NOIP 模拟题 题解

By liu_runda

简(simple)

考察贪心,排序.

标准算法:考虑 2n 个数字中最小的数字,和它一组的数字产生的价值必然等于这个最小的数字.因此让第 2 小的数字和它一组是最优的.排序后从小到大依次分组即可.

部分分设置:

N=4 的 10%数据可以直接手动讨论.

N<=7 的 30%数据可以通过回溯法求解.

N<=1000 的 50%数据不需要 $O(n\log n)$的排序,可以用冒泡排序解决.

Ai<=100 的 20%数据可以用基数排序进行排序.同时也照顾忘记开 long long 的选手,使他们得到 30 分(包括 N=4 的 10%数据)

单(single)

标签:dfs，树形 DP，高斯消元

分析:本题是一道二合一的问题,两个子任务相互对称.

算法 1:

t=0 的数据最直接的想法是从每个点出发做一遍 dfs,时间复杂度 $O(n^2)$,可以通过第 1 个测试点,期望得分 10 分

算法 2:

t=1 的数据最直接的想法是枚举所有可能的 a[]数组判断是否可行.第 2 个测试点 n<=5,1<=a[i]<=20.注意 $20^5=3200000$,直接暴力搜索 a[i]的取值是可以承受的，可以通过第 2 个测试点,期望得分 10 分，结合算法 1,期望得分 20 分.

算法 3:

t=1 的数据中,b 数组的表达式写出来之后,每个 b[i]对应一个关于 a[i]和树上两点距离的方程,例如 b[1]=dis(1,1)*a[1]+dis(1,2)*a[2]+dis(1,3)*a[3]+...+dis(1,n)*a[n],于是可以列出 n 个方程用高斯消元求解,可以通过第 2,3,4 个测试点,期望得分 30 分,结合算法 1,期望得分 40 分.

算法 4:

对于测试点 5,树退化为 1 条链。这个测试点的作用主要在于启发选手想到正解的思路。

t=0 时，我们分别考虑编号小于 i 和大于 i 的点对 b[i]的贡献.那么离得越远的点对答案的贡献越大.分别考虑每条边对答案的贡献,那么左侧的某条边对答案的贡献就是这条边左侧全部 a[i]之和.实际上，我们对 a[i]分别求取两次前缀和，两次后缀和即可完成对 b[i]的计算.

记 suf(i) 为 a[i]+a[i+1]+....+a[n-1]+a[n],pre(i) 为 a[1]+a[2]+...+a[i-1]+a[i]，则 b[i]=pre(1)+pre(2)+...+pre(i-1)+suf(i+1)+suf(i+2)+...+suf(n-1)+suf(n)

考虑 t=1 的情况.
我们知道 suf(2)+suf(3)+...+suf(n)的值（即 b[1]),知道 pre(1) + suf(3) + suf(4) +...+suf(n)的值(即 b[2]),知道 pre(1)+pre(2)+suf(4)+...+suf(n)的值（即 b[3])，注意到这些式子有很多项是一样的,考虑作差.可以得到下面的式子:
b[2]-b[1]=pre(1)-suf(2),b[3]-b[2]=pre(2)-suf(3).....b[i+1]-b[i]=pre(i)-suf(i+1)
这些式子是有实际意义的,考虑从 b[1]变到 b[2]时答案的变化量,变化的就是 1 和 2 之间连边的贡献.
同时,记 SUM=a[1]+a[2]+...+a[n-1]+a[n]，显然 pre(i)+suf(i+1)=SUM,因此 pre(i)=SUM-suf(i+1),将上面式子中所有 pre 换成 suf，我们就知道了
SUM-2*suf(2) ,SUM-2*suf(3)...SUM-2*suf(n)的取值。
注意我们将 n 个式子作差之后得到了 n-1 个等式，实际上是丢弃了一些信息，只保留了 b[i]之间的相对大小而忽略了 b[i]自身的数值大小.于是考虑 b[1]=suf(2)+suf(3)+suf(4)+...+suf(n),我们发现 suf(2)到 suf(n)都恰好出现了一次,而之前得到了 n-1 个形如 SUM-2*suf(i)的式子，将这 n-1 个式子相加,suf(2),suf(3)...suf(n)前面的系数都是 2,SUM 的系数为(n-1),那么把这个式子加上 2*b[1]，就得到了(n-1)*SUM,将求得的 SUM 代入之前的 n-1 个式子可以得到 suf(2),suf(3),suf(4)......suf(n),差分一下即可得到 a 数组.
时间复杂度 O(n),可以通过第 5 个测试点.推出这个做法,树上的做法也就不难想了.
算法 5(满分算法):
考虑树上的问题.
t=0 的时候,我们可以先暴力计算出 b[1],然后从 b[1]出发开始 dfs，由某个点的父亲的 b[i]推出这个点的 b[i],变化的是这个点和父亲的连边的贡献,也就是这条边两侧的点的 a[i]之和的差值.
t=1 的时候，顺着链上的思路,我们先随便找一个点为根建树，将有边直接相连的两个点的 b[i]作差.设 x 的父亲为 prt[x],以 x 为根的子树中所有 a[i]之和为 sum(x),
SUM=a[1]+a[2]+...+a[n-1]+a[n]，那么 b[x]-b[prt[x]]=(SUM-sum(x))-sum(x).
同链的情况一样,得到 n-1 个式子，将树根的 b[i]也列出式子,可以求出全部 a[i]之和,然后就可以根据之前的式子推出所有的 a[i],和链的做法类似，不再赘述.

题(problem)

计数四合一,考察组合数,卡特兰数,动态规划.
对于 n<=100 的 40%数据:存在一个通用的 DP,定义 f[i][j][k]表示 i 步之后走到(j,k)的方案数,复杂度为 O(n^3).
对于 typ=1 的数据:答案为 catalan 数,使用 O(n)的 catalan 数递推公式或者利用组合数 O(1)计算均可.catalan(n)=C(2n,n)/(n+1)
对于 typ=0 的数据:枚举横向移动了多少步.横向移动 i 步时(为了存在合法解,i 必须是偶数),方案数为 C(n,i)*C(i,i/2)*C((n-i),(n-i)/2)
对于 typ=2 的数据:f[i]表示走了 i 步回到原点的方案数,枚举第一次回到原点时走过的步数 j(为了存在合法解,j 为偶数),则此时方案数为 f[i-j]*catalan(j/2-1),复杂度为 O(n^2)所以最大范围只出到 1000.
对于 typ=3 的数据:枚举横向移动了多少步.横向移动 i 步时(为了存在合法解,i 必须是偶数),方案数为 C(n,i)*catalan(i/2)*catalan((n-i)/2)
n<=1000 的数据是为了照顾只会使用杨辉三角形求解组合数的选手.