1.1

1

阿尔·花剌子模（公元9世纪）是一位伟大的阿拉伯学者，以他的著作《代数学》而闻名。事实上，“代数”这个词是派生出来的，源于他著作的阿拉伯书名，而“算法”一词则是衍生出来的，来自于阿尔·花剌子模姓氏的翻译。

2

这个法律问题尚未解决。当前的法律将算法分为两类，一种是数学算法，但是他不能申请专利；另一种是其他算法，如果它能够通过计算机程序实施，则它可以申请专利。

3

略

4

注意:用牛顿法可以得到一个更快的算法来解决这个问题第11.4和12.4条（待定）

5

待定

6

A.

gcd(31415, 14142) = gcd(14142, 3131) = gcd(3131, 1618) = gcd(1618, 1513) = gcd(1513, 105) = gcd(1513, 105) = gcd(105, 43) = gcd(43, 19) = gcd(19, 5) = gcd(5, 4) = gcd(4, 1) = gcd(1, 0) = 1

为了回答这个问题，我们需要比较算法对给定输入进行除法的次数。欧几里得算法的除法次数是11（见a部分）。连续整数检查算法在14142次迭代交换中，每一次都要用到是1到2次除法；因此，乘起来之后，连续整数检查算法的除法总数在1\*14142和2\*14142之间。1\*11442/11≈ 1300，2\*14142/11≈ 2600。换句话说，欧几里德算法的速度将是连续整数检查算法的1300到2600倍。

7

如果 d 整除 u 和 v, 那么 d 一定能整除 u±v;

如果 d 整除 u,那么 d 也能够整除 u 的任何整数倍 ku.

对于任意一对正整数 m,n,若 d 能整除 m 和 n,那么 d 一定能整除 n 和 r=m mod n=m-qn；显然，若 d

能整除 n 和 r，也一定能整除 m=r+qn 和 n。

数对(m,n)和(n,r)具有相同的公约数的有限非空集，其中也包括了最大公约数。故 gcd(m,n)=gcd(n,r)

8

对于输入的任何一对数字m，n，如果0≤ m<n，根据欧几里德算法的要求，会把相对大的数字放在前面，小的放在后面，因此，会在第一次迭代时交换数字：gcd（m，n）=gcd（n，m）。因为如果m<n，则m mod n=m。欧几里德算法的目的就是要算大的数字除以小的数字的最大公约数。

这样的交换仅仅会发生一次，因为gcd（m，n）=gcd（n，m mod n）表示在每次算法迭代完成之后，新配对的数字n肯定大于m mod n

9

A.

对于输入的任何一对正整数m,n,如果满足m≥n≥1，并且m是n的倍数，根据欧几里得算法它最少只需要做一次除法，比如（10，5），经过一次除法，就是（5，0），也就是最大公约数为5。

B

答案是5次，这是由欧几里德算法在计算gcd（5,8）。得到这个答案并不太耗时,检查算法对所有输入进行的分割数对1<m<n≤ 10,

注：相关的一般结果是输入对m，n，其中0≤ n<N，根据欧几里德计算gcd（m，n）的算法至多为[logφ（3−φ） N）]其中φ=（1+√5)/2.(待定)

10

A

以下是非递归版本：

Euclid2（m，n）//定义一个函数Euclid2，里面两个参数m,n

//通过基于减法的欧几里德算法计算gcd（m，n）

//输入：两个非负整数m和n都不等于0

//输出：m和n的最大公约数

While n≠0 do

If m<n

swap(m, n)交换两者的位置，大的赋给m,小的赋给n

m=m-n

return m

B

不难证明可以写在黑板上的整数是由欧几里德算法的减法版本生成的。尽管它们在黑板上的出现顺序可能不同，但它们的总数始终保持不变：它等于m/gcd（m，n），其中m是刚开始输入的一对整数中相对较大的一个整数。因此，可能的移动总数为m/gcd（m，n）−2.因此，如果m/gcd（m，n）是奇数，应该选择先行动；如果是偶数，应该选择后行动。

11

略

12

由于所有的门最初都是关闭的，因此在最后一次走过之后，大门将打开，为了达到这个效果，那么开关门的次数应该控制为奇数次。

假设我们经过门j次（1≤ j≤ n），一扇门开关的次数为i(1≤i≤ n)。也就是第j次经过，对应开关门的次数为i=jk.

因此开关门的次数等于它的除数的个数。另外说一下，如果j整除i，即i=jk，i是j的倍数，那么i也是k的倍数。因此，所有i的除数可以成对表示（例如，对于i=12，这样的对是1和12，2和6、3和4），除非i是一个完全平方（例如，对于i=16，4并没有另一个不同的相匹配的除数）。这意味着如果i是由奇数个除数组成，那么它就是一个完全平方，即i=j\*j。因此，门如果想在最后一次经过的时候，将其打开，那么i应该是完全平方的状态。既然如此，j的范围不会超过n,应该是等于√n ：而i是在1到√n范围内的正整数的平方。

1.2

1

需要向对岸运四次。第一次把羊运过河；第二次把狼运过河，回来时把羊运回；第三次把菜运过河；第四次把羊再运过河。

一个农夫有一条船，和狼、羊、菜，农夫要把这3样东西运到河到另一边，农夫每次最多只能通过这个船运一样东西，要防止狼吃羊、羊吃白菜（即不能在没有农夫在的情况下同时在同一岸边）因此运送的顺序至关重要。

答案一：

农夫带羊过河。农夫返回。农夫带狼过河。农夫带羊返回。农夫带菜过河。农夫返回。农夫带羊过河。

答案二：

农夫带羊过河。农夫返回。农夫带菜过河。农夫带羊返回。农夫带狼过河。农夫返回。

农夫带羊过河。

2（1）甲乙二人先过-------------2分

（2）乙带电筒返回------------2分

（3）乙留下，丙丁二人过-----------10分

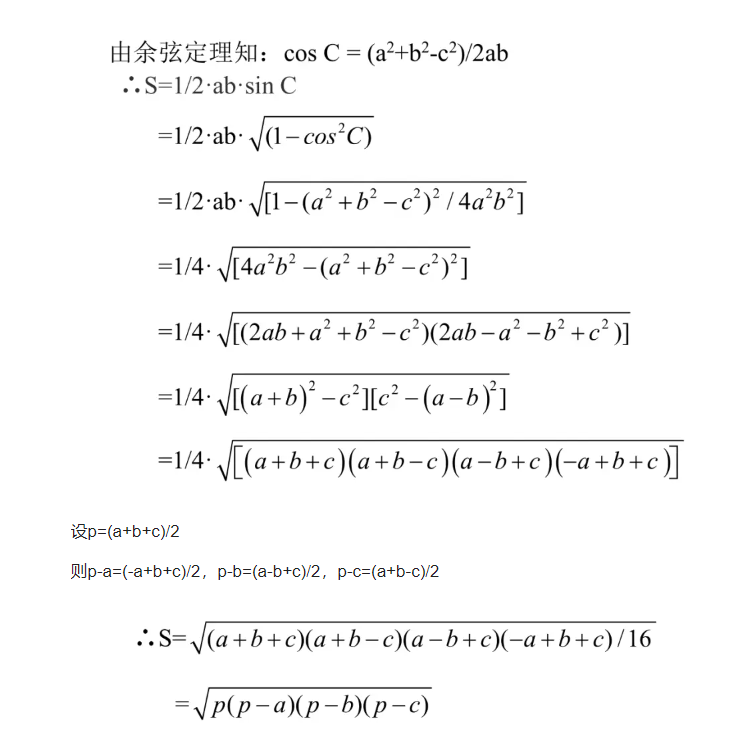
（4）甲带电筒返回--------------------1分

（5）甲乙二人过--------------------2分

合计：17分

3

A



B

适用于所有三角形

作AD垂直BC,交BC于点D

则sinB=AD/c

所以AD=AB\*sinB=csinB

所以S△ABC=1/2\*BC\*AD=1/2acsinB

同理可得：s△=1/2bcsinA=1/2acsinB=1/2absinC

c.

第三个公式没有任何问题

4.

算法 Quadratic(a,b,c)

//求方程 ax^2+bx+c=0 的实根的算法

//输入:实系数 a,b,c

//输出:实根或者无解信息

If a≠0

D←b\*b-4\*a\*c

If D>0

temp←2\*a

x1←(-b+sqrt(D))/temp

x2←(-b-sqrt(D))/temp

return x1,x2

else if D=0 return –b/(2\*a)

else return “no real roots”

else //a=0

if b≠0 return –c/b

else //a=b=0

if c=0 return “no real numbers”

else return “no real roots”

这里Quadratic为二次的意思，而//表示备注，更多的详细解说可以看章节11.4

5

a.将十进制整数转换为二进制整数的算法

输入：一个正整数 n

输出：正整数 n 相应的二进制数

第一步：用 n 除以 2，余数赋给 Ki(i=0,1,2...)，商赋给 n

第二步：如果 n=0，则到第三步，否则重复第一步

第三步：将 Ki 按照 i 从高到低的顺序输出

b.伪代码

Algorithm Binary(n)

//该算法实现了查找的标准方法

//十进制正整数的二进制展开

//输入:正十进制整数n

//输出:list bk bk−1…b1b0 (n个二进制位数)

k ← 0

while n ≠ 0

bk ← n mod 2

n ← [n/2]

k ← k + 1

6.略

7.

a.

π作为无理数，只能近似计算。

b.

略

8.

略

9.

下面的改进版本只考虑相同的元素对,并且避免在最内层循环中重新计算相同的表达式:

算法MinDistance2 (A[0..n − 1])

//Input:一个数组(0 . .N−1]个数字

//Output: 两个元素之间的最小距离d

dmin ← ∞

for i ← 0 to n − 2 do

for j ← i + 1 to n − 1 do

temp ← |A[i] − A[j]|

if temp < dmin

dmin ←temp

return dmin

一种更快的算法基于预排序的思想(参见第6.1节)。

10.

Polya的四个要点是:

1. 理解这个问题

2. 制定一个计划

3.实现这个计划

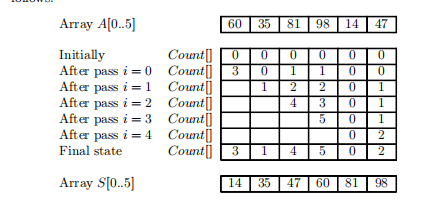
4. 回顾/检查

1.3

1.

a. Sorting 60, 35, 81, 98, 14, 47 by comparison counting will work as

follows:



b.

算法不稳定.比如对列表”2,2\*”排序

c.该算法不合适，因为它使用了两个大小为n的额外数组:Count和S。

2.

略

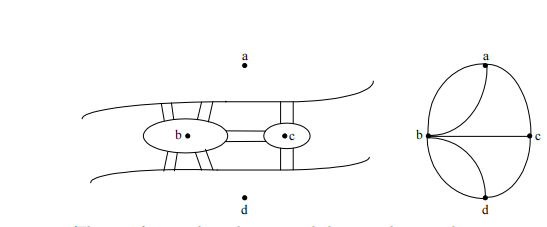
3.

使图案与文本的开头对齐。从左到右比较模式和文本的对应字符，直到匹配所有模式字符(然后停止—搜索成功)或算法耗尽文本字符(然后停止—搜索不成功)或遇到不匹配的字符对。

在后一种情况下，将模式右移一个位置并继续比较。

4.

a.

如果我们用顶点表示河岸和两个岛，用边表示桥，我们会得到下图

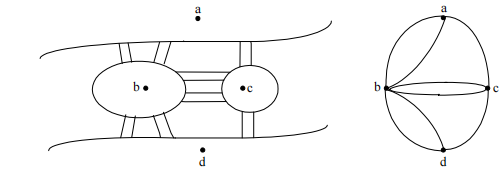
(事实上，这是一个多图，而不是一个图，因为它在同一对顶点之间有不止一条边。但这对手头的问题无关紧要。)问题是，在这个多图图中是否存在一条路径(即相邻顶点的序列)，它只遍历一次所有的边，并返回到一个起始顶点。这样的路径被称为欧拉路径;如果一条路径只穿过一次所有的边，但不返回到它的起始顶点，它被称为欧拉路径。

b.

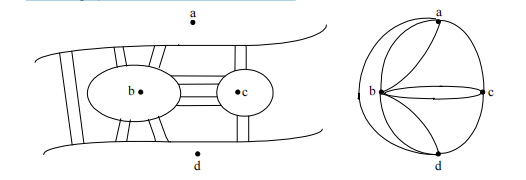
欧拉证明了一个欧拉电路存在于一个连通(多)图中，当且仅当它的所有顶点都是偶度的，其中顶点的度定义为它作为端点的边的数量。

同样，当且仅当连通(多)图恰好有两个奇度顶点时，欧拉路径存在于连通(多)图中;这样的路径必须从这两个顶点中的一个开始，到另一个顶点结束。因此，对于普氏的多重图，既不存在欧拉电路也不存在欧拉路径，因为它的四个顶点都是奇度的。

如果我们要满足欧拉路径，则多图的两个顶点必须是偶数。这可以通过增加一座连接现有桥梁相同位置的新桥来实现。

例如，两个岛屿之间的一座新桥将使步行a - b - c - a - b - d - c - b - d成为可能

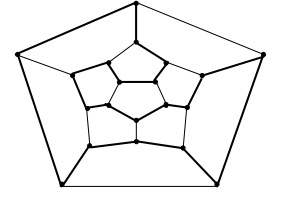
如果我们想要一个回到起点的遍历，对应多图中的所有顶点都必须是偶数。因为一个新的桥/边会改变两个顶点的奇偶性，所以至少需要两个新的桥/边。例如，这里有一个这样的“增强”:



这将使a-b-c-a-b-d-c-b-d-d-a成为可能

5

哈密顿电路标记在下图中：



6.

a.

至少有三个“合理”的标准浮现在我的脑海中:最快的旅程，火车停靠次数最少的旅程，以及需要换车次数最少的旅程。请注意，第一个标准要求车站之间的预期旅行时间和换车所需时间的信息，而其他两个标准不要求这样的信息。

b.

一种自然的方法是模拟地铁规划，用图形的顶点表示站点，如果对应的站点之间有一条铁路线，则用一条边连接两个顶点。如果要考虑换乘火车所花费的时间(例如，因为所讨论的车站在多条线上)，那么车站应该用多个顶点表示。

7.

a.

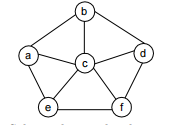
找出n个给定城市的排列，使排列中连续城市之间的距离加上最后一个和第一个城市之间的距离的总和尽可能小。

b.

将图的所有顶点划分为最小数目的不相交子集，这样就不会有来自同一子集的顶点连接边。8.

a.

创建一个图形，其顶点代表地图的区域，当且仅当对应的区域有共同的边界(因此不能被涂成相同的颜色)时，边缘连接两个顶点。以下是所给地图的图表:



解决此图的图着色问题可以得到尽可能少的颜色数的地图着色。

b.

在不丧失通用性的情况下，我们可以将颜色1和2分别分配给顶点c和a。这将强制对其余顶点进行以下颜色分配:3到b, 2到d, 3到f, 4到e。因此，此映射所需颜色的最小数量为4。

注:这是一个众所周知的事实，任何地图都可以有四种或更少的颜色。这个被称为“四色问题”的问题一个多世纪以来一直没有得到解决，直到1976年，美国数学家K.阿佩尔和W.哈肯结合数学论证和广泛的计算机应用，终于解决了这个问题。

9.

如果n = 2，答案总是“是”;我们可以假设n≥3。从给定的集合中选择三个点P1, P2, P3。写出端点为P1和P2的线段的垂线平分线l1的方程，也就是到P1和P2等距点的轨迹。写出端点为P2和P3的线段的垂线等分线l2的方程，也就是到P2和P3等距点的轨迹。通过解两个未知量x和y的方程组，求出直线l1和l2交点P的坐标(x, y)(如果方程组没有解，返回“no”:这样的周长不存在)。计算从P到每个点Pi的距离(或者更好的是距离的平方!)I = 3,4，…， n，检查它们是否都相同:如果是，返回“yes”，否则，返回“no”。

10

略

1.4

1.

a.

将第i个元素替换为最后一个元素，并将数组大小减小1。

b.

用一个特殊的符号替换第i个元素，该符号不能是数组元素的值(例如，0表示正数数组)，以标记第i个位置为空。(这种方法有时被称为“惰性删除”。)

2.

a.

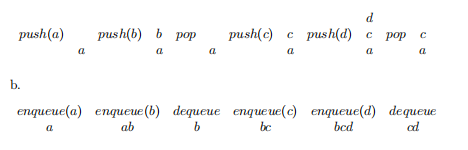
使用二分搜索(如果不熟悉这种算法，请参阅4.3节)。

b.

在排序链表中进行搜索时，一旦遇到大于或等于搜索键的元素就立即停止。

3.

a.



4.

a.

对于邻接矩阵表示:

1当且仅当一个图的邻接矩阵中除主对角线上的元素外的所有元素都等于1，即当每个1≤i, j≤n, i≠j时，A[i, j]=1，图是完备的。

2当且仅当图的邻接矩阵在其主对角线上有一个等于1的元素时，即当某1≤i≤n时，A[i, i]=1，图就有一个循环。

3一个(无向，无循环)图当且仅当它的邻接矩阵有一个全零行时，它有一个孤立的顶点。

b

对于邻接表表示:

1当且仅当图的每个链表都包含图的所有其他顶点时，图才是完整的。

2当且仅当图的邻接表中有一个包含定义该表的顶点时，图就有一个循环。

3当且仅当一个邻接表为空时，(无向，无循环)图有一个孤立的顶点。

5.

第一种算法的工作原理如下。标记一个顶点作为树的根，使其成为要构造的树的根，并用这个顶点初始化一个堆栈。重复以下操作，直到堆栈变为空:如果有一个未标记的顶点与堆栈顶部的顶点相邻，标记前一个顶点，将其附加为树中顶部顶点的子顶点，并将其推入堆栈;否则，弹出堆栈顶部的顶点。

第二种算法的工作原理如下。标记一个顶点作为树的根，使其成为要构造的树的根，并用这个顶点初始化一个队列。重复以下操作，直到队列变为空:如果队列前面的顶点附近有未标记的顶点，标记所有顶点，将它们作为子顶点附加到树的前顶点，并将它们添加到队列中;然后退出队列。

6.

因为高度被定义为从树的根到叶的最长简单路径的长度，这样的传递将包含不超过n个顶点，这是树中顶点的总数。因此，h≤n−1。

高度为h且顶点数最多的二叉树是一棵满树，它的所有h + 1层都填充了尽可能多的顶点。这种树的总顶点数为:1665238206313

因此，对于任何有n个顶点，高度为h的二叉树

1665238306131。

这意味着1665238334879

或者，在两边取二进制对数，然后取

假设h + 1是整数，h + 1≥[log2(n + 1)]。

由于[log2(n + 1)] = [ log2 n]+1(参见附录A)，我们最终得到

H + 1≥log2 n + 1或H≥log2 n。

7.

a.

插入可以通过在数组的最后一个元素之后添加新项来实现。寻找最大的元素需要对数组进行标准扫描，以找到其最大的元素。删除最大的元素A[i]可以通过与最后一个元素交换来实现，并将数组的大小减少1。

b.

我们假设数组A[0..N−1]表示优先队列升序排列。插入值为v的新项可以通过扫描排序的数组来完成，比如从左到右，直到元素a [j]≥v或到达数组的末尾。(查找插入新元素位置的更快算法是4.3节中讨论的二进制搜索。)在前一种情况下，通过移动A[n−1]，将新项插入到A[j]之前，…， A[j]向右的一个位置;在后一种情况下，只需将新项追加到数组的最后一个元素之后。要找到最大的元素，只需返回已排序数组的最后一个元素的值。删除最大元素的方法是将数组的大小减少1。

c.

新元素的插入是通过使用在二叉搜索树中插入新元素的标准算法完成的:递归地，根据新键比根键大还是小，将新键插入到左子树或右子树中。找到最大的元素需要找到二叉树中最右边的元素，方法是从根开始，沿着右子结点的链，直到到达一个没有右子树的顶点。该顶点的键将是最大的元素。删除它可以通过使其父节点的右指针指向被删除顶点的左子节点来完成。如果最右边的顶点没有左子顶点，则该指针被设为“null”。最后，如果最右边的顶点没有父顶点，也就是说，如果它恰好是树的根，它的左子顶点就成为新的根;如果没有左子结点，树就变成空的。

8.

使用位向量，即一个包含n位的数组，如果底层集合的第i个元素当前在字典中，则第i位为1，否则为0。搜索、插入和删除操作将需要检查或更改该数组的一个单位。

9.

使用:(a)优先队列;(b)一个队列;(c)堆栈(和反向波兰表示法——一种不带括号表示算术表达式的聪明方法，通常在数据结构课程中学习)。

10.

最直接的解决方案是搜索第二个单词中第一个单词的每个连续字母。如果搜索成功，删除第一个出现在第二个单词中的字母，否则停止。另一种解决方案是对每个单词的字母进行排序，然后在一个简单的并行扫描中进行比较。我们还可以生成并比较给定单词的“字母向量”:Vw[i] =字母第i个字母在单词w中的出现次数。这样的向量可以通过将其所有组成部分初始化为0，然后扫描单词并在向量中增加适当的字母数来生成。