《算法艺术与信息学竞赛》

刘汝佳 黄亮 著

本页最新更新日期：2004-3-9

欢迎大家来到《算法艺术与信息学竞赛》习题页面！

更新记录

2004-3-9

本页建立

部分习题提示

1.1 节习题

1.1.1

对；不对

1.1.2

提示：考虑规模增长时的时间开销，可以对比运行时间，也可以在程序中统计基本操作数目

1.2 节习题

1.2.1

不一定。这要看枚举量和枚举时间

1.2.3

枚举即可。需要注意的是要保证每个问题只有一个正确答案而不是有个答案。本题有唯一解 cdebeedcba。

1.2.5

注意行有很多但是列不太多。硬币翻两次等于不翻，所以所有行/列最多翻一次。枚举每列是否翻有 2^9=512 种情况，此时可以用 O(n)的时间计算每行是否翻（注意：列确定以后每行独立）。

1.2.6

只需要枚举横坐标相邻的点

1.2.7

设 S1 的长度为 n。注意用串匹配来做的话时间复杂度至少为 O(10^n)，不是有效算法。应该枚举 S1 的“分段方式”。例如 231241，如果分段方式为 23|124|1，则 124 为完整的一段，前面必为 123，后面必为 125，经检验这是可行的。因此如果有一个完整段的情况可以用 O(n^2)次枚举来判定。剩下只需要解决没有完整段的情况，即被分成两段。如 2312，如果被分为 2|312，则需要枚举位数。如果是 3 位，有方程??2 + 1 = 312，无解；如果是 4 位，有方程???2 + 1 = 312?，有解 3122 + 1 = 3123。基本思想如此，但是需要注意有进位的情况。建议读者自己写程序并提交到 ural 在线题库检查自己的程序是否考虑周全。

1.2.12

首先可以证明：覆盖整个草坪当且仅当覆盖草坪的上边界。这样每个圆的作用范围是一条线段，问题转化为用最少的线段覆盖整个区间。先预处理再贪心，具体方法留给读者思考。

1.2.14

本题较难，需要先推出 n=4 时的两种可能最优策略（用代数方法容易推导出），然后用递归的思想把 n>4 的情况转化为 n<=4 的情况加以解决。提示：考虑四人速度为 1,3,4,5 和 1,2,5,6 的情况，最优时间分别为 16 和 13。

1.2.17

本题答案不唯一。例如：解方程组。

1.2.23

枚举去掉的数字位置后，几乎就可以直接解方程了（需要分类讨论和少量附加枚举）。具体方式留给读者思考

1.2.24

把袋子编号为 0,1,2...n-1，如果没有 1717 克的限制，就是第 0,1,2...n-1 个袋子依次取豌豆 0,1,2...n-1 颗，把称得的重量和 0+1+2+...+(n-1)比较，重了几克就是第几个袋子是魔法豌豆！可是现在有总重量限制，因此只有当 0+1+2+...+n-1<=1717 时才能一次称出。记 M(1)为能保证一次称出的最大 n 值，则解不等式得 n<=59。二次称可以用以下策略：59 个袋子为一组，第 0 组不取，第 1 组每个取 1 颗，第 2 组每个取 2 颗...只要总重量<=1717，则多了几克就是第几组的。由于每组只有 59 个，所以再称一次就可以了。解不等式可以求出保证二次称出的最大 n，记为 M(2)。这样递推出 M(10)发现 M(10)>10000，因此用我们刚才的策略就可以在 10 次内称出了。 **1.2.27**

如果某张纸上只有一个程序，则可以在其他顺序恢复后再单独把它插入；如果某张纸上有至少三个程序，则除了头尾之外的中间程序都只会出现一次，也可以最后处理，因此只需要保留恰好有两个程序的纸张。剩下的工作就不难了，留给读者思考。

1.3 节习题

1.3.6

自上而下的读取各行，可以用本节介绍的 floodfill 方法来作，但是更节省空间的方法是 1.4 节介绍的并查集。需要注意的是要正确的处理新块开始、旧块结束、不同块合并、相同块再次合并（形成“洞”）等几种情况。

1.3.7

下确界可以简单的通过求轮廓线的并来实现。交就没有这么简单了，因为在求轮廓线交以后可能形成非矩形的区域，即出现“凹角”。先作一次 floodfill 后删除有三侧属于同一个区域的角，直到不存在这样的角。

1.3.24

设电路对应的函数是 f(i)，其中 i 是一个 n 进制数，n 为输入个数。如果 f(000..0)=f(111..1)，则所有 x 设置为 0 即可。否则考虑序列：000..000,000..001,000..011,000...111, ... 011..1, 111..1，每相邻两项只相差一个字符。显然一定存在相邻两项的 f 值不同，不同的字符保留为 x，其他设置为相同值即可。可以二分查找，总时间复杂度为 O(mlogn)，m 为门的数目。

1.4 节习题

1.4.1

先作标记，等到浪费空间大于一半时重构树。可以证明均摊时间复杂度不变。另外还有保持每个操作最坏情况时间复杂度不变的算法，非常巧妙。

1.4.7

设 s[0]=0，s[i]=a[1]+a[2]+...+a[i]，则信息 i j even 等价于 a[i]+...+a[j]为偶数，即 s[j]-s[i-1]为偶数，即 s[j]与 s[i-1]同奇偶。这样，每条信息都可以变为某两个 s[i]和 s[j]是否同奇偶的信息。记 same[i]为当前和 s[i]同奇偶的 s[j]集合,diff[i]为当前和 s[i]不同奇偶的 s[j]集合,则一条信息 i j even 将导致 same[j]和 same[i-1]合并，diff[j]和 diff[i-1]合并；信息 i j odd 将导致 same[j]和 diff[i-1]合并；diff[j]和 same[i-1]合并。具体细节留给读者思考。

1.4.12

和标准的 Young Tableau 查找算法很类似，稍微修改一下即可。

1.4.16

先按照 x 坐标排序，然后建立一棵静态的 BST 用于统计。需要记录附加信息。建议读者写写程序，要写得尽量简单，很少几行即可写完。

1.5 节习题

1.5.8

先作减法，把两个权变成一个。可以进一步发现如果矩形有重叠，可以把重叠部分去掉和权和保持不变。这样问题变成了即找出 k 个不重叠的矩形使得权和最大。们用区域(i,j)来表示在矩阵 W 中的“第一行第 i 个格子右边所有元素加上第二行第 j 个格子右边的所有元素”这个区域，用 d[s,i,j]来表示在这个区域中选择 s 个子矩阵，它们的元素总和的最小值。看作多阶段决策问题，则决策有五种：决策一：第一行第 i 个格子不用的情况，这种决策转移到状态 d[s,i+1,j]；决策二：第二行第 j 个格子不用的情况，这种决策转移到状态 d[s,i,j+1]；决策三：第一行从第 i 个格子放一个矩形，则大小 L 有 O(n)种选择；转移到 d[s,i+L,j]决策四：第二行从第 j 个格子放一个矩形，则大小 L 有 O(n)种选择；转移到 d[s,i,j+L]决策五：两行一起放宽度为 2 的矩形，也有 O(n)种选择。

1.5.10

如果是求利益最大的方案，显然可以定义状态 d[i]为考虑前 i 个订单并接受订单 i 的最大利润。但是第 k 的方案呢？这种状态表示是不行的。它的一个重要问题在于：问题不具备最优子结构！第 k 大方案所对应的决策的子决策不一定是第 k 大的。无奈之下，我们只好增加一维状态参量，用 d[i,j]表示考虑前 i 个订单并介绍订单 i 的第 j 大利润，而状态转移时也必须考虑所有前趋状态各自的第 1,2,3…j 大利润(想一想，为什么不考虑第 j+1,j+2…k 大利润？)，然后加以比较。需要注意的是可以利用堆来降低时间复杂度，请读者思考。

1.5.11

注意到命令序列长度不超过 50，机器人不可能走得太远。所以可以先枚举终止位置，然后单独考虑每个机器人，让每个机器人删除的指令数都最少。用 d[x,y,i]表示要让前 i 条指令被执行（或被删除）后机器人处于位置(x,y)所需要删除的最少指令数，请读者列出状态转移方程。

1.5.12

首先，我们直观的猜测：任意一副筷子中 A 和 B 一定是长度相邻的两只筷子。证明如下：对于某副筷子(A1,B1,C1)和另一副筷子(A2,B2,C2)，如果 A1<=A2<=B1<=B2，那么交换一下筷子重新组合成(A1,A2,C1)和(B1,B2,C2)质量和会更优。对于某副筷子(A,B,C)和闲置的筷子 D，如果 A<=D<=B，那么交换一下重新组合成(A,D,C)质量和也会更优。

这样，我们得到了一个线性结构，只需要从左往右或从后往左递推。在本题中，由于第 3 根筷子比另 2 根长，所以我们从长筷子往短筷子递推。在递推之前，首先将 N 只筷子从小到大排序，Li 是第 i 只筷子的长度。

用 d[i,j]表示用 i..n 的单只筷子组成 j 副筷子的最小质量值之和，则当且仅当 n-i+1>=3j 的时候状态是合法的。请读者自己写出状态转移方程。

1.5.13

这道题目有一个很讨厌的条件：“只要有任务是可以完成的，那么工人不能闲着没事做”。如果工人必须按照任务序号递增的顺序，那么本题可以用简单的动态规划解决，但是本题没有规定任务完成的顺序，如果我们用 d[i]代表 i 时刻以后还需要的最少时间，那么我们做状态转移的时候会很为难：我们似乎无法知道那么任务是已经完成了的（它们不能被重复执行），因此决策集合无法确定。即：问题有后效性！

真是这样吗？（解决后效性的通常办法是增加状态参量，即记录已经有哪些任务完成了，但是这样做状态量会大增，并不是一个好办法）我们需要避免重复选择任务，但是细心的读者一定已经发现了：根本不会

重复选择任务。原因在于一个容易被忽略的条件：t<=di-ai<2ti。当完成一个任务以后，严格的时间期限已经不可能允许工作再重复选择这个任务了。接下来的任务就十分简单了，请读者自己思考。

1.5.17

提示：本题很容易想到O(n^3)的直接动态规划，但是时间复杂度还可以降低。先连接各点和圆心，把面积最大转化为正弦函数和最大，再利用正弦函数的性质进行优化。

1.5.18

铁球下落的高度和总是一定的，所以我们关心的问题只是它的水平移动距离。我们称平台i的两头为平台端点的横坐标为x[i,0]和x[i,1]，并设端点i纵坐标为y[i]。为方便处理，我们将铁球的初始位置抽象成宽度为0的平台0。

每落到一个平台，球有两种决策：向左滚和向右滚，因此我们设从平台i的第k个边缘(k=0为左边缘，k=1为右边缘)落下后还需要移动的最少水平距离为d[i,k]。设p[i,k]为从平台i的第k个边缘落下后到达的平台编号（即状态d[i,k]描述的“当前平台”，如果落下会摔碎，则p[i,k]=-1），则两种决策是：

决策一：向左滚，指标函数为d[p[i,k],0]+|x[i,k] - x[p[i,k],0]|

决策二：向右滚，指标函数为d[p[i,k],1]+|x[i,k] - x[p[i,k],1]|

状态数为O(n)，决策数为O(1)，动态规划的时间复杂度取决于状态转移的时间复杂度，即p数组的计算复杂度。我们可以从高到低依次考察平台i下面平台j，如果x[i,k]处于区间[x[j,0], x[j,1]]内，则p[i,k]=j。最坏情况下的时间复杂度为O(n^2)，比动态规划本身还要高。事实上可以用BST或者线段树来做到O(nlogn)，请读者思考。

1.5.19

本题最大的迷惑点在于佳佳的篮子，如果你在记录当前糖果堆的同时记录篮子内的糖果，那么你就上当了。佳佳很聪明，他只记录当前糖果堆的情况，从而推知哪些糖果曾被拿到篮子里，并标记编号为i的糖果总共被拿出来Ai颗。

本着将尽可能多的糖果放入口袋的原则，佳佳将能配对的糖果统统从篮子内取出，于是令Ai =Ai mod 2，则当前篮子里糖果的数量Tot=sum{Ai}就求出了。

我们用P1,P2,P3,P4分别表示当前4堆糖果剩余的糖果数量，用数组元素a[P1,P2,P3,P4]表示状态P1,P2,P3,P4相应的Tot值。显然，根据a[P1,P2,P3,P4+1]的值，我们可以推得a[P1,P2,P3,P4]的值。如果Tot≤5，那么状态P1,P2,P3,P4是合法的，记为d[P1,P2,P3,P4] = true；否则，这是导致游戏结束的非法状态，记为d[P1,P2,P3,P4] = false。根据这个布尔型状态定义，我们不难写出状态转移方程，请读者自己完成。

1.5.21

本题也是利用匹配点的稀疏性。

1.5.22

只需求原串和逆序串的LCS

1.5.23

标准的LIS问题

1.6 节习题

1.6.1

按照拓扑顺序来搜索，即先搜最里层子天平，再次里层...可以用后文提到的极端法剪枝（估计天平两边的最大、最小可能取值）

1.6.7

仔细分析一下状态空间，你会发现它其实非常少，用简单的启发函数（例如只考虑档住的纵向车移动的最小步数和与Car0步数之和）就可以取得非常好的效果。

1.6.8

此题比较难。可以利用 IDA*搜索，忽略湖泊后用网络流的方法求出的割点数目（事实上还不完全是割点，因为有些点被屏蔽掉了）作为 h 函数，加上可行性剪枝：如果可以加石头的地方都加了石头还是能相互看见，则剪枝。h 函数和可行性剪枝的作用都非常明显，读者不妨一试。

1.6.16

对于大多数数据来说，预处理合并掉相同的块并事先保存好每个块右方和下方可能的块集合后速度将会非常快。

2.1 节习题

2.1.2

先考虑 n=2^k 的情况，构造完成后任意 n 都能处理了。

2.1.3

先展开 m^k - (m-1)^k = A(m) = a[k-1]*m^(k-1) + a[k-2]*m^(k-2) + ...，其中 A(m)为 m 的 k-1 次多项式。代入 m=1,2,3...n 后左右相加得到 n^k = a[k-1]*s(k-1,n) + a[k-2]*s(k-2,n) + ... 其中 a[k]为 A(m)的 k 次项系数。可用此法从 s(1,n), s(2,n) ... s(k-1,n)计算 s(k,n)。

2.1.6

解方程组。

2.1.9

列出方程组以后发现可以直接递推计算而不需高斯消元。

2.1.10

解方程组。

2.1.11

解方程组。注意特殊情况的处理。建议读者自己分析。

2.1.14

n<=50 时可以用递推法得到解；n>=50 时不管 m 为几一定有解。

2.2 节习题

2.2.6

提示：2^k 步以后的情形很有规律。利用这个规律可以用二分法逐步用 2^k 逼近 n。

2.3 节习题

2.3.3

类似 15 数码问题的处理方法，参见书 p190 页

2.4 节习题

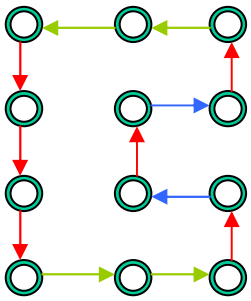
缺少的提示：

2.4.1; 2.4.2; 2.4.7; 2.4.9

2.4.3 Gridland

其实这道题目只需要考虑两种情况。

情况一：两个边长至少有一个是偶数



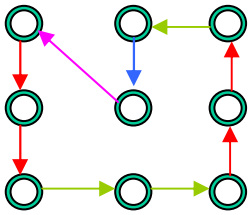
长度为：

2*(height-1)+2*(width-1)+(height-2)*(width-2)

= 2*height-2+2*width-2+height*width-2*height-2*width+4

= height*width

情况二：两个边长都是奇数



长度为：Height*width −1 + sqrt(2)

因为少一条直边但是多一个斜边。

2.4.4 邮递员

首先让我们把报酬的问题理清楚。如果期望值为 Wi 的村子是邮递员第 Ki 个经过的不同村子，那么他会得到报酬 Wi-Ki 元；同时在邮递员走过所有 M 条路后，他会从邮局那里得到 m 元，所以无论邮递员怎么走，只要他完成了任务，报酬都是 $\sum_{i=1}^{n} Wi + \sum_{i=1}^{n} i + m$。这样报酬的问题就不用考虑了，需要解决的只剩如何完成任务。

因为每个村子只可能处于两条路的十子路口上，或者四条路的米子路口上，或者一条路的中央，所以与每个村子相连的路的条数为偶数。这个村落的构造符合一笔画的充要条件，剩下的问题唯有求欧拉回路。

2.4.5 城市观光

首先，完成所有旅程后的兴趣值为一定的。如果这个兴趣值小于 0 则必然不能满足要求，也就是说总的兴趣值大于 0 是路径存在的必要条件。那么是否有满足题意的充分条件呢？实际上，总的兴趣值大于 0 就是路径存在的充分必要条件，构造如下：

1. 由于每个点的度都为 4，所以图中必然存在欧拉回路，找出一条这样的欧拉回路。

2. 从任意一点出发，找到权最小的一个位置。如果这个位置的权是正的，则这条路经已经满足要求。
3. 因为这个权是最小的负权，设为-a(a>0)。所以从这个点出发，回到起点的权至少为 a。而由于是最小负权，所以在从起点到这个点的这段路径中所有点的权都大于-a。换句话说，如果将这个权最小的点作为新起点，则从原起点到这个点的路径上的所有点的权均大于 0。同时，由于这个点的权最小，所以在从这个点到起点的路径中所有的点的权也都大于 0。满足题目要求。

所以，只要构造出一条回路，就总能在这条回路上找到一个点作为起点，满足题目要求。

2.4.6 赌博机

我们来分析一下机器失败的苛刻条件。首先抽象模型：赌博机是图（称为图 T_1）中的顶点，如果赌博机 x 的集合中有一个数 y，那么向图中添加一条从 x 指向 y 的有向边。仅当此图构成一个顶点 1 出度比入度大一，顶点 n 入度比出度大一，其余顶点入度等于出度的欧拉路时，机器才有可能失败。所以下面我们只针对此种情况进行讨论：

如果机器没有失败，那么图中剩余边构成残留图 T_3，机器运作路径构成图 T_2，T_2 是一个顶点 1 出度比入度大一，顶点 n 入度比出度大一，其余顶点入度等于出度的欧拉路。因为 T_1-T_2=T_3，所以 T_3 是顶点 n 入度出度皆为 0 其余所有顶点入度等于出度的欧拉回路。既然是回路就必然包含圈，事实上，只要残留图 T_3 是个圈而非没有边的空图，机器就不会失败。我们从每个顶点出发，找寻不包含顶点 n 的圈。如果找不到，那么机器失败；如果找到了，那么把这个圈从图 T_1 中删除，剩下的图便是使得机器获胜的路径，而这个路径仍然是个顶点 1 出度比入度大一，顶点 n 入度比出度大一，其余顶点入度等于出度的欧拉路。

2.4.8 远程通信

例子给出的方案是怎样找到的呢？我们可以这样来分析：对于 Bornholm 端口 2、Gotland 端口 1、 Gotland 端口 4，由于没有其它端口以它们为目的端口，所以它们必须被设置成发送模式，因此一一对应的 Gotland 端口 5、Bornholm 端口 4、Bornholm 端口 1 必须为接收模式。删除由 Gotland 端口 5、Bornholm 端口 4、Bornholm 端口 1 发出的有向边。结果发现，由于边的删除，Gotland 端口 3、Bornholm 端口 3 也成为只有发送模式没有接受模式的端口，所以它们必须被设置成发送模式，相应的 Gotland 端口 2 为接受模式。

参照样例的解题方法，对于任意待匹配的通讯图，我们不断找出只有发送（接受）模式的端口 x，确定它们以及相应接受（发送）端口 y 的运行方式，将点 x、y 及相关边从图中删除。如此迭代，直至无点、边可删。考虑当前剩余通讯图，图中共有 a+b 个点，a+b 条有向边，每个点入度出度皆为 1。**这样的图只能由若干个有向环构成。**倘若某个有向环顶点个数为奇数，那么此通讯图无合法匹配方案；否则对于每个环任意确定某个端口为发送模式，其余端口的运行模式也就相应确定了。

2.4.10 龙穴迷宫

图中部分点是重复的，于是想到不断把重复的点合并，以求出最少的点数。

按 F_1 的定义进一步定义 F_k：描述以点 k 为起点到点 n 的路径，其中 1≤k≤n，有：

$$F_k(P,C)_{1\le k\le n-1}=\begin{cases}(1,color[k])+F_{ek,1}(P,C)\\(2,color[k])+F_{ek,2}(P,C)\\(3,color[k])+F_{ek,3}(P,C)\end{cases},\quad F_n=('',color[n])$$

显然，若 $F_{ek,i}=F_{ek,j}$，且 $e_{k,i}\neq e_{k,j}$，则可以将点 $e_{k,i}$ 与点 $e_{k,j}$ 合并，这样既减少了顶

点数目又不会影响 F_k，更不会改变 F_1。进而推广到更一般的情况，只要 $F_i=F_j$，即可合并点 i 与点 j。由于 F_n 至 F_1 由简变繁，不难想到，从 F_n 起一步一步向上推导，逐步合并点。

有向无环图层次分明，定义 deep[i]：表示点 i 到点 n 的最大步数，显然有：

$$deep[n]=0$$

$$deep[i]=\max\{deep[e_{i,1}],deep[e_{i,2}],deep[e_{i,3}]\} \quad (1\le i\le n-1)$$

若 $deep[i]\neq deep[j]$，则 F_i 与 F_j 中序列的最大长度不等，一定有 $F_i\neq F_j$，。可见，点 i 和点 j 若可以和并，i、j 一定在同一层中。

于是从 $deep=0$ 的点 n 起，逐层合并点。当 $deep[i]=deep[j]$，因为该层以下的点已合并过，只需符合条件 $color[i]=color[j]$，$e_{i,r}=e_{j,r}(1\le r\le 3)$（i、j 三边指向的点相同），即　可判断出 $F_i=F_j$，合并点 i 和点 j。

算出合并后图的点数，即为所求。

需要注意的是，合并点时，需改变相关的 e 值。

2.4.11 追捕游戏

我们把棋盘上的每一个格子作为一个点，如果两个格子相邻，就在相应的两个点之间连上一条边。由题意知，图中不存在一个由 3 个点组成的环。根据题意，如果 B 抓不到 A，那么这个游戏就可以一直进行下去。所以，我们的首要任务是判断什么情况下 B 抓不到 A。

凭直觉我们可以想到：如果 A 能够成功到达一个环上的某一点，那么 A 就不会被 B 抓住了（因为 A 可以顺着这个环和 B 兜圈子）。但是，这样的解释是不充分的。下面我们就给出一个严格的证明。

证：我们先作一个定义：一个点如果出现在某个环上，那么我们就把它称为安全点。

根据环的含义，***每个安全点至少和两个安全点相连***，否则它就不可能出现在一个环上。由于题目中规定，图中不存在一个由 3 个点组成的环，所以***与一个点相邻的所有点相互之间不相邻***。所以，如果玩家 A 位于一个安全点 P，玩家 B 走到了与 P 相邻的点 Q 上，则必定存在一个与安全点 P 相邻的安全点 R，满足 R 与 Q 不相同。而且，R 与 Q 一定不相邻。如果 B 所在的点与 A 不相邻，那么 A 就可以留在原地不动。所以，A 总是可以位于一个与 B 不相邻的安全点上，这样 B 就永远抓不到 A。

在 A 没有能到达安全点之前，它所能到的点都不是安全点，所以不存在过这些点的环。这些点必定构成一棵树。如果以 A 的出发点作为根 R，那么要从根 R 的一棵子树上的某一点到达另一棵子树上的某一点，必须要经过根 R（在完整的图中也是如此）。否则我们就可以得到一个过根 R 的环，与根 R 不是安全点矛盾。同理，对于根 R 的一个儿子 Q，Q 的一棵子树上的某一点到达另一棵子树上的某一点，必须要经过 Q（在完整的图中也是如此）。……因此，B 一定会在树中的 X 点上第一次出现，这个 X 是一定的。（当 B 的出发点在树上时，这个 X 就是他的出发点）。

在 A 没有到达安全点之前，他到树上的每一个点都只有一条路。他应当事先定好一个目标，朝着一个希望点走（我们把与安全点相邻的树上的点称为希望点）。如果要改变目标，他就只能走回头路，这样显然是不合算的。所以 A 无法根据 B 的走法来及时调整自己的策略。而 B 的目标也是相当明确的，就是朝着根的方向走，尽快地卡住要道。因为，随着树的深度的增加，分叉会越来越多，其中有希望点的几率也相应增大。

对于树上的某一个点 P，如果 X 是 P 的一棵子树上的点、在 P 的另一棵子树上有希望点，那么若 A 能够比 B 早一轮到达 P，A 就一定能安全到达安全点。反之，如果对于所有

这样的 P，A 都不能比 B 早到，那么 A 的所有通向安全点的道路都会被 B 堵死，A 迟早会被 B 抓到。

这里 R 和 X 都在树上，所以 R 和 X 到树上的点的路线都是唯一的。所以，我们可以用宽度优先搜索来判断究竟是 A 能率先到达安全点还是 B 能抢先一步卡住咽喉要道。

我们首先要找出所有的安全点。我们从点 1 开始做宽度优先搜索。每出现一条横向边，就表示找到了一个环。找出这个环上的所有点，然后把它们都记为安全点。

接着，我们用宽度优先搜索分别计算每一回合后 A、B 两人可以到达的格子。我们先搜 B 的，再搜 A 的（这和题中两人走棋的顺序相反）。对于一个点，如果 B 已经到达过了，那么 A 就不能再去了，否则就一定会被抓住。如果 A 能安全到达一个安全点，那么 B 就再也抓不到 A 了，程序就应该马上终止。当 B 能到达所有 A 能到达的格子后，那么 A 就无路可走了。这时所用的步数就是 B 抓到 A 所需要的步数。

我们要开一个布尔型数组，记录每个点是否为安全点。在宽度优先搜索中还需要开几个数组。

在解题的过程中，我们只用到了宽度优先搜索，所以算法的时间复杂度是 O(n+m)，其中 n 是格子的个数，m 是相邻的格子的对数。由于 m≤15000 且 n≤3000，所以这个算法是非常理想的。程序的空间需求也不大，大约为 300k。

2.5.1 最公平路

从小到大枚举最小边，则最大边不减。每次用 O(m)的时间判连通即可。

2.5.3 最小圈

初始时设置 d[i,i]=无穷大，再套用 floyd-warshall 算法即可。

2.5.6 路的最小公倍数

单独考虑每个素数 a，那么一条路 p 的 s(p)值中 a 的指数就是 p 上的权中 a 的最小指数，即路的“瓶颈”。所有 s(p)的最小公倍数就是所有路的瓶颈最大值。把“求和”操作改成“取最小值”操作，然后套用 dijkstra 即可。

2.5.7 货币兑换

这道题目的本质是判断加权有向图是否有正圈，套用 bellman-ford 算法即可。如果迭代 n-1 次以后标号仍然会改变，则有正圈。为了找到正圈，需要记录每个标号变化最后一次是由那个结点引起的。

2.5.8 速度限制

如果每条道路都有限速标志，那么此问题只是最普通的最短路径。基于此，我们尝试给每条道路标上限速标志。

无标志路径的速度由上一条道路的速度决定；如果上一条道路亦无标志，那么它们的速度又由上两条道路的速度决定；如果上两条道路亦无标志，那么它们的速度又由上三条道路的速度决定……如此向前追溯，直至遇到有标志路径或者起点。所以逆向思维，我们从每一条有标志路径或者起点出发，在它们的后面接上所有单独的或者一串连续的无标志路径，使之成为一条有限速标志的新路。这样，此问题就转化为普通最短路径。当然，如何求出所有

单独的或者一串连续的无标志路径可以用 $O(n^3)$的 Floyd 算法。

2.5.9 会议呼叫

首先，激活的边是不可能产生环的，所以形成了一棵树，而且只有一个分叉点。先从 A，B，C 出发调用 SSSP，这一步是 O(m+nlogn)的。然后枚举这个分叉点 x，让 d[A,x]+d[B,x]+d[C,x]尽量大。这一步是 O(n)的。

2.5.11 开发计划

这道题目是“航天计划问题”的直接推广。

2.5.12 因特网宽带

这是个普通的最大流的模型。不同于一般书上此类算法之处在于，本题是无向图的关系，所以每条弧（两点之间的路径）是无向的，因此在具体计算过程中，每次找最短路进行增流，弧的实际方向是动态变化的，这就需要进行判断。判断的方法也很简单：设从第 I 个点对第 J 个点进行分析的，记流量为 F，容量为 C，那么如果 F[i,j]<C[i,j]并且 F[j,i] = 0，即不存在反向的流量，则可以正向扩展 i->j；如果 F[j,i] > 0 ，那么可以进行反向扩展。

2.5.13 出题者的烦恼

X 结点是题目，Y 结点是类别，如果题目 a 属于类别 b 则连边 X_a->Y_b，容量为 1。对于每个 X_a，连边 s->X_a，容量为 1。对于每个 Y_b，连边 Y_b->t，容量为第 b 类需要的试题数目。注意本题不是求最大流而是固定流量为给定数 k 的流，因此用增广路算法比较合适，每次流量增加 1（因此 ford-fulkerson 算法是不适用的！），恰好为 k 时停止。

2.5.14 马戏团

本题是标准的最小路径覆盖问题。

2.5.15 锦标赛

X 结点是还没有进行的比赛，Y 结点是人，如果比赛 a 的双方是 p 和 q，则连边 X_a->Y_p 和 X_a->Y_q，容量都为 1。对于每个比赛 a，连边 s->X_a，容量为 1；对于每个人 a，连边 Y_a->t，容量为...直观的说，这个容量是此人能取胜的最大场数。枚举冠军，则最好情况就是让他所有比赛都取得胜利，然后枚举第二名取胜的分数，就可以计算出每个人最多能够取胜的场数 max_a，这就是刚才提到的容量。

2.5.17 方格取数

把方格黑白染色，相邻的格子连一条线。如果每个数并不是很大，那么我们把写着数 x 的格子拆成 x 个点，本来连接着数 x 和 y 的一条线就变成了拆成了 x*y 条线。这样就是二分图的独立数了。但是这样做复杂度太高，实际上这个匹配数就等于不拆点前的最大流量。这样就不和数的范围有关了。

2.5.18 团队分组

对于第 1 个人，由于一共只有两个组，所以不妨设他在第 1 组。如果他和所有人认识，那么可以不考虑他，把其他组分好以后再加进去。否则假设有一个人 x 和他不认识，显然 x 必须在第 2 组。对于所有 1 不认识的人，我们都可以把他们丢进第 2 组，同样对于第 2 组中某人不认识的，我们也可以丢进第 1 组，直到没有办法再加入其他元素，这时算法告一个段

落。剩下的重新开始，又得到新的两组，现在需要把两组进行合并，应当借助动态规划。

2.5.19 调皮的导盲犬

由于狗每次离开主人最多只能选一个地方，而且一个地方最多只能玩一次，所以我们很容易建立一个二分图模型：左边是顶点集合 A，表示狗某次离开主人；右边是顶点集合 B，表示好玩的地方；边（x,y）表示狗第 x 次离开主人可以去好玩的地方 y。问题要求的即是集合 A 与集合 B 的最大二分图匹配。

2.5.20 会议

最基本的二分图匹配。

2.5.21 神秘之山

无论用什么样的算法，预处理求解每个人分别到达每个端点的时间都是必要的。为方便叙述，不妨假设队员I在端点j的左侧，他距j水平距离X_0个单位路程；连接端点j、j-1的直线交x轴于横坐标p1，连接端点j、j+1的直线交x轴于横坐标p2，队员i向右步行到距j水平距离X个单位路程时开始爬山。很明显，范围$\mathrm{Max}\{0, x_j\text{-}p2\}\le x\le x_j\text{-}p1$。而我们的目标是求 $\frac{\sqrt{X^2+Y^2}}{C_i}+\frac{X_0-X}{W_i}$ 的最小值，可以设角度也可以用韦达定理求解，这里就不再赘言。

如果我们知道最迟到达的时间，那么可以根据该时间构造一个左队员右端点的二分图，然后用匈牙利算法判断是否可以完全匹配。

我们把所有人到达所有端点的时间Qsort从小到大排序，然后二分枚举最迟到达的时间。每次枚举都判断是否可以完全匹配，直到找到那个最早的完全匹配的到达时间。另一个想法是从小到大枚举最迟到达的时间，这样每次可以才前一次的基础上求增广路。

2.5.26 Unix 插头

如果插头 x 可以通过一个插座板转换成插头 y，那么令 a[x,y]=true，否则令 a[x,y]=false。然后我们用 $O(n^3)$的 Floyd 算法求出任意一对插头(x,y)，插头 x 是否可以直接或通过若干连续插座板转换成插头 y。

接下来我们构造一个二分图，图左边是插头，右边是插座，如果插头 x 可以直接或通过插座板间接与插座 y 相连，那么顶点 x 与顶点 y 之间存在一条边，边的容量为 1。剩下的事情唯有求最大二分图匹配。