NOIP2018初赛题解 提高组

单选

1. 直接进行进制转换即可。可以全部换成10进制。

2. C,C++,Pascal都是编译执行的语言，Python是解释执行。

扩展：JS、PHP也是解释运行语言。解释性灵活但是效率较低。一些解释性语言也有了也能在一定程度上编译，或者使用虚拟机。

3. 今年是第35届NOI，因此第一届NOI是1984年。每次都有这种没啥实际意义的题目。

4. 考虑一个等比数列求和。第一层的节点数是 k^0 ，接下来依次是 k^1,k^2,k^3...k^{h-1} ，也就是说节点总数是 sum\_{i=0}^{h-1}k^i ，根据等比数列求和公式 S\_n=frac{1-q^n}{1-q} ，可知答案为A。当然考场上以三叉树为例举例两三层也可以得到A。

5. 考虑累加， T\_n=T\_{n-1}+n=T\_{n-2}+n-1+n=....

=T\_0+(1+2+3+...+n)=frac{n(n+1)}{2}

这题是2015年提高组初赛原题，如果有在洛谷有题上写过这个题应该不会写错。

6. 建出表达式树，然后先序遍历即可。

7. 我们先考虑固定一个端点的情况，如果左端点固定在了最左边那么答案为12，既然左端点更大，那么肯定答案会小于12，因此只能是B

熟悉ODT（即珂朵莉树）理论的同学可以发现ODT复杂度证明里面有这个东西，即随机意义下期望区间长度。

8. 最简单的办法是把n=1带入，然而熟悉卡特兰数理论的同学会发现A选项的项数就是不对的。

9. 可以考虑每一轮一半人红一半人蓝 蓝的进入下一轮 所以一直是11。

当然蓝色的个数也可以写成 2S\_n=2^{-1}+22^{-2}+32^{-3}+...+n2^{-n} ，

熟悉文化课的同学会发现这是一个等差比数列，那么我们对它进行求和，并且把小量近似掉，得到的结果为12，考虑红是12，可以得到同样的结论。

10. 熟悉树状数组的同学会发现B选项就是lowbit操作，可以得到最后一个1，因此不断进行lowbit即可。当然带入x=2也可以快速判断。

多选

1. 常识题，不过实际上很多考场执行标准不一，有的允许带计算器，有的不允许。更多考场规则可以在NOI笔试题库中找到。不过并找不到官方的考场规则。所以这种题目出的很勉强。

2. 把最底层的节点都画出来向上连，先在符合条件的前提下把每2个点连到一起，这样发现有8个点，可以发现这是上界。

同理把尽量多的3个点连在一起，答案是7，可以发现这是下届，故选CD

当然根据仓鼠(@created\_equal )同学指出，本题还存在一个更加OI向的做法。考虑dp[i][j]表示有i个内部节点,j个叶子节点是否可行。那么显然对于2j...3j都是可行的，画一张图模拟一下转移即可。

初始时dp[0][1]=1

3. 熟悉图论的同学这是一个常识题。Dijkstra是单源最短路算法，因此多次调用必然能够得到所有点对的最短路。如果图中出现负环，那么会无法得出正确答案。

同理负权。

4. 树的性质属于常识。n个点，n-1条边，无环，连通，任意两点有且只有一个路径。

5. BCD属于所谓常识，A项为ACM的创办机构。

问题求解

1. 由于点数很小，手动模拟下。从条件3开始找，即可找到答案。

2. 首先如果b是a的子集，那么条件必然成立。然后手动简单玩一下，发现只有1位和2位情况存在特例。手动找到这些的答案即可。

科学的解释是：设a and b=x,a xor x=y,b xor x=z，则(x+y)(x+z)=x(x+y+z)，即yz=0，即a and b=a或a and b=b

读程序写结果

1. 简单模拟即可。这种题非常需要细心、耐心。以前第一题常常会出复杂表达式求值。

2. 可以看到是读入一个序列，每个点都有一个出度。可以发现这是在找环的个数，手动模拟或者画个图数一下都可以。

3. 可以发现magic(l,r)是对于s[l,r]的字符串哈希，底下枚举了两个子串，那么答案其实就是不重复出现的子串个数，手动数一下就好了。

4. 可以发现这一大堆函数唯一的用处就是找到字典序大于他的下一个排列（其实看到输入都是全排列的元素、以及一个next大概就可以猜到了。），对于第一个询问可以手动找到下10个排列。对于第二个询问，可以把后几位带在一起算，每次看把某一位更新成下一个值需要加上多少。当然也可以直接进行康托展开对全排列进行计数。

完善程序

1. 一个有点鬼畜的双向链表。对于第一个空我们会发现如果a[i]=x的话直接cina[i]其实就好了，所以必然是a[x]=i，意思是x是在第几个位置。

2,3,4空 可以考虑仿写。完善程序出现“复读机”套路是很常见的。当然不能全部复读，要讲究对称性。

5空我们发现题目既然要求第i个数后面最近的一个比他大的，那么必然是i的后继，即R[i]。

当然已经做出前几空了把题面的那组数据带进去就会发现的确是R[i]。

如果原理不懂得可以画图进行模拟。不过反正写题本身很简单。

2.

首先先考虑算法：如果第一家便宜肯定选第一家。

我们考虑程序后半段的思路，大致就是假设能够在第一家买够50000，最少花多少钱。

那么如果打过九五折第一家更优，那么必然会选择第一家。反正剩下的物品都是中间物品。

那剩下的一些呢？我们称为“中间物品”。尽可能选择其中一些“中间物品”在A买，凑足50000元，使得比B便宜，但是尽可能的少。这是一个背包问题。

实现上，f[i,j]表示前i个物品，在额外在A店花了j元的情况下，购买B店“中间物品”的最小值。i呢？滚动数组空间降维。

考虑为什么要先进行这个贪心，如果直接进行正常的背包的话，背包的大小将会是 物品个数 物品大小 ，如果先进行贪心的话，背包的大小会减小到50000.这样就可以接受了。

那剩下的一些呢？我们称为“中间物品”。尽可能选择其中一些“中间物品”在A买，凑足50000元，使得比B便宜，但是尽可能的少。这是一个背包问题。

实现上，f[i,j]表示前i个物品，在额外在A店花了j元的情况下，购买B店“中间物品”的最小值。i呢？滚动数组空间降维。

第一空：如果直接a[i]=b[i]的话上文就算过了，没必要单独循环一次。考虑贪心那么必然是看加了优惠之后a[i]是否优于b[i]；

第二空：考虑printf里面的部分，那么如果a的总和已经满足优惠，直接优惠掉即可；

第三空：考虑仿写min里面的部分，那么肯定是当前枚举的优惠幅度超过了50000。这是考虑如果我们要买第i件，还额外花了j元在“中间物品”上的情况；

第四空：这是计算总价，第一店所有东西打折后，加上所有第二点需要购买的东西；

第五空：考虑它为啥要判j=a[i]，这个是做个背包问题都知道，这是背包问题的转移。