**1002**

Please be accurate to 1 decimal place. 保留到小数点后一位。

CPP中控制输出精度 setiosflags(ios::fixed)<<setprecision(1)

采用数组按位索取，用空间换时间

**1003**

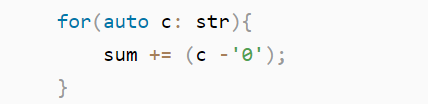
Dijkstra算法的应用，其中涉及到了多条相同的最短路径，因为只需要输出相同的数量，所以采用一个数组保存即可，如果要保存路径最好采用vector。

**1004**

DFS，深度优先思想，通过Quere来按层次搜索，重点是构造一个带有层次数和孩子结点编号的结构体。

**1005**

字符串操作，int->str : to\_string, str->int : str[i] -‘0’，str.at(i)-‘0’

遍历一个string

**1006**

时间比较时，采用换算为秒之后再比较的方式

**1007**

可用动态规划解，求和最大的连续子串，主要考虑的是和小于0是对后面的只有负效应，所以舍弃的思想。**但题目中需要的如果有最大和相同的情况，找到最小的下标**，所以和为0时应该不舍弃。

**1008**

仔细看题！！！确定好输入输出样例都代表啥。

**1009**

注意题中给的值的范围！！！系数相加之后范围最大会扩大一倍，设置数组时要记得设置大一点。同时注意for循环相关参数的设置，大于还是小于，i++还是i--

**1010 重点**

此题很好，考察进制运算。一个数中出现的最大数加1位这个数可能的最小进制数。计算一个进制数的值时采用快速幂。大范围内选择一个值时要有二分查找等算法思想，不能直接憨憨地直接遍历。

**1012 重点**

此题是一种横向纵向比较的很好例子。通过重构sort函数，虽然比较分数但sort改变的顺序是含有各自编号的数组 sort(&r[1],&r[1]+n,cmp); 从而方便后面的查找。

**1013**

连通量分析，主要可以通过DFS，BFS或并查集的方式来计算图中的连通分量。

邻接表的构建可以用vector<int> graph[MAX]来替代，就不需要用链表了。

如果遇上超时时可以将cin，cout换成scanf，printf

并查集分析。逆向，加入被check的那个点不存在，将剩下的点进行并查集处理，合并和路径压缩，最后看有几个集合，就需要n-1条路。（set的特性是不会存在重复的数据，可以用insert的方法查看一个数组中有哪些数）。并查集中找根节点一律用find，不能直接用那个数组。

**1014**

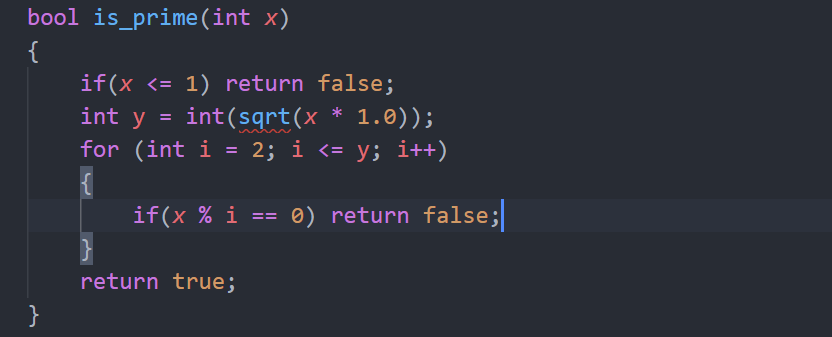
使用队列，将从开始时刻花了多少时间转换成每结果一段时间后服务时间还剩下多少。

queue.front()

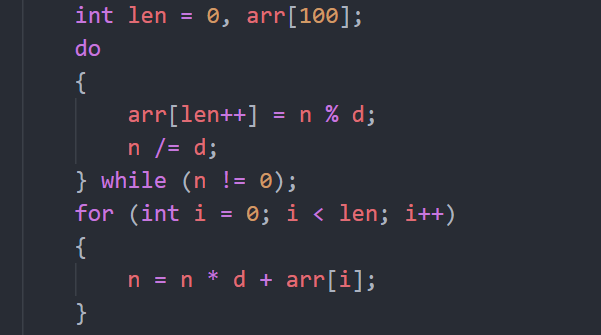
queue.pop()

**1015**

判断是否为质数



将一个数按照某种进制进行逆序处理



例如 73（10）→ 37, 23（10）→ 32， 23（2）→ 29

**1016**

当遇到和时间（年、月、日、时分秒）有关的模拟题时，可以采用结构体保存，同时输入时用scanf（“%d:%d:%d”,&h,&m,&s）;格式方便处理。

本题中设计几个排序，一个将名称按字母表的顺序进行排列（用strcmp），还有就是将电话记录按照时间进行排序，于是可以将这两个结合起来，然后重载sort中的排序方法，而且这样可以只在开始时排序一次，节省时间。

06:00:00 30 1

08:00:00 30 1

10:00:00 30 1

12:00:00 30 1

5 1

3

**1017**

和1014有点区别，此题中各个窗口前没有等待的队列，直接用一个数组表示各个窗口当前服务的结束时刻即可。本题也用正向的服务结束时刻以及顾客的到达时刻来进行比较即可，这样更方便来计算等待时间差。

因为输入时一块输入的，所以用自定义的sort函数将顾客根据到达时间进行排序。

**1018**

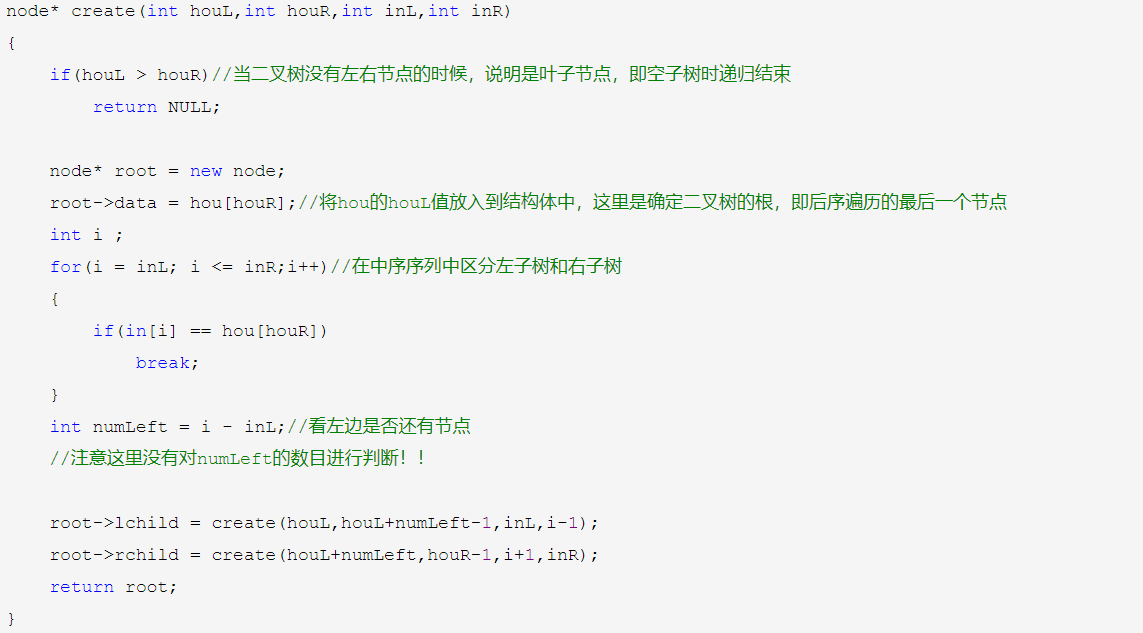
Dijkstra + DFS。如果只有Dijkstra是不可以的，因为minNeed和minBack在路径上的传递不满足最优子结构，不是简单的相加的过程，只有在所有路径都确定了之后才能区选择最小的need和最小的back～  
Dijkstra求最短路径，dfs求minNeed和minBack和path，dfs的时候模拟一遍需要调整的过程，求出最后得到的need和back，与minNeed和minBack比较然后根据情况更新path，最后输出minNeed path 和 minBack，记得path是从最后一个结点一直到第一个结点的，所以要倒着输出～

**1019**

很容易的一题，主要是需要将一个10进制的数换算成b进制的数，然后看这个数是否满足对称数的性质，这里用到vector会很方便，可以知道头尾节点的下标

**1020**

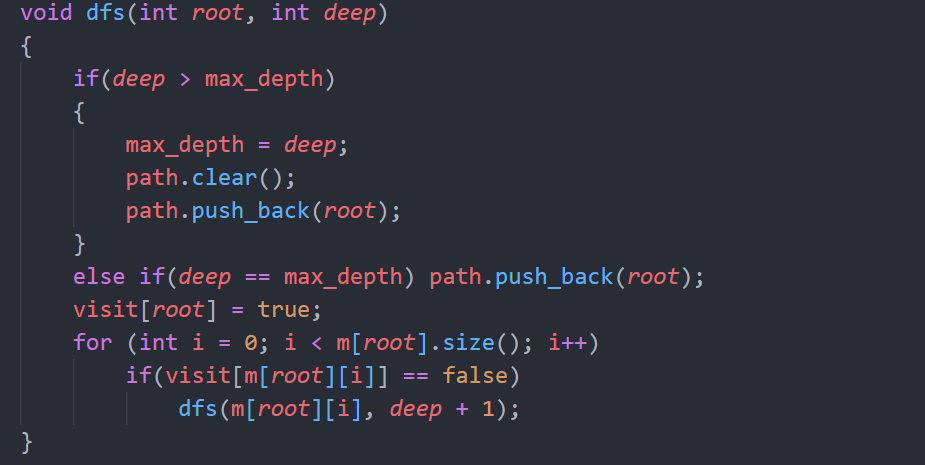
用中序和后序得到二叉树，然后再用层次扫描输出结果。板子如下



层次扫描用queue即可。

**1021**

·用dfs查找树的最大深度



·这题题中说（connected and acyclic）的图是树，要求满足连通且无回路，这就暗含在程序中要先判断是否满足这两条件。题中所给的输入有已经规定了点数N和边数N-1，所以有一下情况：  
 1）首先假设是连通的，因为已经连通，又边数比点数少一，所以必没有回路；

1. 假设不是连通的，此时因为不是连通的，所以在各个连通分量中都可能会出现回路。

综上，因为要排除不连通和有回路的情况，所以只用判断是否连通即可满足判断两者的要求。方法就是用dfs或bfs，将每个点都访问一遍（visit[N]），看有即可连通分量即可。

·正常判断是否含有回路的方法，在任一连通分量中点数要大于边数即可。点数即可在bfs或dfs同时计数，而边数就计数临接表中每个点的边的数量然后再除以二。

·找到树中的最大深度，有两种方法：  
　　1.求出连通图每一个顶点所对应的最大深度，存入到一个数组中，在输出时，只需要遍历这个数组，输出和最大深度相同的结点下标为止。  
　　2.从任意一个顶点开始遍历，在这次遍历过程中，记录下深度最大的顶点小标（可能不止一个），然后从上次遍历记录下的顶点下标（其中任何一个）开始进行深度优先遍历，把具有最大高度的顶点编号记录下来（可能不止一个），两次遍历的并集即为所求（可用set来进行并集处理）。（推荐）

**1022**

要灵活运用map，尤其是要用到字符串匹配的问题时



一个书的查询属性对应多个ID，而且用set能够自然的保持是递增有序的。

类似于Python中对数组中每个元素输出。



输入时也可考虑getline(cin, string); 使用之前用getchar()进行必要的换行

**1023**

读题要认真，简单的题不能因为读题而失误。判断加倍后的数的位数以及原数或只能怪各个数的个数是否都相同。

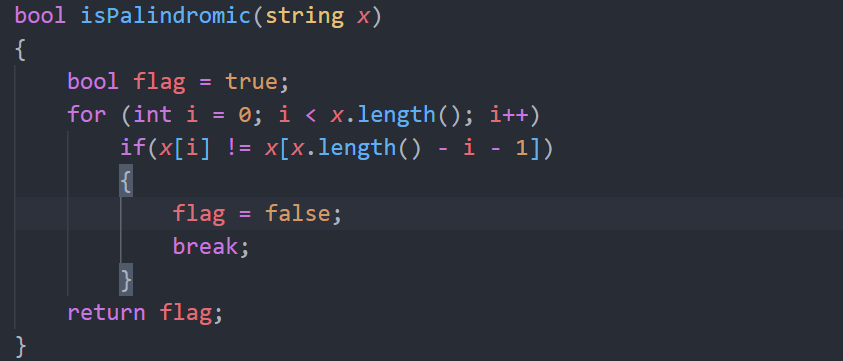
一般大数处理都用字符数组或string，然后按位计算，注意进位符。

对各个数计数一般采用对数组中各个位数进行加和减，不考虑出现次数只考虑出现了哪些可以采用set。

**1024**

遇到大数计算就用string，按位计算。

回文字，Palindromic Number，判断方式。



将string反向reverse(a.begin(), a.end())，在回文字中很方便。

**1025**

先局部sort再全局sort

**1026**

**坑点：**

①playtime超过2小时要压缩为2小时

②虽然本题所有人的到达时间没有超过21点，但是第三组数据有恰好21点到的，此人不能得到服务。（这个点卡了我很久，因为没加入判定是否大于等于21点的）

③vip用户，如果此时有vip桌子空着，用编号最小的vip桌子

④等待时间四舍五入

**思路**

主体思路和1017类似，但由于加入了vip的机制，所以较为复杂，按照下面的分类思路可以将此题AC。

我们可以把这个场景抽象建模，按照到达先后顺序，判断每个会员按照规则所能分配的桌台已经对应时间，分为4类情况——  
　　1 最先空闲下来的是VIP桌子，那么需要检测排队的人群（可能不止一个）中，是否有VIP，如果有，那么先给VIP分配（而不是最早到的那个）；  
　　2 最先空下来的是VIP桌子，但是排队的人群中没有VIP，那么可以给非VIP会员分配VIP桌子；  
　　3 最先空下来的是非VIP桌子，如果排队的第一个人是VIP，那么需要考虑的是，我们现在比较的是最早空下来的桌子和第一个排队的人，但，如果同时存在其他空闲的VIP桌子，需要给VIP安排VIP桌子，所以此时，不能仅仅找到最先空下来的桌子，应该坚持VIP作为第一个排队者，是否存在VIP空闲桌；  
　　4 最后一种情况最简单，空闲的是非VIP桌子，第一个排队的是非VIP，直接分配即可。

**1027**

strncpy(out + i \* 2, t, 2); 很好用，字符串赋值，按地址和个数。

**1028**

printf(“%.6s”); 输出固定位数的字符串

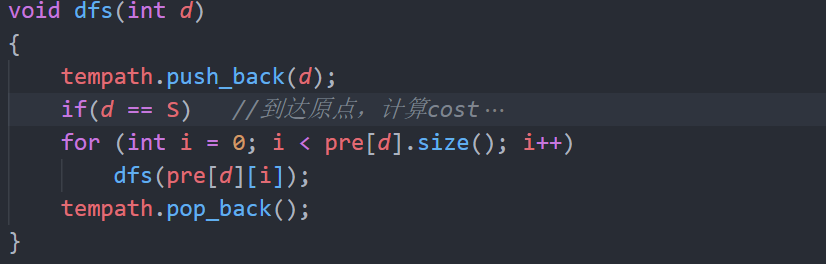
**1029**

如果输入的数很多时，用cin很有可能会超时。

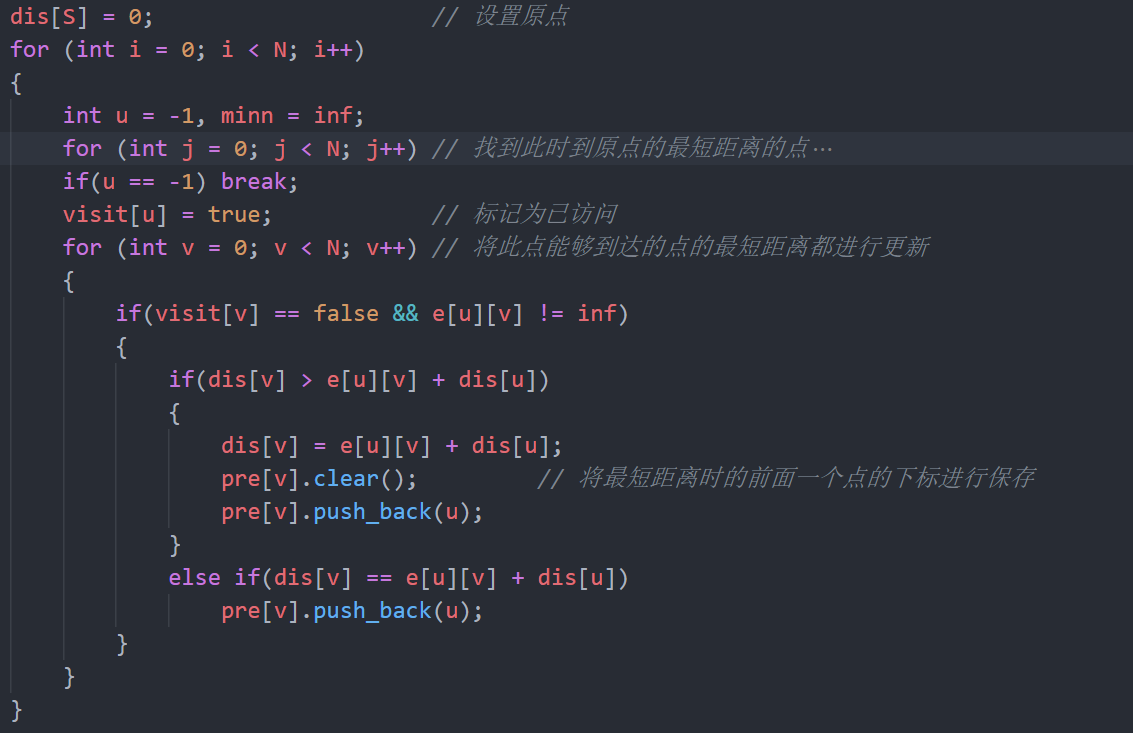
在二路归并时，可以先存储第一个序列，然后对第二个序列边输入边进行归并。

**1030**

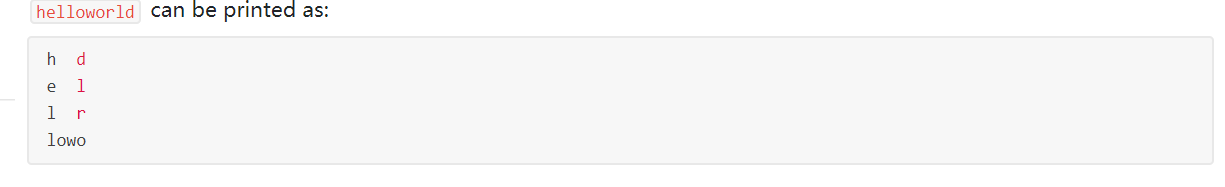
此题和1018类似，先用dijkstra得到每个点到原点的最短距离以及最短距离时每个点前面一个点的下标，然后再用深度优先dfs，对多种情况进行模拟比较，得到结果。

**dfs板子：**  


**dijkstra板子：**



**1031**





此题的重点是计算出n1,n2，由上面的规定可知：

n1 <= n2, 2\*n1 + n2 = N + 2, 故 3\*n1 <= N + 2

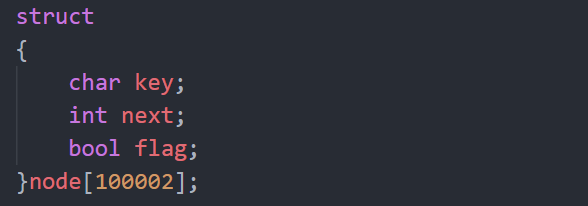
取最大值，故可以求出n1, 从而得出n2，此题就可以解了。

**1032**

不要把题目想得过于简单，要按照题目思路来想代码逻辑。

将所有的节点存在node数组中，并且将flag设置为false，在遍历第一个链表时，将此链表的节点的flag设置为true，在遍历第二个链表时查询节点的flag值，若发现flag为true，这此节点即为要找的共同后缀的起点，在遍历第一个链表的时候，已经将node中属于第一个链表的节点标记出来，当第二个节点访问到true节点的时，表明访问到了既属于第一个链表的节点有属于第二个链表的节点，很容易理解，两个链表的初次交汇点即为共同后缀的起点。

**链表的板子：**



**1033 重点**

这里首先将各加油站按照距离进行**排序**，如果没有距离为0的加油站，则直接输出The maximum travel distance = 0.00；若有则进行下面的判断。

起点的加油站会有一个油价，那么该加多少最为合适？这里要用到**贪心的策略**。

如果在能到达的最大范围内，有加油站的油价比当前的油价低，那肯定加到能够恰好到这个油价比当前低的加油站距离的油量就是一个局部最优策略（这个并不一定是范围内加油站最小的油价）。

如果在最大的范围内没有加油站油价比当前低，那就加满（加满是因为，无论在范围之外有多小多小的油价，即使加满也无法到达，还需要中间一个加油站中转，但既然当前油价最低，能加满就加满，之后油钱高的就可以少加点），下一个到达的加油站因为在范围内油价最小的加油站（即使最小也比当前油价高）

不断的循环这个过程，在正常情况下到达目的地就终止循环，根据上述策略，可以将目的地也看做加油站，其距离设置为相应距离，油钱为0，这样其油钱一定比所有的加油站油钱都要低，在最后一定会选择目的地作为下一到达加油站。但是要注意，有可能当前加油站最近的一个加油站超过了车能行驶的最大距离（即最大的油量\*每升行驶路程），则无法到达目的地，最长距离就是当前加油站的位置+车能行驶的最大距离。

注意：

1）在范围内扫描时，出现价格相等时要去更近的站，因为因为此次油价是由当前决定的，越远的话花费的越贵；

2）因为加油一共就两种情况，加满和加到能够到达下一个油站，那么是否会出现要加油时油箱里的油已经够用了？不会。

可能够用的情况只有可能出现在“加到能够到达下一个油站”中，此处的终点站是此次扫描中油价最低的情况，小于本站，而且油如果要够用就表示在前一站油是加满的，价格要低于此站时。以从“300”到“500”的过程为例：

（1）

0 300 500

6 7 8

这种情况中，从0出发时需要加满到“300”处，然后因为“500”处的价格高于“300”的，所以在300出仍会加满；

（2）

0 300 500

6 8 7

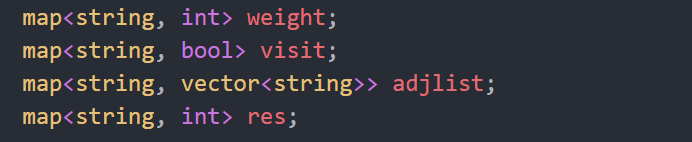
这种情况中，若先不管“0”处的情况，在“300”到“500”的过程中可能出现够用，而因为“500”处的价格低于“300”，所以从“0”开始时就直接去往“500”了。

**1034 重点**

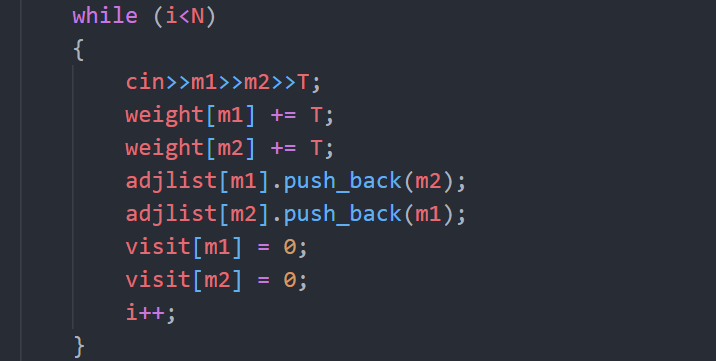
和1013类似，可用并查集和DFS两种方法来做，自己用的并查集结果有几个样例过不了。

这种连通分量的题两种方法都可，用DFS可能容易些。

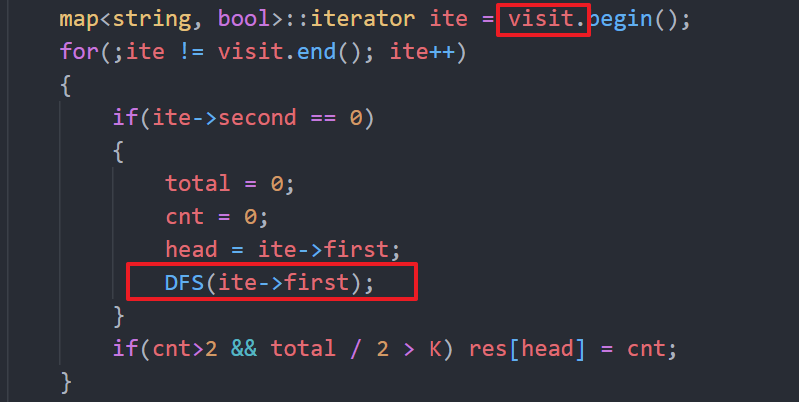
一个通话记录确定两个结点的通路，通话时间可以看做两者的权重，N张通话记录确定一张关系图，而一个团伙对应图中的连通分量，我们可以对整张图做深度优先搜索，确定连通分量的个数，然后判断该连通分量是否存在符合要求的大佬，统计大佬个数。  
　　由于每个人的ID是通过字符串给出的，需要离散化。遍历的过程中需要实时更新权值最大的节点，使用map容器来完成对图的构建，以及使用map来对节点权值的索引；可以说此题就是在考察对map的灵活使用，当要遍历某个数据结构的时候，如果要检索的索引是字符串，就要考虑使用map,并且使用map来存储数据，天然已经按字符串的字典序排好了。最后输出的时候避免排序



map需要熟练掌握，可用直接索引，即使在为空时。



DFS的关键就是visit数组和递归。



**1035**

难度一般，主要是输出内容需要注意，要与题目一致。

能够在主函数中解决的问题就尽量不要在写其他函数了，容易出错。字符数组传引用进入数组时如何判断数组的末尾也指得考虑，最好用string。

**1036**

常规排序题，重写cmp即可。

**1037**

简单的贪心，需要用到排序，用sort即可，读懂题就能写出来。

**1038**

处理0，处理0，处理0！！！

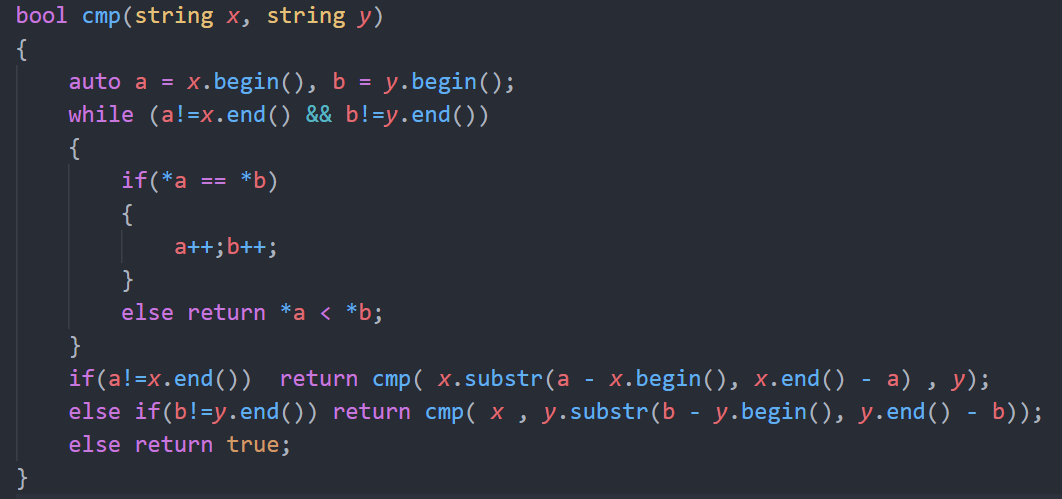
注意题中说的数字是按int还是char来处理，涉及到不同的0的处理方式。

题目给你一些数字的片段（number segments），所以应当用string存储而不是int,希望拼接之后能拼出的最小的数字，这是一道很神奇的题目，我分类讨论分了很多，最后突然发现它的最终解法无比简洁。

其实就是一个序的关系，所有的组合有n!种，（像"所谓组出最小数其实是获得字典序最小的拼接方式"这种废话我就不说了）。假设我们获得了其中的一个组合，然后又两个相邻的数字片段a,b。然后我们就要想，把a和b交换能不能使整个序列变小呢？这个问题的其实等价于b+a 是否小于a+b（此处"+"为连接符）,也就是说对于这样一个序列，如果某两个相邻的元素之间发生交换可以使得整个序列的值变小，我们就应该坚决的交换啊，所以这里定义一个新的序，用<<来表示，若a+b < b + a 则a应当在b前面，即a << b。然后呢，这种序是满足传递性的若a<<b ,b << c，则a<<c，所以迭代到最后，我们就会获得一个任何两个相邻元素都不能交换的局面，也就是所谓的答案。



或者用按位比较的方式，后面有个迭代的部分可以活用到其他题目中。



**1039**

和1022基本上一样，都是要灵活运用map，让id和属性相对应。

主要在循环中输入时使用cin很容易超时，用scanf会好一些

**1040**

可以暴力解，从一个点往左右两边遍历即可，但要注意对称字串是单数还是偶数的，单数的话就从某一点往左右进行遍历，如果是偶数就与左边或右边的数组成一组再想左右遍历

解法二： 利用动态规划解题，dp[i][j]表示子串i->j是否是回文，因此动态方程为dp[i][j] = s[i] == s[j] && dp[i + 1][j - 1].

**1041**

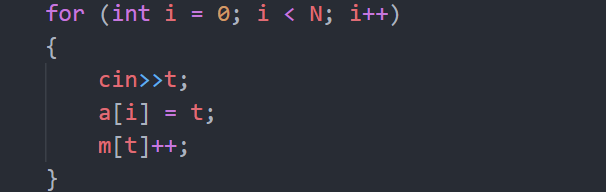
题目的意思是会给出一列数，第一次出现的不重复的数即为所求

5 31 5 88 67 88 17

其中 31、67、17都是不重复的数，但31是第一个出现的，所以答案是31。

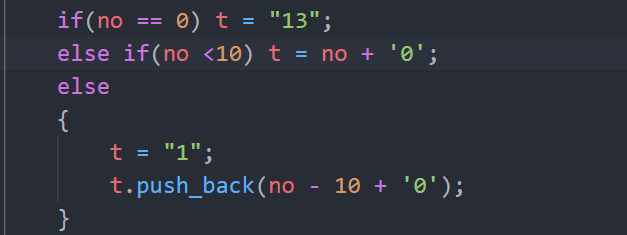
可以模拟一个hash表，用于对各个数进行检索



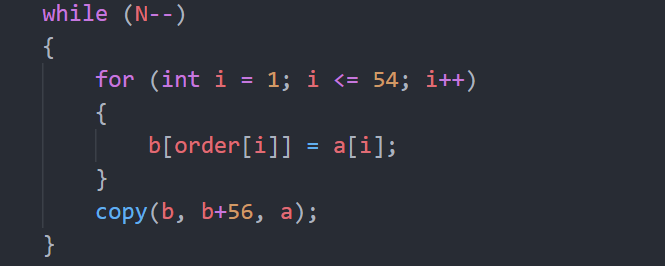


**1042**

一到shuffle的题，题不难，主要是string的处理



如何将1～13转换成字符串的形式。然后再利用按照索引进行读写的方式将a数组每次混洗的结果存入b数组，然后再下次混洗前将b复制给a，最后结果由b输出。



**1043**

树的创建，主要还是用到递归，确定数组中左右子树的下标范围然后进行递归。

此题因为会遇到翻转的问题所以就设置了两个函数分别进行处理。

**1044**

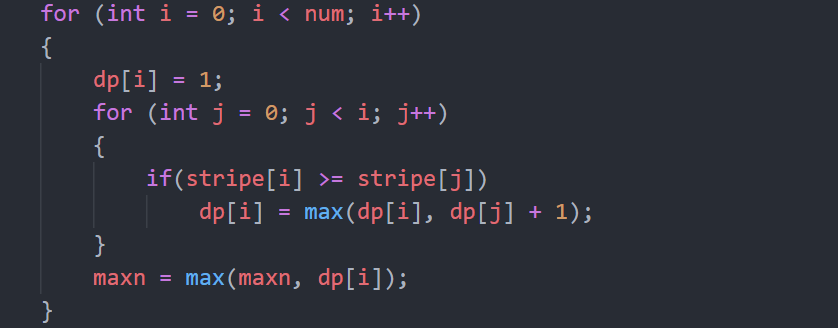
此题不能用dp来做会超时。采用两个“指针”（i、j）来遍历整个数组，当一段范围（i，j）内的数之和大于题目指定的值时，j再往后是没有意义的，所以需要i向前遍历直到和小于指定值。

此题的目的主要是根据题中的要求尽可能地减少操作步骤，不要过多的进行不必要的工作，再次方面需要多多思考。

**1045 重点**

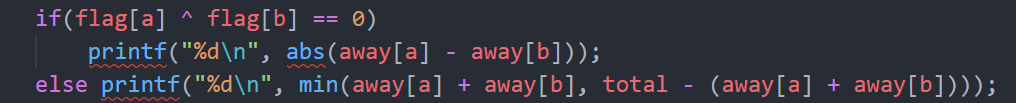
因为喜欢的颜色是不重复的，把喜欢的颜色的序列按照存储到数组中，book[i] = j表示i颜色的下标为j。

先在输入的时候剔除不在喜欢的序列中的元素，然后把剩余的保存在数组a中。按照**最长不下降子序列的方式**做，对于从前到后的每一个i，如果它前面的所有的j，一下子找到了一个j的下标book[j]比book[i]小，此时就更新dp[i]使它 = max(dp[i], dp[j] + 1)；并且同时再每一次遍历完成一次j后更新maxn的值为长度的最大值，最后输出maxn～



**1046**

一个回路上两点之间最小距离的问题，在输入距离时存储整个回路的长度以及每个点到原点的距离，再计算出每个点到原点的最小距离以及是顺时针还是逆时针的，然后只用查表就能得出任意两点之间最小距离的关系



如果两个点到原点的最小距离都是顺时针或逆时针则这两者之间的最小距离就是两者差的绝对值，不相同则是两种情况的最小值。

**1047**

也是学生和课程的问题，不要迷恋用map和set，容易超时。

直接用二维数组的方式



用空间换时间。

然后最后在输出前对一个课头的学生姓名进行sort排序，直接就能实现升序。