 1 的弧。 显然，求一遍最小费用最大流就可以得到最小代价。	
其它	时间复杂度为 O（MinCost-MaxFlow）。	
POI 2001	和平委员会	有 N 个党派，每个党派有两个人，给出一些冲突信息，要求从每个党派中选出一个人，总共选出 N 个人，使得 N 个人互相之间不冲突。
算法讨论	经典的 2-sat 模型，详细的可以看 2003 国家集训队伍昱的论文（貌似只有个 PPT）。 算法流程： 1．构图（如何构图？若 I 和 J 冲突，则 I 向 J'连边，J 向 I'连边，保证对称性） 2．求图的极大强连通子图（利用 Kosaraju 法，正向遍历图记录后序，按之前的后序顺序反向遍历图，每次遍历出的就是一个强连通分量） 3．把每个子图收缩成单个节点，根据原图关系构造一个有向无环图 （显然强连通分量中若选某个点必然会选择到其他的点） 4．判断是否有解，无解则直接输出（对于任意的 I 或 I'若处于同一强连通分量，则无解） 5．对新图进行拓扑排序 6．自底向上进行选择、删除（之所以自底向上，是因为最下面的点没有出度，所以不会产生错误） 7．输出	
其它	时间复杂度为 O（M）。	
PKU 2723	Get Luffy Out	N*2 种类型的钥匙，M 道门，每道门可以用两种类型之一的钥匙打开，门必须按顺序打开，问最多可以打开几道门。
算法讨论	先二分枚举答案，然后利用 2-Sat 进行验证。 先把编号进行预处理，每个门上的锁变成编号从小到大。 构图： 1）若 X 和 Y 在同一个门上，则 X 向 Y'连一条边，Y 向 X'连一条边。 2）若两个 X 在同一个门上，则 X 必然会使用，则 X'向 X 连一条边。	
其它	时间复杂度为 O（MLog（N+M））。	

PKU 3678	Get Luffy Out	有 N 个变量，为 0 或 1，给出一堆逻辑关系（AND,OR,XOR），问是 N 个变量是否有一种取值方案。
算法讨论	把每个变量 X 看成两个点，I 代表 0，I'代表 1。 把每个变量 Y 看成两个点，J 代表 0，J'代表 1。 Add（I,J）代表从 I 向 J 连一条有向边。 构图，分情况讨论： 1） X AND Y=1，Add（I',J'），Add（J',I'），Add（I,I'），Add（J,I'） 2） X AND Y=0，Add（I',J），Add（J',I） 3） X OR Y=1，Add（I,J'），Add（J,I'） 4） X OR Y=0，Add（I,J），Add（J,I），Add（I',I），Add（J',J） 5） X XOR Y=1，Add（I',J），Add（J',I），Add（I,J'），Add（J,I'） 6） X XOR Y=0，Add（I',J'），Add（J',I'），Add（I,J），Add（J,I） 接下来就是 2-Sat 验证。	
其它	时间复杂度为 O（M）。	
PKU 2749	Building roads	有两个中转站，和 N 个点。每个点要么连到 1 号中转站，要么连到 2 号中转站。给出一些限制信息，即哪两个点必须连到一个中转站上，哪两个点必须不能连到一个中转站上。若两个点在同一中转站上，则距离是两者到中转站的距离和，否则是两者到中转站的距离和加上中转站之间的距离。求一种方案，使得任意两点之间的最大距离最小。
算法讨论	题目要求取最大的值最小，我们则用二分来枚举答案，然后 2-Sat 验证。 把 X 点连到 1 号中转站上设为 I 点，连到 2 号中转站上设为 I'点。 把 Y 点连到 1 号中转站上设为 J 点，连到 2 号中转站上设为 J'点。 四种构图方式： 1） Add（I,J'），Add（J,I'） 2） Add（I',J），Add（J',I） 3） Add（I,J），Add（J',I'） 4） Add（I',J'），Add（J,I） 对于题目中的输入信息，当两个点必须连到一个中转站上时，使用 3，4 号构图方式。当两个点必须不连到一个中转站上时，使用 1，2 号构图方式。 设 Dis1[I]代表 I 到 1 号中转站的距离，Dis2[I]代表 I 到 2 号中转站的距离，D 代表中转站之间的距离，每次二分答案 Ans。 若 Dis1[I]+Dis1[J]>Ans，使用 1 号构图方式 若 Dis2[I]+Dis2[J]>Ans，使用 2 号构图方式 若 Dis1[I]+Dis2[J]+D>Ans，使用 3 号构图方式 若 Dis2[I]+Dis1[J]+D>Ans，使用 4 号构图方式。 接下来就是进行 2-Sat 验证，继续二分。	
其它	时间复杂度为 O（N^2*LogAns）。	
SGU 213	Strong Defence	给定一个无向图，图中有一个起点 S 和一个终点 T。要求选 K 个集合 S1，S2，…，SK，每个集合都含有图中的一些边，任意两个不同的集合的交集为空。

		并且从图中任意去掉一个集合，S 到 T 都没有通路。要求 K 尽量大。
算法讨论	先求出 S 到 T 的最短路 L，显然，割切集合的数量一定是 L 个。（证明见 Zhou Yuan 神牛的集训队作业） 所以，独立割集合问题就转换成了最短路问题，我们只要求出 S 到 T 的最短路，把路经的边进行记录，输出即可。	
其它	时间复杂度为 O（Short-Path）。	
[SGU 121]	[Bridges painting]	给定一个无向图，要求给这个图上的边 0、1 染色，从而保证每个度不小于 2 的点都至少能连出一条 0 边，也至少能连出一条 1 边。
算法讨论	对于图中的不同连通块，用相同的算法分别进行处理。 对于每个连通块，建立一颗深度优先搜索树，记录每个结点的父结点、儿子结点、深度。 对于每条正向边（X,Y），按如下方式进行构造： 1、如果 X 是根节点，则第一个儿子染 0 号颜色，其它染 1 号颜色。 2、如果 X 非根节点，则根据上一次的颜色进行染色，即染其祖父节点连向其父节点的相反颜色。 下面考虑反向边（X,Y），按如下方式进行构造： 1、如果 X 非叶子节点，则只染 1 号颜色。 2、如果 X 是叶子节点，则把该反向边看成正向边染色。 显然，这样构造之后，如果根节点的儿子大于等于 2，则是一个有效的染色方案，否则要进一步讨论。 如果根节点的儿子只有 1 个，不考虑反向边，那它只有一条 0 号边。加入反向边（X,Y），我们可以发现： 1、如果 X 非叶子节点，Y=根节点，则根节点拥有了 1 号边，有效染色方案。 2、如果 X 是叶子节点，Y=根节点，且反向边染的是 1 号颜色，有效染色方案。 3、当上面两个有效染色方案都不成立时，则需要进行变换。任找一个（X 是叶子节点，Y=根节点）的边。对于 X 向父节点走，直到走到一个儿子数大于等于 2 的节点，所经的边，颜色全部反向。这样，又是一个有效染色方案。 4、否则，无解。 很好的一个题啊，更详细题解见周源的 SGU 表格。	
其它	时间复杂度为 O（M）。	
[PKU 3680]	[Intervals]	数轴上有 N 个区间，每个区间有一个权值 C，你可以选择任意多的区间，但是数轴上某一段不能被覆盖超过 K 次，求最大能获得的权值。
算法讨论	首先进行离散化，把每个区间看成两个端点。 新增源点 S，汇点 T。 把 S 设为所有端点中最左的端点，把 T 设为所有端点中最右的端点。 将所有端点排序。 对于排序后的每个端点，从 I 向 I+1 连一条容量为 K，费用为 0 的弧。 对于原区间（A,B），排序后新位置为（W,Y），则从 W 向 Y 连一条容量为 1，费用为-C 的弧。	

<table>
<tr><td></td><td colspan="2">例如：
3 1
1 3 2
2 3 4
3 4 8
我们可以得到如下的图（图画错了，把所有的（2，0）改成（1，0）。然后，带颜色的边最终方案）：


一旦走了费用不为 0 的边，则在走其对应的区间时，流量少一，当走完它所走的区间，流量又补回来了一，这样便保证了区间的覆盖次数不超过 K。
然后求一遍最小费用最大流即可。</td></tr>
<tr><td>其它</td><td colspan="2">时间复杂度为 O（MinCost-MaxFlow）。</td></tr>
<tr><td>SGU 122</td><td>Strong Defence</td><td>N 个点的无向图，每个点的度都大于等于（N+1） DIV 2，求一条从节点 1 开始的哈密顿回路。</td></tr>
<tr><td>算法讨论</td><td colspan="2">哈密顿回路本来是个 NP 问题，但是此题特殊之处在于每个点的度都大于等于（N+1） DIV 2，所以我们可以利用经典算法在 O（N^2）的时间内构造出这条哈密顿回路。
基本思想就是，找链，延长链，转环，判断，再找延长链，再、、、如此反复。
0、初始时，链中仅有一个 1 号节点。
1、每次，我们选择未在链（环）中且和链中最后一个元素相连的点来延长链。
2、当无法延长时，我们将这条链转成一个环，即存在（S,I-1）（I,T）这样的关系，我们将（I,T）与（S,I-1）位置互换，（S,I-1）反转。（实际上就是将（I,T）反转）
3、如果此时环已经有了 N 个节点，则是一个有效方案。
4、否则，更新这个环，若在环（链）中的某点 X 和未在环（链）中的某点 Y 相连，则将 X 移动到环（链）尾部。
这样，就可以构造出一个哈密顿回路。。。</td></tr>
<tr><td>其它</td><td colspan="2">时间复杂度为 O（N^2）。</td></tr>
<tr><td>SGU 138</td><td>Games of Chess</td><td>有 n 个人下棋，每次有两个人下，赢者留下再比(可以与刚下败的人)。给出每个人下的次数，要求出一种可行方案，输出每次比赛的过程，赢者放在第一列，败者放在第二列。</td></tr>
<tr><td>算法讨论</td><td colspan="2">把每个人看成图中的每个点，则题目中给出了每个点的度。
这个题有一个特殊的地方，失败的人除了减少一个点的度数之外，没有任何影响。
所以，我们可以先从大到小排序，按顺序找出所有比赛中胜利的人，接着把失败的人依次填入即可。
胜利规则如下：</td></tr>
</table>

	1）如果在胜利场次之内，每个人至多胜利 A[I]-1 场，最后一场必输给第 I+1 个人，否则无法传递到下面。 2）如果非胜利场次之内，每个人至多胜利剩余的场次。	
其它	时间复杂度为 O（N^2）。	
SGU 190	Dominoes	N*N 的棋盘（N<=40），有 M 个点被放置障碍。要求用 1*2 或 2*1 的骨牌覆盖整个图，骨牌之间不能相互覆盖。问是否有方案，方案是什么。
算法讨论	对棋盘进行 01 染色，相邻的两个格子间必然不同颜色。 这样，就形成了一个二分图。 我们对相邻的两个均未被放置障碍的格子进行连边，之后进行二分图匹配。 若最大匹配*2=N*N-M，则有解，根据匹配输出即可，否则无解。	
其它	时间复杂度为 O（N^4）。	
SGU 210	Beloved Sons	N 个男的、N 个女的（N<=400），互相匹配。每个男的有权值，求最后权值最大。
算法讨论	咋看此题，很容易的就可以想出最佳匹配。但是复杂度会偏高，所以还可以进行优化、转换。 当按权值将男生排序后，依次匹配，根据最大匹配的性质，一旦匹配上了就不会再取消匹配。 所以，最后一定是最优的。	
其它	时间复杂度为 O（NM）。	
SGU 212	Data Transmission	N 个点 M 条边，每个点的距离标号已给出，且 I 阶段的点只向 I+1 阶段的点连边，求图中不可增广的可行流。
算法讨论	由于本图比较特殊，而且不用求最大流，所以我们不用再退流，利用多路扩展一直流下去即可。 但是，这样的复杂度也会比较高，会 TLE，所以还需要继续优化。 一个最容易想到的优化就是贪心初始流，这样就可以在很快的时间内出解了。	
其它	X	
PKU 3207	Ikki's Story IV - Panda's Trick	一个圆上有 N 个点，顺时针从 0 到 N-1 排列。给出 M 条线连接两个点，要么在圆外要么在圆内，判断给出的所有线段是否不规范相交。
算法讨论	把每个线段看成两个点，I 代表在圆外，I'代表在圆外。 显然，本题要求的就是对于所有 I 和 I'只取一个，问是否可以取完所有 M 组点。 所以就变成了一个简单的 2-sat 验证问题。 构图，若 I 和 J 相交，则 I 向 J'连一条边，J'向 I 连一条边，I'向 J 连一条边，J 向 I'连一条边。	
其它	时间复杂度为 O（M^2）。	
PKU 2832	How Many Pairs?	N 个点 M 条边的无向图 Q 次询问，对于每次询问，如果（U,V）之间某条路径的最长边小于等于询问

		的值，则是有效点对。问每次询问的有效点对有多少个。
算法讨论	先对边和询问从小到大进行排序。 之后对于每次询问，将所有比它小的边从上次询问的基础上加入图中。 用并查集合并连通分量，统计个数。 由于并查集是合并两个集合，则先把两个集合的有效点对减去，再把大集合的有效点对加上。	
其它	时间复杂度为 O（QLogN+M）。	
SGU 218	Unstable Systems	求一个二分图匹配的完备匹配，使得匹配中最大边最小。
算法讨论	最大边最小，显然可以二分答案。 然后利用二分图匹配进行验证即可。	
其它	时间复杂度为 O（NM*LogAns）。	
PKU 3683	Priest John's Busiest Day	有 N 个区间（A,B），和一个值 C。你可以选择使用（A,A+C）或（B-C,B）中一个区间，使得所有选择出来的 N 个区间不相交，求方案。
算法讨论	典型的 2-Sat。 设区间（A,B）分为 I 和 I'两个区间。 若 I 和 J 和 J'相交，则 I 肯定不能选，从 I 向 I'连一条边。 若 I 仅和 J 相交，则 I 向 J'连一条边，J 向 I'连一条边。 若 I 仅和 J'相交，则 I 向 J 连一条边，J'向 I'连一条边。 接下来 2-Sat 判定，构造即可。	
其它	时间复杂度为 O（N^2）。	
SGU 219	Synchrograph	给出一个有向图，图的每条边都有一个非负权值，对这个图可以进行如下操作：每次取一个所有指向它的边的权值都为正数的顶点，将所有指向这个顶点的权值都减 1、将所有由这个顶点指向其他点的权值都加 1。对于一个顶点 V，如果对这个图任意执行多次这样的操作后，还能找到一个这样的操作序列，使这个序列包含 V，则 V 称为活点。现在求图中所有的活点。
算法讨论	没有思路，再次读题，可以推出两条重要的性质。 性质 1：每个强连通分量，要么全部是活点，要么全部不是活点。 性质 2：如果某个点的前驱不是活点，则这个点就不是活点。 所以，我们可以得出如下算法： 1）先对所有强连通分量进行缩点。 2）对缩点后的图进行拓扑排序，如果某个点的所有前驱都是活点，则该点为准活点。 3）对于每个准活点，对其内部按规则进行遍历，如果该点内部的所有点都恰好可以遍历到，则该点是活点。	

<table>
<tr><td></td><td colspan="2">这样，对活点和非活点进行标记，输出即可。</td></tr>
<tr><td>其它</td><td colspan="2">时间复杂度为 O（N+M）。</td></tr>
<tr><td>SPOJ 1553</td><td>Backup Files</td><td>给出 N 个点（N<=100000）在 X 轴上的坐标，任意两点可以连一条线段，求 K 条线段，每个点至多被一条线段占据，最后使得 K 条线段的长度最短。</td></tr>
<tr><td>算法讨论</td><td colspan="2">首先容易发现，选的边一定不会互相覆盖，即对于四个点 a<b<c<d，若要求选两条边，则(a-c, b-d)必定不是最优解，因为显然(a-b, c-d)更优。由此可得到一个 O(NK)的 DP 解法，即用 opt[n][k]表示前 n 个点中选出 k 条边的最优解，则有方程 opt[n][k] = min(opt[n-1][k], opt[n-2][k-1] + pos[n] - pos[n-1])。但这个是明显 TLE 的。
换一个角度来考虑，如果我们把奇数位置的点当做 X 集，偶数位置的点当作 Y 集，相邻点之间连边，容易发现这是一个二分图，又是一个求二分图的最优匹配的问题。如果转化为最小费用流，则 k 次增广后即可得到结果。当然还要借助图的特殊性，实际上每条增广路对应的是连续一段边，并且端点处的两条边是未被选择的，一次增广实际上就是把这段区间内的边状态反转一下(已选<->未选)。比如有六个点从左到右编号为 1-6，现在选择的是(2-3),(4-5)，则一次增广后变为(1-2),(3-4),(5-6)。由连续最短路（最小费用增广路）算法可知，每次选择最小的增广路，即可得到最优解。
于是算法也就有了，初始状态把每条边 i 都是独立的，区间是[i,i]，权值为此边长度。每次选择权最小的边（设权为 v），再找出它左右相邻的边（设权为 v1,v2），把这些权值都删除掉，再加入新边，区间为这三部分的合并，权值为(v1+v2-v)。权值可以用平衡树或堆之类维护，可以使每次操作做到 O(logN)，则总的复杂度为 O(KlogN)。
另外要注意的是最两端的边是不能“增广”的，也就是说一旦被选中，只把它删除就好，不必加入新边了。
（转自：http://hi.baidu.com/roba/blog/item/59678744f391f24e510ffea4.html）</td></tr>
<tr><td>其它</td><td colspan="2">时间复杂度为 O（NlogN*M）。</td></tr>
</table>

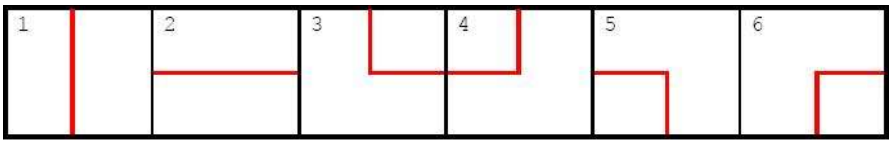
搜索策略：

题目来源	题目名称	题目大意
PKU 1915	Knight Moves	在 0..300 范围的棋盘上，给定起点和终点和八种行走方式，求起点到终点最小步数。
算法讨论	给定了起始状态和末尾状态，求最小步数。显然是用 BFS，为了节省时间，我选择了双向 BFS。何为双向 BFS，即从起点向终点搜，从终点向起点搜，各自扩展各自的状态，直到某一次两人扩展的状态重合。一个优化，每次选择节点少的开始扩展。	
其它	双向 BFS 练习题。	
PKU 1321	棋盘问题	给一个 N*N 的棋盘，上面有的点可以放棋子，有的不可以放。给你 M 个棋子，要你放在棋盘上，且每一行、每一列只能有一个棋子。求方案数。
算法讨论	跟八皇后类似，不过有障碍点，而且棋子数目不一定，不过思路还是一样的，DFS 每一行，放或不放，若放则循环放该行的每一列，列是否能放用位运算来完成。	
其它	X	
PKU 2531	Network Saboteur	N 个点的无向图，任意两点之间有权值，将 N 个点分成两个集合 AB，求 A 到 B 的权值和的最大值。
算法讨论	很明显的 DFS 题，我们以 A 集合内元素的个数为阶段进行 DFS。有三大优化：1）只搜一半，因为另一半是对称的 2）若某点想加入 A 集合，则必须要比原来的值大才行 3）若上次是 x 加入了 A 集合，则下一次只用从 x+1 开始搜。另外，还有一个小优化，由于元素只有 20 个，所以我们可以用位运算来判断某元素是否属于 A 集合。	
其它	跑了 79ms。	
PKU 3411	Paid Roads	给你一个有向图，起点为 1，终点为 N，M 条边（0<=N、M<=10），每条边有两个权值，第一个权值<=第二个权值，但是第一个权值必须在经过指定节点 K 后才能使用。两点之间可能有多条边，每条边可以重复经过无限次。求 1 到 N 的最短路。
算法讨论	利用 BFS，状态定义为选了哪些点（用位运算实现），末尾节点是谁。这样，不断的 BFS，一旦 F[I,J]'小于 F[I,J]则更新 F[I,J]，将 F[I,J]重新纳入队列里。复杂度相当的低。	
其它	时间复杂度为 O（2^N）。	
PKU 1077	Eight	给你一个棋盘，让你还原成 12345678X 的状态，求一条步数最小的路径。

算法讨论	利用启发式搜索，估价函数为当前棋盘与目标棋盘棋子不一样的个数，启发函数就等于估价函数+深度，每次扩展将启发函数最小的拿出来进行扩展。找最小可以利用堆，判重可以利用康托展开。	
其它	启发式搜索练习题。	
PKU 1084	Square Destroyer	给你 N*N 个 1*1 的小正方形组成的大正方形，问你破坏哪些边可以使得这个大正方形不再拥有任何子正方形。初始的时候，它可能会自动缺少某些边。
算法讨论	先对问题进行抽象，相当于给定了一堆集合，求最少去除几个集合中的元素，可以使得全集均被破坏。我们对每条边是否去除进行 dfs，然后加上 So Many 的减枝进行优化，使得时间复杂度骤降。代码里总共有 3 个优化，2 个最优性减枝，2 个可行性减枝。	
其它	练习优化的搜索题，从 TLE 优化到 800ms，优化的到 400ms，优化到 100ms，优化到 0ms，优化无止境。	
PKU 2449	Remmarguts' Date	N 个点 M 条边，求起点到终点的第 K 短路。
算法讨论	启发式搜索，设置状态为，X.v 当前状态的顶点，X.w 当前状态的路径长度。由于此图可以重复走点，所以每个状态都不一样，不用判重。估价函数 H 为该点到终点的最短路，由于 H=H*，所以肯定是相容的。然后 A*搜索，每次搜到终点就记录，直到记录了 K 次后就是第 K 短路了。	
其它	经典题。	
PKU 1011	Sticks	给你 N 根不超过 50 长度的木棒，若将他们等长度的分成几组，求最小长度。
算法讨论	大方向和我一样，不过有很强的减枝。1）为了避免重复，我们要求第一组最长木棒的长度要大于第二组，依次类推。所以，当搜索每一组的第一根木棒时，只用找一个最大的未用过的木棒即可，别的木棒不用考虑。2）若一根木棒正好可以填满一组，则不用再考虑比它小的木棒了。	
其它	X	
PKU 2286	The Rotation Game	给你八种操作，求中间一圈都是同一个数字时的操作序列，和那个数字。
算法讨论	就是一个模拟式的 IDA*，注意模拟细节就行。	
其它	X	
PKU 1054	The Troublesome Frog	给你个 N*M 的方阵，求一条路径，这条路径满足：1）起点在方阵外，终点在方阵外 2）路径上相邻的两点距离相等，一条包含点数最多的路径就是解。

算法讨论	枚举第二起点和第三起点，然后求这条路径上包含的点有多少个。剪枝：1）检查第一起点是否在方阵外 2）极端法检查是否可能比答案更优。为了满足空间性质，我们需要使用前向星法来存点，二分枚举求点是否存在。	
其它	时间复杂度为 O（N^2*LogN）。	
PKU 1117	Pairs of Integers	给定一个数 N，求 A+B=N，其中 A 比 B 的位数多 1 位（B 可以有前导 0），且 A 中删除一个数字后可以得到 B。
算法讨论	N 高达 10^9，所以单纯的枚举是行不通的。根据此题的性质，我们可以进行构造式的枚举。我们从枚举 A 的每一位为 X（0..9），如果 2*X 等于 N 的这一位则处理下一位；否则此位为 N 的这一位减去 X，由于 A 和 B 只有一个数不一同，所以以后就没有必要再枚举了，B 的第 I 位就等于 A 的第 I+1 位。这样我们就完成了构造，处理好进位问题即可。当构造长度超过 N 的长度时，我们判断是否可行，若可行则加入输出数组。	
其它	X	
PKU 1691	Painting A Board	给你 N 个小矩形，每个小矩形有一个颜色，让你求最少几步可以把所有矩形消去。同样颜色的矩形可以一步消去。若某矩形被消去，其上方的所有矩形必须全部被消去。
算法讨论	先抽象模型，若 A 矩形在 B 矩形的上方，则 A 向 B 连一条边，每次可以删入度为 0 的同样颜色的点。所以我们用 DFS 搜每个入度为 0 且未消去的点，然后在里面套用 BFS 消去同样颜色的点。优化 1：最优性减枝 优化 2：位运算 优化 3：处理重复状态。	
其它	X	
PKU 1753	Flip Game	有一个 4*4 的棋盘，每个棋盘上有一盏灯，B 代表亮着，W 代表灭了，没按一盏灯，它和它上下左右的四盏灯都会变成相反的状态，问最少按几盏灯可以使得整个棋盘全亮或全灭。
算法讨论	先枚举整个棋盘是亮还是灭，我们假设是亮的。然后我们枚举第一行，即第一行是按还是不按。然后第一行的状态就出来了，若第一行的 A[I,J]还想改变状态只能依靠 A[I+1,J]，所以从第二行开始进行构造，若第一行的 A[I,J]是灭的，则按 A[I+1,J]，反之不按，这样第二行的状态也有了，依此类推，构造下面的所有行。然后判断最后一行构造完毕后，是否全亮，若是则可行，更新 ans 即可。	
其它	时间复杂度为 O（2^N*N^2）。	

动态规划：

<table>
<tr><th>题目来源</th><th>题目名称</th><th>题目大意</th></tr>
<tr><td>PKU 1038</td><td>Bugs Integrated, Inc.</td><td>给你个 N*M 的棋盘，可以在非障碍点上放 2*3 或者 3*2 型号的木块，问最多能放几块。</td></tr>
<tr><td>算法讨论</td><td colspan="2">由于列数很少，我们可以考虑使用四进制状态压缩压缩一行的情况，然后逐格进行 DP。设置四种状态，
0：代表(I-1,J)和(I-2,J)可用。
1：代表(I-1,J)可用，(I-2,J])不可用。
2：代表(I-1,J)和(I-2,J)都不可用。
3：代表(I,J)不可用。
所以有如下三种转移方式：
1）对状态不进行任何操作，直接转移
2）若(I-2..I)(J..J+1)可用，且不非法，则放置 3*2 的木块。
3）若(I-1..I)(J..J+2)可用，且不非法，则放置 2*3 的木块。
然后 ans 在这其中取最优值即可。特别注意的是，每换行一次要处理一下所有的状态，让其四进制值减去 1。</td></tr>
<tr><td>其它</td><td colspan="2">时间复杂度为 O（NM*2^N*K）。</td></tr>
<tr><td>URAL 1519</td><td>Formula 1</td><td>给你个 N*M 的棋盘，有的点有障碍，求总共有多少种哈密顿回路。</td></tr>
<tr><td>算法讨论</td><td colspan="2">由于范围较小，我们使用状态压缩动态规划来解决这类连通性问题。
总共有这样几种放插头的方式：


将轮廓线上的插头转换为括号表示法，0 代表无括号，1 代表左括号，2 代表右括号，根据括号不可交叉的原理，下面分类讨论，进行动态规划。
设（I,J-1）的右插头为 p，（I-1,J）的下插头为 q。
1）新建一个连通分量
p=0，q=0，使用 6 号插头方式，W[I]=1，W[I+1]=2。
2）合并两个连通分量
p=1，q=1，使用 4 号插头方式，修改 q 对应的右括号为左括号。
p=2，q=2，使用 4 号插头方式，修改 p 对应的左括号为右括号。
p=1，q=2，使用 4 号插头方式，只能出现在最后一个非障碍点。
p=2，q=1，使用 4 号插头方式，不对任何括号做更改。
3）延续一个连通分量
p=0，q<>0，使用 1 号插头方式。
q=0，p<>0，使用 2 号插头方式。</td></tr>
<tr><td>其它</td><td colspan="2">经典题。</td></tr>
</table>

<table>
<tr><td>PKU 1739</td><td>Tony's Tour</td><td>给你 N*M 的棋盘，左下角是起点，右下角是终点，问起点到终点的路径有多少条，每条路径必须经过所有的点。</td></tr>
<tr><td>算法讨论</td><td colspan="2">和 Ural 1519 的那个题是大致一样的。唯一的不一样就是需要把图进行改造，改造方式如图：
....
#.. 改造成 .#####.
... .##..#.
.......</td></tr>
<tr><td>其它</td><td colspan="2">X</td></tr>
<tr><td>PKU 1947</td><td>Flip Game</td><td>给你一颗无根树，求从这颗无根树里截取 P 个节点，最少要删几条边。</td></tr>
<tr><td>算法讨论</td><td colspan="2">这是一个无根树，所以我们任选一个节点为根。定义状态 F[I,J]代表在以 I 为根的子树里选 J 个节点的最小值，所以我们利用背包的思想，F[I,J]:=Min{F[I,J1]+F[Ch,J2] | J1+J2=J}。</td></tr>
<tr><td>其它</td><td colspan="2">时间复杂度为 O（N^3）。</td></tr>
<tr><td>PKU 1155</td><td>TELE</td><td>给你一颗以 1 为根的树，每条边有一个权值 p，每走一条边总费用减少 p，每个叶子节点也有一个权值 q，每访问到一个叶子节点总费用增加 q。求总费用不为负数的情况下最多能访问到多少个叶子节点。</td></tr>
<tr><td>算法讨论</td><td colspan="2">和 PKU 1947 没有什么两样，利用树型 DP+背包问题二次 DP 求解。定义 F[I,J]代表在以 I 为根的子树里选取 J 个叶子节点的最大值，所以 F[I,J]:=Max{F[I,J1]+F[K,J2]-Cost[I,K] | J=J1+J2}。</td></tr>
<tr><td>其它</td><td colspan="2">X</td></tr>
<tr><td>PKU 1692</td><td>Crossed Matchings</td><td>给你上下两排点，要求你将上面的点 a[i]和下面的点 b[j]进行匹配。1）a[i]=b[j]，且与 a[i']=b[j']这条匹配线相交，且这两条匹配线不再和任何匹配线相交，且 a[i]<>a[i']。 2）一个点只能引出一条匹配线。求大的匹配线数目。</td></tr>
<tr><td>算法讨论</td><td colspan="2">显然，数目必然是偶数。定义 F[I,J]代表上面前 I 个点下面前 J 个点进行互相匹配的最大值，所以 F[I,J]:=Max{F[I-1,J],F[I,J-1],F[MatchB[I,J]-1,MatchA[I,J]-1]+2}。MatchA[I,J]代表上面的点 I 在下面的前 J-1 个点最早相等的位置，MatchB 同理。</td></tr>
<tr><td>其它</td><td colspan="2">时间复杂度为 O（N^2）。</td></tr>
<tr><td>PKU 1112</td><td>Team Them Up!</td><td>给你 N 个点，若任意两个点之间互相有边，则可以分到一组。求将这 N 个点分成两组的方案，且要使</td></tr>
</table>

		得两组人数尽可能的接近。
算法讨论	我们先构图。若某两个点互相存在有向边，则连一条无向边。 此时，图就变成了一个无向图，接着将此图反过来，即原来无边变有边，有边的变无边。 这样，就构成了一堆连通分量，对其进行 01 染色，因为两个不认识的人之间连的是连有边的，所以若某点的颜色既染过 0 又染过 1 则无解。否则，我们记录 t0，t1 代表染 0 或 1 号颜色的人的数量。 接下来利用背包求解，定义 F[I,J]代表前 I 个连通分量，差值为 J 是否可行，中间记录路径即可。更加详细的见： http://www.cppblog.com/linyangfei/archive/2008/08/08/58295.html	
其它	时间复杂度为 O（N^2）。	
PKU 1946	Cow Cycling	N 头奶牛，每头牛初始有 E 的体力，他们合作骑车骑 D 圈。若领头的牛每分钟骑 X 圈，则领头的牛消耗 X^2 的体力，其他的牛消耗 X 的体力，任何牛都可以中途退出比赛。求每头牛消耗的体力不超过 E 的情况下，N 头牛合作最少需要几分钟骑完 D 圈。
算法讨论	定义状态 F[I,J,K]代表，前 I 头牛，骑 J 圈，消耗体力为 K 时的最优值，所以可以推出。1、F[I,J+P,K+P^2]=Min{F[I,J,K]+1}。2、F[I+1,J,J]=Min{F[I,J,K]}。初始 F[1,0,0]=0，求 F[N,D,K]。	
其它	时间复杂度为 O（NE*D^2），但实际上远远比这个复杂度小。	
PKU 3691	DNA repair	N 个串，一个主串，求在主串中最少更改多少个字母，可以使得 N 个串不在主串中出现。
算法讨论	先建立一颗具有“AC”性质的 Trie 树，新增一个 Boolean 域，用来判别其是否是危险节点。若 J 号节点是危险节点，当且仅当其是某串的结尾，或其 Fail 指针指向的节点是危险节点。然后进行动态规划，定义 F[I,J]代表在主串的 I 位置，在 Trie 树的 J 号非危险节点最少需要更改的字母数。所以，F[I,Trie[J].Son[K]]:={F[I-1,J]+Ord(K<>S[I])}，即从 J 号节点走向其儿子节点 son 的价值，若 son 与 S[I]相同，则不需要更改 son；否则，需要更改 son，代价+1。还需要注意的是，若某个节点的 son 为空，则在进行 bfs 的时候需要重新定向。	
其它	时间复杂度为 O（Len^2）。	
PKU 1625	Censored!	可以使用 N 个字符，组成一个长度为 M 的字符串，并且这个字符串不能包含一下 K 个字符串中的任何一个。求出一共有多少中组成方法。
算法讨论	一些处理方法见 http://www.cai0715.cn/read.php?190。本题与那个题的最大的不同之处在于 DP 方程，F[I,Trie[J].Son[K]]:=Sum{F[I-1,J]}。由于结果会非常大，所以需要使用高精度。还有，必须将字符映射进数组里，不然就会 RE。。。	
其它	时间复杂度为 O（N^2*M*K）。	

<table>
<tr><td>PKU 1015</td><td>Jury Compromise</td><td>有 N 个点，每个点有 A 和 B 两个权值，从中选取 M 个，使得 Abs（Sum{A} - Sum{B}）最小，若有多种方案，选取 Sum{A} + Sum{B}最大的。</td></tr>
<tr><td>算法讨论</td><td colspan="2">设 C[I]=A[I]-B[I]；
定义 F[I,J][K]代表，前 I 个点里选取 J 个差值为 K 的方案是否可以行。
定义 S[I,J][K]代表，前 I 个点里选取 J 个差值为 K 的方案若可行，和为多少。
定义 W[I,J][K]代表，前 I 个点里选取 J 个差值为 K 的方案若可以，是否选取第 I 个点。
所以 F[I,J,K]:=F[I-1,J,K] or F[I-1,J-1,K-C[I]]，然后更新 S[I,J][K]和 W[I,J][K]即可。
可能由于记录路径那里比较那啥，所以无法使用滚动数组，但是空间上还是完全过得去的。</td></tr>
<tr><td>其它</td><td colspan="2">时间复杂度为 O（NM*MaxAns）。</td></tr>
<tr><td>RQNOJ 490</td><td>石子合并</td><td>经典问题。</td></tr>
<tr><td>算法讨论</td><td colspan="2">先来介绍下四边形不等式。
设 I<=I'<=J<=J'，若满足 W[I,J]+W[I',J']<=W[I,J']+W[I',J]，则称 W 满足四边形不等式，则可以使用单调性对动态规划进行优化。
在本题中，
第一问，求最小值：
定义 F[I,J]代表在[I,J]这个区间内取得的最小值，有如下状态转移方程：
F[I,J]:=Min{F[I,K-1]+F[K,J]}+Sum[I,J] （I+1<=K<=J）
根据四边形不等式，我们可以进行如下优化：
P[I,J]:=Max{K | F[I,J]=Min{F[I,K-1]+F[K,J]}+Sum[I,J] }
F[I,J]:=Min{F[I,K-1]+F[K,J]}+Sum[I,J] （P[I,J-1]<=K<=P[I+1,J]）
初始 F[I,I]=0，P[I,I]=I （证明见实用算法新编）
第二问，求最大值：
定义 F[I,J]代表在[I,J]这个区间内取得的最大值，有如下状态转移方程：
F[I,J]:=Max{F[I,K-1]+F[K,J]}+Sum[I,J] （I+1<=K<=J）
根据四边形不等式，我们可以进行如下优化：
F[I,J]:=Max{F[I,J-1],F[I+1,J]}+Sum[I,J]
所以，整体时间复杂度从 O（N^3）变成了 O（N^2），此题还有 O（NlogN）的办法，不过思想会更难了。</td></tr>
<tr><td>其它</td><td colspan="2">时间复杂度为 O（N^2）。</td></tr>
<tr><td>PKU 1015</td><td>Jury Compromise</td><td>有 N 个点，每个点有 A 和 B 两个权值，从中选取 M 个，使得 Abs（Sum{A} - Sum{B}）最小，若有多种方案，选取 Sum{A} + Sum{B}最大的。</td></tr>
<tr><td>算法讨论</td><td colspan="2">设 C[I]=A[I]-B[I]；
定义 F[I,J][K]代表，前 I 个点里选取 J 个差值为 K 的方案是否可以行。
定义 S[I,J][K]代表，前 I 个点里选取 J 个差值为 K 的方案若可行，和为多少。
定义 W[I,J][K]代表，前 I 个点里选取 J 个差值为 K 的方案若可以，是否选取第 I 个点。
所以 F[I,J,K]:=F[I-1,J,K] or F[I-1,J-1,K-C[I]]，然后更新 S[I,J][K]和</td></tr>
</table>

	W[I,J][K]即可。 可能由于记录路径那里比较那啥，所以无法使用滚动数组，但是空间上还是完全过得去的。	
其它	时间复杂度为 O（NM*MaxAns）。	
PKU 1636	Prison rearrangement	有两个监狱，每个监狱里面有 n 个囚犯，现在希望交换 n/2 对囚犯。但是考虑有一些原本在不同监狱的囚犯对在一起是很危险的，所以希望经过交换后他们还是不在一个监狱里面。那么如果保证这个条件，希望尽可能多的交换囚犯。
算法讨论	把 2N 个囚犯看成两个点，M 对关系看成连接两个点的一条无向边。 显然，这个图会形成 K 个联通分量，对于每个联通分量记录两边的点各有多少个，T1 和 T2。 这样，会形成 K 个数对 T1,T2，我们要使得 Sum{T1}=Sum{T2}<=N DIV 2，最大的 T1 和 T2 就是 Ans，这个部分可以用背包来完成。	
其它	时间复杂度为 O（N^3）。	
PKU 1722	SUBTRACT	有 N 个数，一个缩写变换操作是将相邻的元素 ai 与 ai+1 用他们的差 ai-ai+1 进行替换，至多进行 N-1 次。问经过 N-1 次变换问，是否可以变换成某个特定的数 Goal，若可以则输出方案。
算法讨论	先将题目进一步抽象，即对 I 和 I+1 位置添加+或-，且 1 和 2 之间必为-，是否可以得到 Goal。 定义状态 F[I,J]代表前 I 个数是否可以得到 J，F[I,J]:=F[I-1,J+A[I]] or F[I-1,J-A[I]]。 记录路径后，得到一个+-的序列。 对于这个序列我们进行构造，即先考虑+位置，连续的+号段从头到尾输出，然后输出任意-位置。	
其它	时间复杂度为 O（NK）。	
PKU 1732	Phone numbers	有一个电话号码，和 N 个名字，求这个电话号码最少可以用几个名字组成，可以重复。
算法讨论	先把名字转成数字，存进 Trie 树中。 定义 F[I]代表前 I 个数字最少可以由几个名字组成，由如下方程： F[I]:=Min{F[J-1]+1} （S[J..I]是一个存在的名字）。 检查 S[J..I]的时用 Trie 树进行即可。	
其它	时间复杂度为 O（N^3）。	
PKU 1821	Fence	有 K（1<=K<= 100)个工人进行粉刷栅栏的工作，第 i 个工人能粉刷 L[i]个栅栏（必须连续），且必须包含第 P[i]个栅栏，粉刷每个栅栏的工费为 Cost[i]，问现在有 N 个栅栏（1<=N<=16000）需要进行粉刷，问怎样安排这 K 个工人进行粉刷，使得最终的总工费（各个工人的所得之和）最多？

<table>
<tr><td>算
法
讨
论</td><td colspan="2">先以 P 为关键字对工人进行排序，使这个顺序作为动态规划的阶段。
定义 F[I]代表粉刷前 I 个栅栏最大价值，有如下状态转移方程：
F[I]:=Max{F[J]+Len*Cost}
显然，这个方程的复杂度是 O（K*N*N）的，我们需要对其进行优化。
顺序枚举工人 I，递减的枚举右边界 J，然后定义一个变量 K，初始时 K=P[I]，代表第 I 个工人粉刷的左边界。
显然 F[J]:=Max{F[K-1]+（J-K+1）*Cost[I]}
现在的关键就在于维护这个 K。
当递减循环到 J'的时，设 T=J'-Len[I]+1，当((K-1)-T+1)*Cost[I]>F[K-1]-F[T-1]时，K:=T，否则 K 不变。（画画图就明白了）
所以，维护 K 只需要 O（1）的时间，整个 DP 复杂度就降到了 O（N^2）。</td></tr>
<tr><td>其它</td><td colspan="2">时间复杂度为 O（N^2）。</td></tr>
<tr><td>PKU 1949</td><td>Chores</td><td>N 个工作，给出做每个工作需要提前做的其他工作，求做完所有工作最少的时间。</td></tr>
<tr><td>算
法
讨
论</td><td colspan="2">由于题目给出的序列已经是拓扑序列，所以我们可以直接 DP。
定义 F[I]代表做完工作 I 最少需要的时间，有如下方程：
F[I]:=Max{F[I],F[J]+W[I]}
Ans=Max{Ans，F[I]}。</td></tr>
<tr><td>其它</td><td colspan="2">时间复杂度为 O（N*100）。</td></tr>
<tr><td>PKU 2127</td><td>Greatest Common Increasing Subsequence</td><td>给定两个串，求其公共的最长上升序列，输出长度和路径。</td></tr>
<tr><td>算
法
讨
论</td><td colspan="2">和普通的最长上升没有太大的区别。
定义 Last[I,J]代表，在第一个串 I 位置，第二个串前 J 位置，最近的一个和 A1[I]相等的元素地址。
例如，在样例中 Last[1,1]=0,Last[1,2]=2,Last[1,3]=2,Last[2,4]=4。
这个数组可以在 O（N^2）的复杂度内求出。
定义 F[I,J]代表，在第一个串前 I 位置，第二个串前 J 位置，最长公共上升序列是多少，有如下方程：
F[I,J]:=Max{F[K,Last[K,J]]}+1。
在得到最优值的过程中，记录路径，输出即可。
有一个小优化，即过滤掉只在一个数组中存在的数，这样可以提升不少的速度。</td></tr>
<tr><td>其它</td><td colspan="2">时间复杂度为 O（N^3）。</td></tr>
<tr><td>PKU 2176</td><td>Folding</td><td>一个字符串，重复的字串可以压缩，括号可以嵌套，求压缩后的最小长度，并且输出。</td></tr>
<tr><td>算
法
讨
论</td><td colspan="2">定义 F[I,J]代表（I,J）压缩后的最小长度，初始时，F[I,J]=J-I+1，有如下方程：
F[I,J]:=Min{F[I,J]，F[I,K]+F[K+1,J]}。
上面的方程很容易的可以推出来，下面考虑压缩的部分，有如下方程：
F[I,J]:=Min{F[I,J]，F[I,K]+Num{Tot}+2}，其中 F[I,K]重复 Tot 次可以得到 F[I,J]。
然后记录路径输出就行了。</td></tr>
</table>

<table>
<tr><td>其它</td><td colspan="2">时间复杂度为 O（N^3）。</td></tr>
<tr><td>PKU 2228</td><td>Naptime</td><td>奶牛**喜欢睡觉补充能量,他将时间分为 N 个时段。他将选择 M 个时段睡觉(M<=N)每个时段有一个补充能量的值，奶牛希望通过睡 M 个时段（不一定都连续）来获得最大的能量。不幸的是，如果奶牛选择 ai1,ai2,ai3....aik 这连续 k 段作为睡觉的时段(k<=M)，那么获得的能量就是 w(ai2)+w(ai3)+...+w(aik)，(ai1 作为奶牛的进入睡眠时间)。更不幸的是，你要解决的问题是环形的。</td></tr>
<tr><td>算
法
讨
论</td><td colspan="2">先不考虑环，定义状态 F[I,J][K]代表前 I 个时段用 J 个时段睡觉，当前时段是否睡觉（K=0 则不睡，K=1 则睡），有如下方程：
F[I,J][0]:=Max{F[I-1,J][0],F[I-1,J][1]}
F[I,J][1]:=Max{F[I-1,J-1][0],F[I-1,J-1][1]+A[I]}
初始状态 F[1,0][0]=0,F[1,1][1]=0，其它为-Maxlongint 这样
Ans=Max{F[N,M,0],F[N,M,1]}
下面考虑环，显然，若出现环，则 1 时段和 N 时段必然被选，所以定义初始状态 F[1,1][1]=A[1]，其它为-Maxlongint，Ans=F[N,M][1]。
这样，在两个动态规划里取一个最大值即可。</td></tr>
<tr><td>其它</td><td colspan="2">时间复杂度为 O（NM）。</td></tr>
<tr><td>PKU 2342</td><td>Anniversary party</td><td>N 个点的一棵树，每个点都有权值，从中选取一些点，使得这些点中任意两点不直接相连，求最大价值。</td></tr>
<tr><td>算
法
讨
论</td><td colspan="2">定义 F[I,0]代表，以 I 为根的子树，不选 I 节点的最大价值。
定义 F[I,1]代表，以 I 为根的字数，选取 I 节点的最大价值。
有如下方程：
F[I,0]:=Sum{Max{F[CH,0],F[CH,1]}}。
F[I,1]:=Sum{F[CH,0]}+V[I]。</td></tr>
<tr><td>其它</td><td colspan="2">时间复杂度为 O（N）。</td></tr>
<tr><td>PKU 2411</td><td>Mondriaan's Dream</td><td>有个 N*M 的棋盘，你可以在里放 1*2 的长方形，问放满整个棋盘有多少种方案数。</td></tr>
<tr><td>算
法
讨
论</td><td colspan="2">定义 F[I，J]代表第 I 行，使用 J 号状态时的方案数。
由于长方形为 1*2 的，所以只和上一层的状态有关，有如下方程：
F[I，J]：=Sum{F[I-1,K]} （K 状态能导出 J 状态）
下面的问题就转换成了，如何判断 K 状态是否能够导出 J 状态。
枚举 K 状态，利用 DFS，构造出可导出状态 J。
0 代表 1*2 的长方形和 2*1 的长方形上面的部分，1 代表 2*1 长方形下面的部分。
1、若 K 状态的第 P 位为 1，则 J 状态的第 P 位必然为 0。
2、若 K 状态的第 P 位为 0，则 J 状态的第 P 位可以为 1。
3、若 K 状态的第 P 位为 0，且可以放下 1*2 的长方形，且 K 状态的第 K+1 位不为 0，则 J 状态的第 P 位第 P+1 位可以同时放 0。</td></tr>
<tr><td>其它</td><td colspan="2">时间复杂度为 O（N*4^N）。</td></tr>
</table>

PKU 3280	Cheapest Palindrome	长度为 N 的字符串，可以增加或删除字符，使其构成回文，增加或删除不同的字符有不同的花费，求最小花费。
算法讨论	定义 F[I,J]代表将这个区间内构成回文的最小花费，有如下方程： F[I,J]:=Min{F[I,J-1]+Cost[J],F[I+1,J]+Cost[I]}，特别的当 S[I]=S[J]时， F[I,J]:=Min{F[I,J],F[I+1,J-1]}。 由于增加或删除性质是一样的，所以 Cost 取其最小值即可。	
其它	时间复杂度为 O（N^2）。	
PKU 2948	Martian Mining	一个 N×M 的网格，每个格子有两种矿石含量为 aij,bij。处理 a 矿石的工厂在地图的左边，处理 b 矿石的工厂在地图的上边。现在需要在地图上修若干条铁轨，每个格子的铁轨要么东西朝向，要么南北朝向，运输矿石的轨道不能拐弯。要求总共能运输多少矿石。
算法讨论	定义 F[I,J]代表以(I,J)为右下角的矩形最大能运输矿石量，有如下方程： F[I,J]:=Max{F[I,J-1]+SumB[I,J]，F[I-1,J]+SumA[I,J]} F[N,M]就是最大矿石运输量。	
其它	时间复杂度为 O（NM）。	
PKU 1925	Spiderman	有 N 根柱子，蝙蝠侠在第一根柱子上，要去第 N 根柱子。每次蝙蝠侠可以用网套在某根柱子上，进行飞跃。问去到第 N 根柱子，最少需要多少次飞跃。
算法讨论	通过题目可以知道，蝙蝠侠飞跃后必然只会停留在固定高度上，所以定义 F[I]代表蝙蝠侠在 I 位置的固定高度上，最少需要的跳跃次数。 我们通过循环每根柱子，来判断其最大可飞跃距离，即 Sqrt（B[I]^2-（B[I]-B[1]）^2），这样，我们可以通过 F[J]推出 F[2*A[I]-J]，即 F[2*A[I]-J]:=Min{F[J]+1}。 特别的，当 2*A[I]-J>=A[N]时，蝙蝠侠触碰到了第 N 根柱子，我们则可以更新 Ans。	
其它	看 AC 率和通过人数，应该是个难题了、、、不过等做完了会发现，其实是个很简单的题、、、Orz	
PKU 3034	Whac-a-Mole	在 N*N 的棋盘上进行打鼹鼠游戏，给出每个鼹鼠出现的坐标和时间，每一秒，你可以从一个位置直线移动到另一个位置，将直线上所有点内的老鼠打死，问最多能打多少只鼹鼠。
算法讨论	状态应该很容易找，定义 F[I,J,K]代表前 K 秒，当前锤子在(I,J)位置，最多能打的鼹鼠数量，有如下方程： F[I,J,K]:=Max{F[I2,J2,K-1]+Line（I,J,I2,J2,K）}，Line（I,J,I2,J2,K）代表第 K 秒这条直(I,J)(I2,J2)线上所有的鼹鼠数。 下面的问题就是如何求出 Line（I,J,I2,J2,K），分 4 种情况讨论： 1）I=I2,J=J2，则 Line（I,J,I2,J2,K）=A[I,J,K]。 2）I=I2,J<>J2 或 I<>I2,J=J2，则直接循环求直线上的。 3）I<>I2,J<>J2，根据斜率构造。	

<table>
<tr><td></td><td colspan="2">这样，Ans=Max{F[I,J,Time]}。
此题还有一个需要注意的地方，虽然鼹鼠是在（0..N-1,0..N-1）的范围内，但是并不一定锤子必须在这个范围内，所以我们需要扩大坐标范围进行 DP。</td></tr>
<tr><td>其它</td><td colspan="2">时间复杂度为 O（T*N^5）。</td></tr>
<tr><td>PKU 3254</td><td>Whac-a-Mole</td><td>N*M 的棋盘，每个格子不是 0 就是 1，1 代表可以种玉米，否则不能。相邻的两个格子不能同时种下玉米，问总方案数。</td></tr>
<tr><td>算法讨论</td><td colspan="2">由于范围很小，很容易就想到状态压缩动态规划，为了进一步降低复杂度，可以选择格递推的状态压缩动态规划。
用 State 数组来存储状态，F 数组来存储对应状态的方案数。

有如下方程：
1）不种玉米，将 State 的第 I 和 I+1 位设置成 0。
2）种玉米，当该点不是障碍点，且 State 第 I 和第 I+1 均为 0 时，将 State 的第 I 和第 I+1 位设置成 1。</td></tr>
<tr><td>其它</td><td colspan="2">时间复杂度为 O（NM*2^N）。</td></tr>
<tr><td>PKU 2486</td><td>Apple Tree</td><td>有一颗 N 个节点的苹果树，每个节点有一定的苹果数量，你最多可以走 M 步，每走一条边算一步，问最多可以吃到多少苹果，边和节点可以重复走，苹果不可以重复吃。</td></tr>
<tr><td>算法讨论</td><td colspan="2">一开始的想法是枚举一条路径作为不往回走的路径，然后简单的树型 DP，可是这样的复杂度高达 O（N^2*M^2），TLE。
于是寻找新的做法，先转成左兄弟右儿子表示法，然后我们这样定义状态。
F[0,I,J]，代表在以 I 为根的子树种走 J 步，不需要回来的最大价值；
F[1,I,J]，代表在以 I 为根的子树种走 J 步，需要回来的最大价值。
显然，有如下 4 种转移方式：
1）不选 I 节点。
2）选择 I 节点，不选右儿子节点。
3）选择 I 节点，不选左儿子节点。
4）选择 I 节点，左右儿子节点均选。
当 T=0 时
1）F[0,I,J]:=Max{F[0,Rch,J]}
2）F[0,I,J]:=Max{F[0,Lch,J-1]+A[I]}
3）F[0,I,J]:=Max{F[0,Rch,J-2]+A[I]}
4）F[0,I,J]:=Max{F[0,Lch,K]+F[1,Rch,J-3-K]+A[I],
F[0,Rch,K]+F[1,Lch,J-4-K]+A[I]}
当 T=1 时
1）F[1,I,J]:=Max{F[1,Rch,J]}
2）F[1,I,J]:=Max{F[1,Lch,J-2]+A[I]}
3）F[1,I,J]:=Max{F[1,Rch,J-2]+A[I]}
4）F[1,I,J]:=Max{F[1,Lch,K]+F[1,Rch,J-4-K]+A[I]}
Ans=F[0,1,M]。</td></tr>
</table>

其它	时间复杂度为 O（N*M^2）。	
PKU 3017	Apple Tree	有 N 个数字，你可以把它分成连续的若干份，每份的总和不得超过给定的 M，要求使得 Sum{每份的最大值的}最小。
算法讨论	定义 F[I]代表前 I 个数字分成若干份的最优值，有如下方程： F[I]:=Min{F[J]+Max[J+1,I]} 显然，直接用这个方程是会 TLE 的，所以我们需要优化。 设 P[I]为 F[I]取得最优值时的 J。 我们可以发现，当第 I+1 个数加入进来之后，如果 Sum[P[I],I+1]不超过 M，则有如下性质： 1）A[I+1]<=Max[P[I]..I]，那么显然 I+1 是可以忽略，直接加入上一次的区间，即 F[I]=F[I-1]，P[I]=P[I-1]。 2）A[I+1]>Max[P[I]..I]，那么显然 F[I+1]不会在（P[I]+1..I）这个区间内取得最优值，我们可以直接从 P[I]开始循环。 这样，复杂度会相当相当的低。	
其它	数据貌似弱了点、、、	
SGU 149	Computer Network	N 个节点的树，边上有权值，问树上每个点到其它的最大距离。
算法讨论	定义 F[0,I]代表以 I 节点为根的子树中到它的最大距离， F[1,I]代表以 I 节点为根的子树中到它的次大距离， F[2,I]代表向上扩展经过 I 节点的父亲的最大距离。 第一遍 DP，先求出 F[0,I]和 F[1,I]，并在取得最优值时，把 J 记录进 P 数组，有如下动态转移方程： F[1,I]:=Max{F[1,I],F[0,J]+Dis[I,J]} F[0,I]:=Max{F[0,I],F[1,I]} 第二遍 DP，求出 F[2,I]，有如下动态转移方程： F[2,I]:=Max{F[2,Fa[i]]，F[0,Fa[I]] (P[0,Fa[I]]<>I)，F[1,Fa[I]] (P[1,Fa[I]]<>I)}+Dis[I,Fa[i]] Ans[I]:=Max(F[0,I]，F[2,I])。	
其它	时间复杂度为 O（N+M）。	
SGU 183	Painting the balls	N 个球（N<=10000），每个球都是白色的，要求你把这些球中的某些涂成黑色，每个球涂成黑色的代价不同。最后，使得任 M 个连续的球中必须有两个黑色的球。球最小代价。
算法讨论	定义 F[I,J]代表前 I 个球，第 I 个球涂成黑色，第 I 个球后面的第 J 个球也涂成黑色的最小代价，有如下转移方程： F[I,J]:=Min{F[J,K]}+A[I] （1<=J<=M-1，1<=K<=M-J） 这样写，时间复杂度为 O（NM^2），我们需要优化。 定义 G[I,J]代表前 I 个球，把第 I 个球涂成黑色，第 I 个球后面 J 个球中某个涂成黑色的最小代价，有如下转移方程： G[I,J]:=Min{G[I,J-1],F[I,J]} 所以，F[I,J]:=G[J,K]+A[I] （1<=J<=M-1，K=M-J）。	

	时间复杂度为 O（NM），使用滚动数组，空间复杂为 O（M^2）。
其它	时间复杂度为 O（NM）。

其它试题：

题目来源	题目名称	题目大意
BOI 2003	团伙	经典问题，有敌人的并查集。
算法讨论	如果是朋友则直接合并。如果是敌人，显然，一个人对应的敌人集合必定只有一个，所以，我们用 e 数组记录这个集合的代表元素，之后再和此元素是敌人的，将其与代表元素合并即可。	
其它	X	
PKU 1151	Atlantis	给你 N 个矩形，求这 N 个矩形的面积。
算法讨论	其实应该是用离散化+线段树来做，但是我邪恶的用了指针。。。方法就是离散化+枚举。	
其它	时间复杂度为 O（NLogN）。	
PKU 2299	Ultra-QuickSort	给定 N 个数，可以任意交换相邻的两个数，最后使其变成升序，问需要交换多少次。
算法讨论	这个题目的模型就是个逆序对。可以使用归并排序来做，这个就不再说了。这里说另一种求逆序对的方法，利用树状数组，先离散化，然后按顺序插入，每次查找之前插入的比它大的数的个数即可。	
其它	时间复杂度为 O（NLogN）。	
PKU 2406	Power Strings	给你 N 个字符串，对于每个字符串，看其是由几个重复子串构成的，例如，abcd，是由 1 个 abcd 构成的；aaaa，是由 4 个 a 构成的，ababab，是由 3 个 ab 构成的。
算法讨论	根据 P 数组的性质，S[1..P[I]]=S[I-P[I]+1..I]，也就是说 S[1..Len-P[Len]]才有可能是该串的重复子串，所以如果 P[Len] mod Len-P[Len]=0，则代表 S[1..Len-P[Len]]是重复子串，否则重复子串为 S。	
其它	时间复杂度为 O（N）。	
PKU 1961	Period	给你若干个字符串，对于每个字符串，分解成 N 个前缀子串。对于每个前缀子串，看其是由几个重复子串构成的。例如，aabaabaab，前缀子串 aa，是由 2 个 a 构成的；前缀子串 aabaab，是由 2 个 aab 构成的，前缀子串 aabaabaab 是由，3 个 aab 构成的。

算法讨论	和 PKU 2406 没有什么太大区别。	
其它	时间复杂度为 O（N）。	
HDOJ 2222	Keywords Search	N 个串，一个主串，为 N 个串有几个出现在了主串中。
算法讨论	AC 自动机，通俗一点说，就是一个建立在 Trie 上的 KMP，拥有一个失败指针，Fail。 假设有一个节点 k，他的失败指针指向 j。那么 k,j 满足这个性质：设 root 到 j 的距离为 n，则从 k 之上的第 n 个节点到 k 所组成的长度为 n 的单词，与从 root 到 j 所组成的单词相同。 算法流程： 1）将 N 个串存入 Trie 中。 2）利用 Bfs 求出 Fail 指针。 3）利用 AC 自动机，对主串进行匹配。	
其它	时间复杂度为 O（Len*K），K 为常数。	
PKU 1635	Subway tree systems	给出你两个有根树的括号序列，判断两颗树是否同构。
算法讨论	很显然，我们将两棵树用最小表示法构造出新的括号序列，判断其是否相等即可。 如何对树使用最小表示法？我们先用 DFS 构造出这棵树，在构造的途中，记录每个节点其儿子的括号序列（0 为左括号，1 为右括号）。 快速排序后便构造出了这个节点的最小表示。 最后，整棵树的最小表示也就求了出来。	
其它	X	
PKU 2019	Cornfields	在一个 N*N 的矩阵里，有 Ask 个询问，每次给定对坐标（X,Y），问你以（X,Y）为左上角的 B*B 正方形中，最大值-最小值是多少？
算法讨论	先进行预两遍处理。 竖向扫描，定义 Fmin[I,J]代表以（I,J）为顶端向下 B 个格中的最小值，Fmax 同理。 横向扫描，定义 F[I,J]代表以（I,J）为左上角的 B*B 正方形中最大值-最小值是多少。 上面两个数组都可以在 O（N^3）时间内很容易的求出，所以整体时间复杂度是 O（N^3）。 如果用单调队列来实现上面两个预处理，时间复杂度可降至 O（N^2*LogN），但是编程复杂度会比较高。	
其它	时间复杂度为 O（N^3）。	
PKU 1019	Number Sequence	有这样一个序列，1，112，112123，1121231234，112123123412345...，求它的第 X 位是多少。

算 法 讨 论	构造，分步骤来求，逐步逼近答案。 先对序列进行分组，即按 1，12，123，1234，12345 这样分组，利用递推求出前 I 组数的位数，即 Sum[I]:=Sum[I-1]+F[I]， F[I]:=F[I-1]+Trunc(ln(I)/ln(10))+1。 这样，我们可以很容易确定第 X 位的所在组（当组大于 40000 时，位数已经超过 Maxlongint） 然后在用 While 循环求出第 X 位是这个组中的第几个数，再求出是这个数的第几位即可。	
其它	X	
PKU 2029	Get Many Persimmon Trees	给定 N*M 的矩形和 T 个点，让你求 A*B 的子矩形最多能覆盖几个点。
算 法 讨 论	定义 Row[I,J]代表以从(I,J)开始，向左数 B 个格子，总共能覆盖多少个点。 定义 Col[I,J]代表从(I,J)开始，向上数 A 个格子，每个格子向左数 B 个格子即 Row[I,J]，总共能覆盖多少个点。 Ans：=Max{Col[I,J]}	
其它	时间复杂度为 O（NM）。	
SGU 164	Airlines	一个 N 个点的无向完全图，每条边都被染成[1, M]的一种颜色，要求选择一部分颜色，作为一个集合 S，且 S 中的元素个数不超过[(M+1) / 2]，并满足性质：图中任意两个点 a 和 b，一定存在一条长度不大于 3 的通路，其每条边的颜色都属于 S。求这样一个 S 集合。
算 法 讨 论	将所有颜色分为两组，前[(M+1) / 2]个颜色为一组，剩下的颜色为一组，这两组颜色中，必有一个满足条件。 下面引用何林的证明： 证明：如果 A 集合符合题目要求，则命题成立，否则假设 A 集合不符合题目要求，也就是存在两个点 a,b，他们的距离大于 3、或者两者无法互相到达。 下面考虑 B 集合。 对于任意点对(x,y)，如果 A，必有∈B，所以在 B 中的距离<=3。 对于任意点对(x,y)，∈A，如果 a 在 A 中不和 x,y 中的任意一个有直接连边，则在 B 中存在；如果 b 在 A 中不和 x,y 中的任意一个有直接连边，则在 B 中存在。因此可以假设 a, b 在 A 中都和 x 或 y 有直接联系。 这样在 A 中就存在或类似的路径，长度不超过 3。这和我们的假设前提“a,b 在 A 中的距离超过 3 或者不连通”矛盾。 因此不可能存在 a,b 在 A 中都和 x, y 有直接联系这种情况。 所以不论∈A 还是 A，他们在 B 中的距离都不超过 3。所以，如果 A 不满足题目要求，B 必然满足要求。 命题得证明。	
其它	时间复杂度为 O（N^3）。	
SGU 165	Basketball	一列篮球队员身高从 1950mm 至 2050mm 不等，而他们的平均身高却正好是 2000mm。要求找到这些队员的一个排列，满足对任意 K 任意连续 K 个球员，

		他们的总身高和 2000*K mm 相差不超过 10cm：即 100mm。
算法讨论	由于每个人的身高在 1.95-2.05 之间，所以我们先把每个人的身高减去 2 备用。 设 Sum[I]代表前 I 个队员的身高和，此题要求的就是对于任意的 I 所以，我们只要对于任意的 I，使得 Abs（Sum[I]）<=0.05 即可。 这样，问题就转换成了维护一个前缀和。 假设已知 Sum[I-1]，若 Sum[I-1]<=0，则我们把一个非负数数放到第 I 个位置，反之把一个非正数放到第 I 个位置。 这样，就可以保证 Abs（Sum[I]）<=0.05。	
其它	时间复杂度为 O（N）。	
SGU 177	Square	N*N（N<=1000）的矩形中，给出 M 个染色信息，问最后多少个格子是白颜色。
算法讨论	此题若用线段树可以优化达到 O（N*LogN^2）的复杂度，但是写起来会比较麻烦。 所以，可以这样写。 我们从后往前染色，采用类似路径压缩的思想，若对区间(a,b)进行染色，当某个格子被染色后，下一次经过它直接跳到 b+1 的位置。 这样，每个格子至多被染色一次，复杂度为 O（N^2）。	
其它	时间复杂度为 O（N^2）。	
SPOJ 744	Longest Permutation	长度为 N 的串（N<=100000），求一个连续的最长的 1..M 的排列。
算法讨论	定义 Sum［I］代表前 I 项和。 定义 Last［I］代表在之前出现同样数字的位置。 定义 Next［I］代表在之后出现同样数字的位置。 由于排列必然包含 1，所以我们枚举每个 1 的位置，从它向左右扫描，即范围在 Last［I］到 Next［I］之间。 假设排列中的最大值在 I 的左边，当扫描到左面的 J 位置时，更新 P:=Min{P,Next［I］}，Q:＝Max{Q,A［I］}。 当一个排列满足条件，则必然 J+Q－1=I，且 Sum[J+Q-1]-Sum[J-1]=Q*(Q+1) Div 2，然后更新 Ans:=Max{Ans,Q}。 由于每个点至多被左扫一次右扫一次，所以复杂度为 O（N）。	
其它	时间复杂度为 O（N）。	
PKU 2452	Sticks Problem	有 N 个数（N<=50000），求这样的 I，J，A[I]比 A[I+1]..A[J]都要小，A[J]比 A[J-1]..A[I]都要大。最后使得 J-I 最大，输出。
算法讨论	设当前区间为[L,R]，可以推出这样一个结论。 当前区间的最大值只能作为所求区间的 J 位置，在[L,J]区间的最小值作为 I 位置是当前区间最优的。 所以，再分别处理[L..I-1]和[J+1..R]即可。	
其它	时间复杂度为 O（NLogN）。	

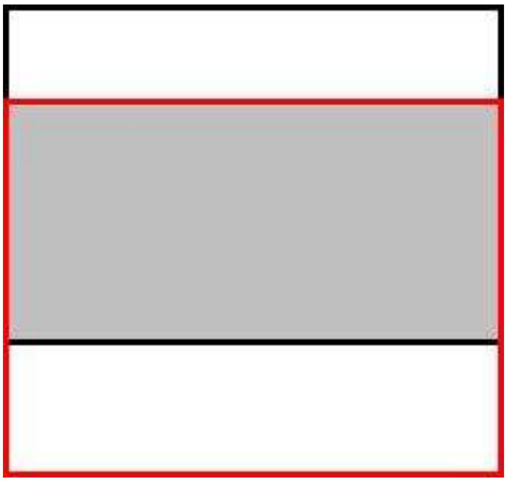
竞赛原题：

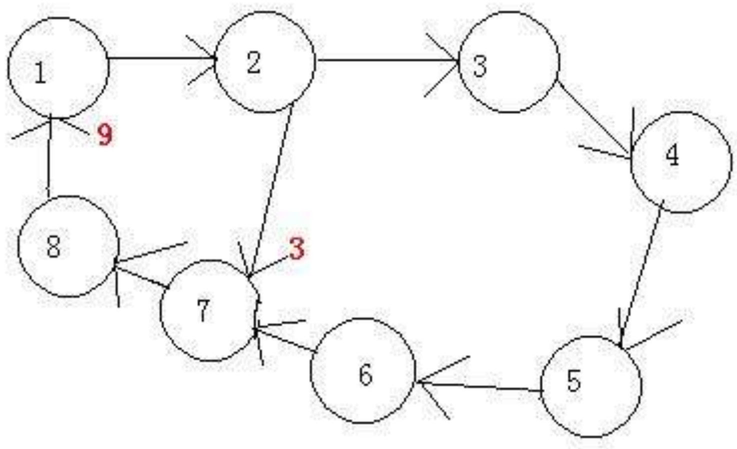
<table>
<tr><th>题目来源</th><th>题目名称</th><th>题目大意</th></tr>
<tr><td>IOI 1999</td><td>TRAFFIC LIGHTS</td><td>有 N 个路口，每个路口有一个红绿灯，从 I 路口到 J 路口当且仅当两路口的灯颜色是一样的。求从起点路口到达终点路口的最短距离。</td></tr>
<tr><td>算法讨论</td><td colspan="2">先抛开红绿灯，就是一个赤裸裸的最短路。然后我们加上红绿灯，我们可以发现，越早到一个路口越是好的，即便需要等待。所以，我们将最短路的松弛操作稍加改造即可，Dis[I]:=Dis[J]+G[J,I]+Wait[J,I,Dis[J]]，其中 Wait[J,I,Dis[J]]代表在 Dis[J]时间到达 J 路口，想到达 I 路口需要等的时间。</td></tr>
<tr><td>其它</td><td colspan="2">X</td></tr>
<tr><td>CSTC 2000</td><td>丘比特的烦恼</td><td>N 男 N 女（N<30）在一个平面上，给定他们的坐标和他们之间的缘分值。任意两点之间可以匹配的条件是，两点之间的距离不得大于给定值 K，且两点之间连线的线段上不能有第三者插足（这个词、、、），求一个完全匹配使得缘分值最大。</td></tr>
<tr><td>算法讨论</td><td colspan="2">显然，这个题是一个半裸奔二分图的最佳匹配题。
根据题目中给定的条件，连出可以进行匹配的边，然后进行二分图匹配或者最小费用最大流即可。</td></tr>
<tr><td>其它</td><td colspan="2">X</td></tr>
<tr><td>CSTC 2000</td><td>快乐的蜜月</td><td>N 个区间，每个区间有一个价值，选择一些不相交的区间，使得最后获利是第 K 大。</td></tr>
<tr><td>算法讨论</td><td colspan="2">如果是求最大，显然只用保存最大的值进行动态规划。
但是求第 K 大，那就保存 K 个最优值进行动态规划，复杂度 O（RK）。</td></tr>
<tr><td>其它</td><td colspan="2">X</td></tr>
<tr><td>IOI 2000</td><td>邮局</td><td>有 N 个点，让你设置 M 个邮局。使得每个点到邮局的距离定义为离其最近的邮局的距离，求所有点到邮局的距离和，输出最小值。</td></tr>
<tr><td>算法讨论</td><td colspan="2">定义 W[I,J]代表，在[I,J]这个区间内设置一个邮局的代价，显然要使这个代价最小，肯定是在中位点设置邮局。
定义 F[I,J]代表，前 J 个点设 I 个邮局，有如下状态转移方程：
F[I,J]:=Min{F[I-1,K]+W[K+1,J]} （I<=K<=J-1）
但是，这个算法是 O（N^3）的，对于加强版数据 N,M<=2000，是无法承受的。
其实，我们可以发现，F[I,J]是满足四边形不等式的，所以，我们可以将方程优化成：
F[I,J]:=Min{F[I-1,K]+W[K+1,J]} （P[I-1,J]<=K<=P[I,J+1]）</td></tr>
</table>

	时间复杂度降低为 O（N^2），对于极限数据也可以顺秒。	
其它	时间复杂度为 O（N^2）。	
NOI 2001	炮兵阵地	给你个棋盘，有障碍，可以在非障碍点放置士兵，但是其上下左右 4 个方向的两个格子范围内不能再放士兵，求最多可以放多少士兵。
算法讨论	由于 M<=10，所以我们先枚举出 2^M 种可能不互相的冲突，存进 b 数组。然后以行为阶段进行 DP，定义状态 F[I,S,S1]代表，第 I 行用 S 号放置方案，第 I-1 行用 S1 号放置方案的最大放置数，Tot[S]代表 s 状态下的士兵数，F[I,S,S1]=Max{F[I-1,S1,S2]+Tot[S]}。总复杂度为 O（N*60^3），但是由于冲突的出现以及全部利用位运算的优化，复杂度远远达不到这个数。	
其它	X	
NOI 2002	贪吃的九头龙	给你颗树，要求你给树染色，这颗树的根节点必须是 0 号颜色，且必须有 K 个节点是 0 号颜色，若某边的两端是同一颜色则增加一个难受值，若有方案则输出这个全部节点染色完毕后最小的难受值。
算法讨论	先将树转成左儿子右兄弟的二叉树。 对于 M>2 的情况，我们可以知道，只需要计算 0 号颜色（即大头）相邻情况下的难受值即可，因为其他节点完全可以用不同的颜色交错进行染色，不会出现重复。 所以我们定义状态 F[I,J,K]代表以 I 为根的子树中 J 个点染 0 号颜色，其父节点的颜色是 K（K=0 或 1，0 代表 0 号颜色，1 代表其他颜色），所以状态转移方程如下：F[I,J,K]:=Min{F[Lch,P,0]+F[Rch,J-P-1,K]+D[K,0]*Const[Fa,I] , F[Lch,P,1]+F[Rch,J-P,K]}，其中 D[0,0]=1，其他为 0，Const 代表两点之间的权。这样，从叶子节点往回返回的时候就可以得到正确解了。 对于 M=2 的特殊情况，由于只有两种颜色，假设为 0 和 1，所以状态转移方程后半截有点变化，即 F[Lch,P,1]+F[Rch,J-P,K]+D[K,1]*Const[Fa,I]，D[0,0]=1，D[1,1]=1，其他为 0。	
其它	时间复杂度为 O（NK^2）。	
NOI 2003	逃学的小孩	在一颗树上，有 ABC 三个点，A 为起点，AB<AC，求 AB+BC 的最大值。
算法讨论	这个题目解决的很巧妙（具体的见国家集训队 2007 年陈瑜希的论文），我们利用两次 DFS 来求解。第一遍 DFS 记录了其儿子到它的距离最大值和次大值，并且记录来源。第二次 DFS，依次考虑每个点 x 为汇聚点 S（因为 A 到 B，B 到 C 必然有且只有一个汇聚点）时的情况。利用第一次 DFS 得出的结论，求出三个最值，分别为 x 的子树到 x 最大值次大值，在从其父亲为根的树里找一个最大值（若来源不是 x，否则为次大值）。这样，我们就求出了三个最值，也就是三条路径。排序后记做 T1,T2,T3，则 AS=T3，SB=T2，SC=T1，所以 Ans=Max{T1+2*T2+T3}。	
其它	X	

NOI 2003	智破连环阵	有 A 和 B 两个国家，A 国用导弹打 B 国的连环阵，每一枚导弹可以打某范围内所有连续的连环阵，且想打 I 号连环阵必须保证 I-1 号连环阵已被消灭，求最少需要发射几枚炮弹。
算法讨论	DFS，设置状态为发射了几枚导弹，打到了哪里。利用最优性减枝、可行性减枝、调整搜索顺序，可以过 5 组数组。 部分搜索，有 A 和 B 两国，我们可以通过搜索，将 A 国可以分成 X 段，然后利用二分图匹配，X 段和 N 个导弹进行匹配，若能匹配上则可行。优化 1：最优性减枝。 优化 2：利用 BFS 计算可扩展长度。 具体的见 WC2004，楼天成的论文。	
其它	X	
IOI 2005	River	一颗树，0 号节点是伐木场，还可以建立 M 个伐木场，求点到最近的伐木场的距离*点包含的木材数的最小值。
算法讨论	转成左儿子又兄弟。定义状态 F[I,J,K]代表在以 I 为根的子树里放 J 个伐木场，离 I 最近的伐木场是 K。所以，F[I,J,K]:=Min{F[Lch,P,I]+F[Rch,J-P-1,K] , F[Lch,P,K]+F[Rch,J-P,K]+Cost[I,K]}。接下来就是递归求解了。	
其它	时间复杂度为 O（N^3*K）。	
IOI 2005	Garden	给定一个 N*M 的矩阵，每个位置有一个价值。在里面，给定一个 AN*AM 的大矩形和一个 BN*BM 的小矩形，AN*AM 必须包含 BN*BM，且小矩形不能贴在大矩形的边界上，求 Max{大矩形的总价值-小矩形总价值}。

算法讨论	数据范围是 N,M<=250，时限是 0.5s。
	显然，我们的直观思想是直接枚举两个矩形，但是复杂度相当相当高。
	我们扩宽思路，深度挖掘题目，其实可以转换成枚举所有单个矩形，选择两个不相交的。
	定义 Up[I]代表从第 I 行往上一个满足条件的小矩形的最小边长；
	Down[I]代表从第 I 行往下一个满足条件的小矩形的最小边长；
	Left[I]代表从第 I 列往左一个满足条件的小矩形的最小边长；
	Right[I]代表从第 I 列往右一个满足条件的小矩形的最小边长。
	所以，答案就是 Min{Up[I]+Down[I+1],Left[I]+Right[I+1]}。
	现在的关键一步就是如何求出这几个数组，也就是说如何高效的求出单个满足条件的矩形。
	显然 O（N^4）的算法是很容易想到的，经测试可以过 29 组数据，有 7 组会 TLE。
	我们先枚举矩形的左边界 J1，右边界 J2，上边界 I1，下边界 I2。预处理出 T[I]代表在 A[I,J1]+...+A[I,J2]。
	即 For J1:=1 to M do
	For J2:=J1 to M do
	For I1:=1 to N do
	For I2:=I1 to N do	

	 通过这幅图我们可以发现，每次 I1 往下走一格，上一次计算的结果和本次要计算的结果是有一定重复的。 所以，我们完全没必要重新计算，即没必要 I2 再次从 I1 开始循环，只用在 I1 往下走一格的时候减去 T[I1]即可。 所以，复杂度降到了 O（N^3）。	
其它	时间复杂度为 O（N^3）。	
WC 2006	水管局长	有 N 个点 M 条边无向简单图，Q 次询问，询问有两种类型。1）求从 X 走到 Y 的的代价，即任一条 X 到 Y 路径上最大权值的最小值得。2）某条边无法继续使用。询问是按时间顺序发生的，任意时刻保证图永远是连通。对于所有 1 号询问进行输出。
算法讨论	正向求解的话，比较难，所以我们不如倒过来求解。 所以，题意变成，初始时某些边是无法使用的，随着时间的推移，某些边恢复了使用，对于所有 1 号询问进行输出。 我们先对可以使用的点建立一颗最小生成树。 之后，对于某些边的恢复使用，则在 MST 中添加这条边，必然会产生一个环，只要把这个环上的权值最大的一条边删除，就又可以保证这是一颗最小生成树。 找最大边的话，就用朴素算法找他们的最近公共祖先，期望复杂度应该在 O（LogN）左右。 所以，整体复杂度应该在 O（QLogN）左右。（不知道估计的对不对，我写的程序跑了 0.6 秒，应该差不多）	
其它	时间复杂度为 O（QLogN）。	
IOI 2006	Writing	给定一个模板串 T，和一个主串 S，问 T 的排列在 S 中出现过几次。例如，cAda 的排列在 AbrAcadAbRa 中出现了两次，分别是 Acad、cadA。
算法讨论	先预处理出 T 中每个字母的出现次数 A，然后在 S 中逐个字符向前推。	
其它	时间复杂度为 O（N）。	

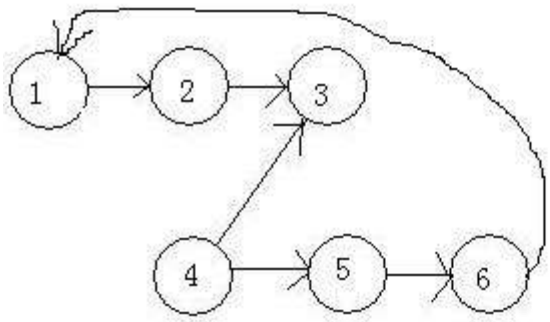
<table>
<tr><td>IOI 2006</td><td>Pyramid</td><td>给定一个 N*M 的矩阵，每个位置有一个价值。在里面，给定一个 AN*AM 的大矩形和一个 BN*BM 的小矩形，AN*AM 必须包含 BN*BM，且小矩形不能贴在大矩形的边界上，求 Max{大矩形的总价值-小矩形总价值}。</td></tr>
<tr><td>算法讨论</td><td colspan="2">直观的思想就是枚举大矩形的位置，然后在里面枚举小矩形的位置。可以这样的复杂度高达 O（N^2*M^2），对于 N 和 M 都高达 1000 的数据范围，显然不行，所以我们需要优化。通过枚举大矩形，我们可以发现，里面好多的小矩形都重复枚举了，做了好多无用功，所以我们就在这里进行优化。
定义 Matrix[I,J]代表，以（I,J）为左上角的小矩形的价值。
定义 ColMin[I,J]代表，从（I,J）为起点往下数，共 I+an-2-bn+1 个位置的矩阵价值最小值，即 ColMin[I,J]=Min{Matrix[I..I+an-2-bn,J]}。an-2-bn+1 的意思是说，在大矩形的内部纵向上总共可以放几个矩形。
定义 RowMin[I,J]代表，从（I,J）为起点往右数，共 J+am-2-bm+1 个位置的矩阵价值最小值，即 ColMin[I,J]=Min{ColMin[I,J+am-2-bm]}。am-2-bm+1 的意思是说，在大矩形的内部横向上总共可以放几个矩形。
所以，这样，RowMin[I,J]就代表了，在以（I-1,J-1）为左上角的大矩形内部放小矩形的最小值。
所以 Ans=Max{以（I,J）为左上角的大矩形价值-RowMin[I+1,J+1]}。
如果使用线段树或平衡树等数据结构，可以在 O（NMLogN）内算出 RowMin[I,J]，当然也可以用一种 O（NM）的预处理方法。</td></tr>
<tr><td>其它</td><td colspan="2">时间复杂度为 O（NM）。</td></tr>
<tr><td>NOI 2008</td><td>假面舞会</td><td>有 N 个人，每个人带一个面具，每个面具有一个编号，假设总共有 K 个编号。其中带 I 号面具的人可以看见带 I+1 号面具的人，带 K 号面具的人可以看见带 1 号面具的人。且 K 必然大于等于 3，求 K 最大是多少，最小是多少。若 K 怎样都无法大于 3，则输出-1。</td></tr>
<tr><td>算法讨论</td><td colspan="2">对于输入中 X 可以看见 Y，则我们从 X 向 Y 连一条有向边。
我们的第一印象是找所有连通分量中的环和最长链，然后我们得到一个算法。若存在环，则必然存在这样的冲突。


1 号节点初始被标记为 1，但是又被 8 号节点标记为 9。</td></tr>
</table>

<table>
<tr><td></td><td colspan="2">7 号节点已经被 6 号节点标记为 7，但是又被 2 号节点标记为 3。
但是每个节点必然只有一种编号，所以我们可以通过寻找一个 P，使得 1 MOD P=9 MOD P，7 MOD P=3 MOD P，显然这里 P=4，所以这个环中面具的最大种类数是 4。

设最大面具数为 Max，最小面具数为 Min。

所以，Max=所有连通分量中面具种类的最大公约数；Min=所有连通分量中面具种类不小于 3 的的最小公约数。
若不存在环，则寻找最长链，Max=所有连通分量中最长链的长度和；Min=3。

此算法正确吗？？？？
其实再思考思考，可以发现，单纯的这样找，是不正确的。
例如：


显然 Max=4，1 的编号为 1，2 的编号为 2，4 的编号为 2，3 的编号为 3，5 的编号为 3，6 的编号为 4。
但是按照刚才的方法则 Max=3。
所以本题的关键就在于，同类点需要被合并，即同被一个点指向或同指向一个点的同类点需要被合并。例如，图中 2 和 4 是同类顶点，3 和 5 是同类顶点，所以我们必须将 2 和 4，3 和 5 进行合并。
单纯的合并是很麻烦的，所以我们在原图上建立反向边，每次 DFS 添加一个 Deep 值，走正向边加 1，反向边减 1。
利用这个关系，再按上面的方法，就可以得到正确结果。</td></tr>
<tr><td>其它</td><td colspan="2">时间复杂度为 O（M）。
做这个题，千万要三思而后做，理清题目意思，挖掘题目更深度的意思，不要被表面现象所迷惑。</td></tr>
<tr><td>NOI 2008</td><td>设计路线</td><td>有 N 个城镇，N-1 条公路，1 号节点是首都，即根节点。每走一条公路，不舒适值会加 1，所以我们可以在城市间建造铁路，但是一个城市只能位于一个铁路中，每条铁路仅能经过每个城市一次。求所有城镇到首都的最大不舒适值的最小值，且求出有多少种方案。</td></tr>
</table>

<table>
<tr><td>算
法
讨
论</td><td colspan="2">由于三叉树才能增加 1 点不舒适值，所第一问的答案必然不会超过 10。
所以，我们可以通过枚举第一问的答案，然后计算方案数，如果方案数大于 0 则可行，输出。
定义，F[I,0,K]代表在 I 号城镇，最大不舒适值为 K，不修铁路的方案数。
F[I,1,K]代表在 I 号城镇，最大不舒适值为 K，修 1 条铁路的方案数。
F[I,2,K]代表在 I 号城镇，最大不舒适值为 K，修 2 条铁路的方案数。
定义，T1=F[J,0,K]+F[J,1,K]，代表 J 是 I 的一个子节点，在 I 和 J 之间修铁路的方案数。
T2=F[J,0,K-1]+F[J,1,K-1]+F[,J,2,K-1]，代表 J 是 I 的一个子节点，不在 I 和 J 之间修铁路的方案数。

所以我们可以得出如下方程：
对于每一个 J， F[I,2,K]:=F[I,2,K]*T2+F[I,1,K]*T1，即不和 J 连接的方案数+和 J 连接的方案数。
F[I,1,K]:=F[I,1,K]*T2+F[I,0,K]*T1，同上。
F[I,0,K]:=F[I,0,K]*T2。
初始时，F[I,0,K]=1。</td></tr>
<tr><td>其它</td><td colspan="2">时间复杂度为 O（NLogN^2）。</td></tr>
<tr><td>NOI 2008</td><td>志愿者招募</td><td>有 N 天，每天需要 A[I]个员工，有 M 个员工，可以从 S[I]天工作到 T[I]天，费用为 C[I]，求满足每天的需求 A[I]的条件下，最少需要多少费用。</td></tr>
<tr><td>算
法
讨
论</td><td colspan="2">难点在于利用不等式构图，接下来就是最小费用最大流了。

由于还不是非常非常明白，所以推荐去这里看题解：
http://www.byvoid.com/blog/noi-2008-employee/</td></tr>
<tr><td>其它</td><td colspan="2">Day 1 的压轴题，的确很难。</td></tr>
</table>

成套试题：

<table>
<tr><th>题目来源</th><th>题目名称</th><th>题目大意</th></tr>
<tr><td>重庆省选 2009</td><td>中位数</td><td>有 N 个数，且这 N 个数各不相同。给你一个数 K，问这 N 个数中有多少个连续子序列的中位数是 K。</td></tr>
<tr><td>算法讨论</td><td colspan="2">根据原数列，我们构造一个新数列。比 K 大的赋值为 1，比 K 小的赋值为-1，K 赋值为 0，则题目转化为求连续子序列和为 0 的子序列个数。设 F[I]代表，1）若 I<=K 则 F[I]为 A[1]..A[I]的和；2）若 I>K，则 F[I]为 A[K+1]..A[I]的和。我们将 I>K 的 F[I]进行计数，存进 Tot 数组。接下来，枚举 I（I < K）作为子序列的起点，累加 Tot[-(F[K]-F[I-1])]即可。</td></tr>
<tr><td>其它</td><td colspan="2">时间复杂度为 O（N）。</td></tr>
<tr><td>重庆省选 2009</td><td>叶子的颜色</td><td>有一颗 M 个节点的无根树，可任意选根（不能为叶子节点），其中有 N 个叶子节点，可以对任意节点染 0 或 1 号颜色，叶子节点的最终颜色则为根到它的最后一个颜色，已知每个叶子节点的最终颜色，求最少需要给几个节点染色。</td></tr>
<tr><td>算法讨论一</td><td colspan="2">贪心。首先，我们先选择根，很显然，树的直径最中间的点当根一定比其它点当根要好。选择出根后，我们从叶子节点往上推，考察每一个节点的染色情况，先初始化叶子节点的颜色为最终颜色。若某个节点 T 的儿子节点 0 号颜色比 1 号颜色，1）多，则 T 染为 0 号颜色，其儿子中的 0 号颜色变为无色；2）少，则 T 染为 1 号颜色，其儿子中的 1 号颜色变为无色；3）相等，若 T 不为根则不染色，否则随意染色，将任意儿子节点某颜色变为无色。</td></tr>
<tr><td>算法讨论二</td><td colspan="2">动态规划。先枚举一个根，对每个点有 3 个状态：黑、白、无色，表示当前节点在这个状态下要满足题意需要的代价。每个叶子节点涂对应颜色的代价为 1，其余为无穷大。一个节点，例如涂黑色，那么代价就是每个 min(儿子涂黑色-1,儿子涂白色,儿子不涂)的和加 1。如果不涂色，就是每个 min(儿子涂黑色,儿子涂白色,儿子不涂)的和。到最后只要用枚举根的三个值里的最小值去更新全局答案就可以了。O(n^2)的算法会超时。实践发现任选节点作为根的答案是一样的。当然也可以进行优化，能够在枚举根的情况下优化到 O(n)级别。我没有具体实践过，留给读者思考。</td></tr>
<tr><td>其它</td><td colspan="2">时间复杂度为 O（N）。</td></tr>
<tr><td>重庆省选 2009</td><td>跳舞</td><td>有 N 个男生和 N 个女生,给出他们互相喜欢的关系，若他们互相喜欢则可以组成一对跳舞，否则每个人只可以和不超过 K 个不喜欢的人跳舞。同一对男女生不会再次跳舞，求有多少种方案使得所有男女生均能跳舞。</td></tr>
<tr><td>算法讨论</td><td colspan="2">二分枚举答案，网络流判定。先利用二分枚举答案 Ans，然后构图网络流。将男生女生各拆成两个点，B 代表 "喜欢" 男生，B' "不喜欢" 男生，G 代表 "喜欢" 女生，G'代表 "不喜欢" 男生，增加源点 S，汇点 T。1）S 向 B 连一条容量为 Ans 的弧；2）B 向 B'，G 向 G'连一条容量为 K 的弧；3）B 向喜欢</td></tr>
</table>

	的女生 G 连一条容量为 1 的弧，B'向不喜欢的女生 G'连一条容量为 1 的弧；4）G'向 T 连一条容量为 Ans 的弧。若此图的最大流满载，即 Ans*N，则代表 Ans 可行，否则不可行。	
其它	时间复杂度为 O（N^2*M*LogN）。	
重庆省选 2009	循环赛 （Un AC）	N 只队伍打比赛，赢了得 3 分，平了得 1 分，输了不得分。已知每个队伍最终得分，求有多少种方案。
算法讨论	搜索。枚举每只队伍和另一只队伍的对阵结果，由于是对阵的，所以只用枚举一半。利用可行性剪枝和极端法减枝优化，再利用提前判断解，可以很快的过掉前 22 组数据。由于此题未 AC，就不再多说了。	
其它	听说此题的正确解法是最小表示来压缩状态，不过有点难想，在考试规定时间内，用尽量少的写一个拿高分的算法，性价比是很高的。	

题目来源	题目名称	题目大意
CEOI 2005	Depot Rearrangement	有 N*M 个数字，共有 M 个不同的数字，在 N*M 个数字后面有一个空位。将这 N*M 个数字从 1 到 N*M 顺序分成 N 段，每段数字中任意两个数字均不同。求将原始序列变成这样的序列最少需要交换几次，每次交换只能和空格进行交换，输出其中一种方案。例如，2*2 个数字，1122，则需要交换 3 次。2 号位置和 5 号位置交换，4 号位置和 2 号位置交换，5 号位置和 4 号位置交换，则变成了 1212。
算法讨论	我们将 N 段看做 N 个点，M 个数字看做 M 个点，构建一个二分图。 构图：设某段 X，某个数字 Y，X 中含有 T 个 Y。若 T>1 的，则 X 向 Y 连 T-1 条边。若 T=0，则 Y 向 X 连一条边。 显然，最后我们要使得对于所有（X,Y,T），T=1。 通过观察可以发现，二分图中每一个环对应了一个有效的互相交换。而 2*Tot 条边构成的环，总共需要交换 Tot+1 次。 所以，对于二分图中每一个有 E 条边的连通分量，总共需要交换 E/2+C 次，C 为环的个数。由于，图的特殊性质，每个点的入度=出度，所以每个连通分量必然可以画出一条欧拉回路。 所以，若要使得 Ans 最小，则回路必然是欧拉回路，即即 E/2+1。 然后根据得出的欧拉回路构造解，具体的自己可以画图思考思考。	
其它	时间复杂度为 O（M）。	
CEOI 2005	Mobile Service	有 N 个工作地点，M 个工作，有 3 个工人，Cost[I,J]代表从 I 地到 J 地的花费，每个工作地点同时最多可容纳一个工人，问完成所有工作，最小花费是多少。

算法讨论	定义 F[I,A,B,C]代表，完成第 I 个工作后，3 个工人分别在 A,B,C 位置。 显然，此方程很容易的可以求出最优解，但是，显然会 MLE，我们需要利用题目的条件进行优化。 经过研究，我们可以发现，在完成上次工作后，3 个工人中有一个工人的地点是确定的，所以我们重新定义状态。 F[I,A,B]代表，完成前 I 个工作，三个人分别在 A,B,P[I]位置。 这样，可以大大的优化内存，之后记录路径输出即可。	
其它	时间复杂度为 O（N^2*M）。	
CEOI 2005	Multi-key Sorting	X
算法讨论	通过研究我们可以发现，每次排序只和最后一个出现的有关，所以当重复出现某个数字时，我们只记录最后一个即可。	
其它	时间复杂度为 O（N）。	
CEOI 2005	Critical Network Lines	N 个点 M 条边的无向图，某些点可能具有 A 属性，某些点可能具有 B 属性。当删除某条边后，使得某个点无法和 A 属性点或 B 属性点连通，则称该边为有效边。问总共有多少条有效边，并输出。
算法讨论	显然，有效边必然是割边，所以我们先将图中的双连通分量缩成一个点，剩下的边，就是割边。 对剩下的点，建立一颗树，定义 Fa[I]代表以 I 为根的子树中有多少个 A 属性点，同理定义 Fb[I]。 这样，对于树中每条边（X,Y），（Fa[Y]=0）或（Fb[Y]=0）或（Fa[Root]-Fa[Y]=0）或（Fb[Root]-Fb[Y]=0），则该边为有效边，把该边和未缩点前相同类型的边输出即可。 此方法相当的麻烦，写了将近 300 行的代码。。。其实有更简单的方法，见： http://adn.cn/blog/article.asp?id=8	
其它	时间复杂度为 O（NLogN）。	
CEOI 2005	Ticket Office	有 N 个人 M 个座位，每个点给定一个指定的座位 A[I]，代表其想购买从 A[I]开始长度为 Len 的座位票。如果能完全满足其要求，则收入全价票 2 元。若不能从指定的 A[I]开始，则收入半价票 1 元。求最大获利。
算法讨论	通过研究，我们可以发现，放弃一张全价票，至多换来两张半价票。 所以，我们有了一个贪心策略，在使得全价票尽量的多的前提下，半价票最多。 定义 F[I]代表前 I 个座位的最大获利，C[I]代表 I 座位是否有人预定，有则为 2，否则为 1，有如下方程： F[I]:=Max{F[I-1],F[I-Len]+C[I-Len+1]} 这样，记录路径，可以确定全价票的收入。 剩下的位置，从头寻找空挡插入半价票即可。	

其它	时间复杂度为 O（N）。

题目来源	**题目名称**	**题目大意**
CEOI 2006	[Link]()	有 N 个节点 N 条边，每个点的出度均为 1。要求添加最少的边，使得任何节点从 1 号节点开始访问，至多经过 K 条边就可以到达。
算法讨论	根据这个图的性质，我们可以总结出下面几点： 1）图中每个连通分量，若 1 无法到达，则 1 必然会向其连一条边。 2）删掉 1 节点，把图中每条边反向，则必然可以得到由一个环作为根的一颗树。 根据上面两点性质，我们可以先求解出不考虑环时最少需要添加几条边。 这个应该很简单，利用拓扑排序作为阶段，从 1 号节点开始，正向动态规划。 定义 F[I]代表 I 节点从 1 号节点最少需要经过多少条边可以到达，即 F[I]:=Min{F[J]}+1，若 F[I]>M，则添加一条边，F[I]:=1。 下面，我们开始考虑环。 有这样一个算法，在每个环里任找一个未被标记的点，显然它必然被标记，所以我们枚举它是被谁标记的，这样至多枚举 K 次，每次经过 N 个点，所以复杂度为 O（NK）。 面对庞大的数据范围，这样的复杂度是不能承受的，我们需要进行优化。 通过研究可以发现，每次枚举时，我们都要经过 N 个点，好多点是重复经过的，这样就产生了冗余，导致算法低效。 先进行预处理，定义 Next[I]代表若标记 I，它至多能向前推进到哪个节点，满足如下性质： 1）若 Dis（I,J）2）若 J 节点已经被标记，则可以推进。 显然，Next[I+1]必然比 Next[I]推进的节点编号大，这样预处理的复杂度就是 O（N）。 预处理完后，依然用上面的枚举，但是这次不是一个一个向后走，而是根据 Next 直接跳过那些冗余的点，这样至多经过 N/K 个点，从而复杂度降低为 O（N）。	
其它	时间复杂度为 O（N）。	
CEOI 2006	[Walk]()	一个人在城市里从(0,0)出发前往(x,y)，城市中有不可穿越的矩形建筑物（建筑物周围至少一圈是空地），求最短路程以及所走的路径。
算法讨论	根据这个图的性质，我们可以总结出下面几点： 1）图中每个连通分量，若 1 无法到达，则 1 必然会向其连一条边。 2）删掉 1 节点，把图中每条边反向，则必然可以得到由一个环作为根的一颗树。 根据上面两点性质，我们可以先求解出不考虑环时最少需要添加几条边。 这个应该很简单，利用拓扑排序作为阶段，从 1 号节点开始，正向动态规划。 以下摘自：http://lbwdruid.blog.hexun.com/21165066_d.html 对于每个矩形，我们将其右下角定为(x1,y1)，左上角定为(x2,y2)。那么每个矩形会带来两个决策点（即对最终路径有影响的点）(x1+1,y1-1)以及	

	(x1+1,y2+1)。之所以选择矩形右侧的两点是因为这样处理最终路径的时候会方便，不用判断终点附近的地形情况。对于每个决策点(x,y)，我们定义 dist(x,y)为从原点到其最短路径长。如果矩形 a[t]是在水平方向上“挡”住(x,y)离(x,y)最近的点，那么 dist(x,y)=min{dist(a[t]. x1+1,a[t].y1-1)+abs(a[t].y1-1-y), dist(a[t].x1+1,a[t].y2+1)+abs(a[t].y2+1-y)} +abs(a[t]. x1+1-x)。所以如果我们知道这个矩形是哪一个，我们就可以在 O(N)的时间内做完动态规划。 现在问题转化为如何求这个矩形。我们利用平衡树来解决。对于每个矩形，我们规定它下面的边为“插入边”，上边的边为“删除边”。考虑一纵坐标 y0。若此时有插入边，则将该矩形插入平衡树。如果有决策点(x0,y0)，则在平衡树中找到小于 x0 的最大元素，它对应的矩形就是所求。如果有删除边，则将该矩形从平衡树中删除。这样做我们每次操作的代价是 O(log N)，所以这个步骤总的复杂度是 O(N log N)，程序的时间复杂度是(N log N+N)。
其它	时间复杂度为 O（NLogN+N）。

题目来源	**题目名称**	**题目大意**
POI 2005	Toy cars	N（N<=100000）辆玩具车，地上至多可以放 K 辆，给出 Jasio 要玩玩具车的序列 P（P<=500000）。如果车在地上则 Jasio 自己拿，否则需要妈妈去高处拿，拿下来后若地面已有 K 辆玩具车，则必须再拿一辆放到高处去。问，最少要妈妈拿多少次玩具车。
算法讨论	通过对题目的研究，我们可以发现，此题的关键在于妈妈从高处拿下玩具后，要把谁放回高处去？ 此时，我们可以进行一个贪心选择。 设 Next[I]代表序列 P 中第 I 件玩具再次从序列 P 中出现时，最早的位置。若不会再次无限，则设置为较大的常量 Max。 例如：序列 P（1，2，3，1，1，2），Next[1]=4，Next[2]=6，Next[3]=Max，Next[4]=5，Next[5]=Max，Next[6]=Max。 下面进行贪心，在地上存在的 K 件玩具中，我们把 Next 值最大的拿回高处显然比 Next 值小的拿回高处更优。 所以，我们用堆来维护地上存在的 K 件玩具中 Next 值最大的玩具。 当需要玩序列 P 中第 I 件玩具时，有如下流程： 1）若 I 在堆中，则更新堆中对应的元素。 2）若 I 不在堆中，且地面上不足 K 个玩具，则把 I 加入堆，累加 Ans。 地面上已满 K 格玩具，则把堆顶元素删掉，把 I 加入堆，累加 Ans。	
其它	时间复杂度为 O（PLogK）。	
POI 2005	Bank notes	N（N<=100000）辆玩具车，地上至多可以放 K 辆，给出 Jasio 要玩玩具车的序列 P（P<=500000）。如果车在地上则 Jasio 自己拿，否则需要妈妈去高处拿，拿下来后若地面已有 K 辆玩具车，则必须再拿一辆放到高处去。问，最少要妈妈拿多少次玩具车。

算法讨论	考试时写的方法很烂，用 DFS 来写的、、、不过竟然能得 80 分，Orz。 此题的正确算法就是一个分组背包，详见 DD 得背包九讲。 此题若用普通的分组背包，则需要 O（NBK）的复杂度，显然不行。 所以，我们对其进行优化。 设面值为 AI 的货币有 BI 张，则我们把 BI 可以分解成一些 2 进制数列。若最后无法分解，则直接把剩余值放到数列中 例如：8=>（1，2，4，1） 100=>（1，2，4，8，16，32，37） 15=>（1，2，4，8） 这样，我们可以发现，不管真正用到 AI 货币是多少张，总可以用这个 2 进制数列拼出来。 所以，我们分组背包时，选择量就是这个 2 进制数列的每一个数。	
其它	时间复杂度为 O（NK*LogB）。	
POI 2005	The Bus	有一个 N*M 的（N,M<=10^9）矩阵，给出 K（K<=10^5）个信息，每个信息给出第（I,J）格的权值 D。一个人初始时在（1,1），它仅可以向下走或向右走，问走到（N,M）可以获得的最大权值是多少。
算法讨论	如果此题从 N*M 来考虑的话，必然会 TLE，因为考虑的太多的无用信息。 所以我们转而从 K 来做考虑。 若（I',J'）能走到（I,J），则必然满足 I'<=I，且 J'<=J。 所以，我们可以设计出这样一个算法。 先对 K 个信息（I,J,D），以 I 为第一关键字，J 为第二关键字升序排列。 之后，我们只用考虑 J 序列。对 J 序列进行离散化，建立一颗线段树。 在动态规划时，我们只用取出[1..J]这个区间内的最大值来更新当前点。然后再把当前点 J 插入线段树中进行更新。	
其它	时间复杂度为 O（KLogK）。	

谢谢您的支持！

由于编者水平有限，时间匆忙，书中难免有疏漏和错误之处，恳请读者批评指正。

QQ：155389632
邮箱：Cai0715@126.com