题解

1. 近海之主

20pts

暴力枚举排列，然后依次判断当前房间是否能进入。复杂度$O(n! \times n)$

40pts

给存在其他复杂度较大的解法的部分分。

50pts

显然每个宝藏一定能被取到，所以答案就是n。

70pts

钥匙存在的关系可以看成一条边，一个$a_i! = -1$的点就可以连出一条出边，很显然这个图是一个菊花图。

我们考虑对于每个联通块，计算其贡献。我们发现每一个联通块他贡献的点数只和根在其排列中的位置有关，然后我们直接枚举根的位置，判断符合条件的排列有多少种即可。

一个联通块的贡献即: $\sum_{i=1}^{sz} \frac{(sz-1)!}{sz!} \times i$

复杂度$O(n)$。

100pts

同样我们考虑上面连图的方式。首先如果一个联通块内有环，那么它的贡献一定是0。剩下情况中，这个联通块一定是一棵树。我们考虑一个点的贡献，如果一个点最终能被取到，那么一定满足在排列中他的祖先是按深度顺序依次出现的。由于对其有影响的只是祖先节点，其他点可以忽略，只考虑祖先节点的排列，所以一个点被计算到的概率是$1/dep[x]!$，所以总答案$\sum_{i\text{在树内}} \frac{1}{dep[i]!}$

2. 养马

本题即让你求一个dfs顺序，满足各个时刻体力不小于0的最小休息值。

10pts

大力dfs，枚举遍历树的顺序，然后计算答案。

15pts

显然答案=0。

25pts

将边权不为0的边断开分成两个树，决策必然是先把所有和1相连的点走完，然后再走边权不为0的子树。中途计算一下答案即可。

35pts

没有任何决策，直接模拟即可。

50pts

首先在各个节点休息是不必要的，所以我们可以在根节点直接休息最小需要休息的时间，然后再走。

由于我们固定了一定在根节点休息，所以同时我们也需要知道进入一棵子树它最少需要携带的体力值时多少。而且走进一棵子树再走出来，它对体力的花费一定是固定的。

我们定义f_x表示以x为根节点，走完x这个子树最少需要休息的时间。$cost_x$表示以x为根节点对体力的花费。

那么假设我们在rt点计算，当前体力是t，接下来走的儿子是son，与其相连的边权值是w。接下来考虑更新：

1. 若走下去满足不需要休息，则有$f_{son}+w\le t$且$w\le t+cost_{son}-w$
2. 否则就必须休息，直到满足上面的不等式为止，即
   $f_{rt}+=max(f_{son}+w-t,w\times 2-cost_{son}-t)$

然后$w+=cost_{son}$。

所以我们的转移只和儿子遍历的顺序有关，由于二叉树，直接枚举即可。

70pts

由于儿子是不是很多，而这又是一个枚举排列，所以状压儿子。由于$cost$数组与转移无关，所以可以在知道取了哪些儿子，以及取了这些儿子的休息时间，可以快速得到当前的体力值。然后在转移。

或者直接爬山也可以，是要复杂度是对的，基本是可以通过的（没卡）。

80pts

我们先考虑假设当前体力是t，然后不休息是否能走完所有的点。（相当于二分答案）

相当于给定一个集合，集合里的元素用两个值，一个是权值v_i一个是限制l_i，开始手里有权值s，满足$s\ge l_i$，就可以取这个点的权值。

我们把儿子按是否满足$v_i\ge 0$分成两组，我们一定先取$v_i\ge 0$，能取就取。如果取不了，那一定没有解。

所以第一部分对于$v_i\ge 0$直接按l_i排序，然后按顺序取。

对于$v_i < 0$的我们倒着取，也就是先把所有点都取完，然后倒着还原操作，也就是如果存在$s + |v_i| \geq l_i$，那么可以直接去。

这样就把权值变成大于0的了，可以按上面的方法直接取，如果无解那一定无解。因为如果存在一个解满足条件，那么这个点倒着取一定也是满足条件的，即它满足了步步满足了$s_i \geq l_i$，也就是步步满足了$s_i - |v_i| + |v_i| \geq l_i$。

100pts

可以在每一个点排序完之后二分答案。

其实并不需要二分，考虑上面决策的顺序，如果存在了非法，那可以直接在当前体力上加上差值，然后最后的答案就是加上的体力值了，这样同样能保证最优解。

3. 膜拜大会

由于n, m同阶（测试点3除外），复杂度计算时只计算n。另外，下面的题解中默认算方案数，最后的概率就是方案数乘以A_n^m的逆元。

另外，由于各种排列组合在100pts算法中基本都会用到，所以不在非100pts档中一一说明。

20pts

直接暴力dfs搜索，可以通过$n = 9$的小测试点。

30pts

由于$m = 1$，所以只有一种决策方案，枚举这种决策后$O(1)$计算即可。复杂度$O(n)$。

50pts

首先，下面的所有内容（包括70pts及100pts）是基于“如果$A_i = 0$，那么不符合条件”的，所以如果$K = 0$直接特判输出1。

考虑只有一个$A_i > 0$时怎么写。设只有$A_p > 0$。

如果$p \neq 1$，那么需要计算这个点的值“传递”到1上的概率。由于是一个环，传递有两种方向，对于每个方向，如果需要经过c个点（不包括1，包括p），那么这c个点都要被选，且按照p到1的顺序递增。同时，不能在最后一个点被选后再选1。**由于$m < n - 1$，所以不可能同时从两个方向传过来**（这很重要，要不就是另一道题了），计算两个方向的概率求和即可。这个概率需要用到一些组合数学推导。具体推导见100pts部分。

如果$p = 1$，那么分以下几种情况（当然由于出题人良心，你不判$p = 1$也能过一个点）：

1. 如果$A_1 < \frac{K}{2}$，那么显然答案为0。
2. 如果$\frac{K}{2} \leq A_1 < K$，那么当且仅当1、2、n都被选，且1是三者中最先选的。可以用组合数学计算概率。同见100pts部分。
3. 如果$A_1 \geq K$，那么只要不是选了1，且在1之后没有2和n即可。枚举选不选2和n，如果都不选容易计算，如果选一个要把它放在1后面，如果选两个至少有一个在1后面，分别用排列组合计算。

70pts

相当于计算50分的“不是选了1，且在1之后没有2和n”的概率。不详细说了。

100pts

~~（一开始我以为分类讨论很少的但是不知道为什么标算写着写着就发现自己一大堆东西没判需要各种讨论了不过其实如果从上面的档一点一点过渡过来还是蛮自然的吧qwq~~）~~

如果对于一种方案，存在最大的r使每个$i \in [2, r]$都被选，且i号在$i + 1$号之前，那么称r是这种方案的“极右点”（如果2不被选，$r = 1$）。同理可以定义出“极左点”l。不难发现，如果$1, 2, n$都被选，且1是三者中最先选的，最后A_1是l到r这一段环上的和加上原先A_1的值；否则，最后A_1就是l到r这一段环上的和。

将这两部分分开考虑。先讨论2和n都被选的情况（因为最复杂），那么可以枚举l或r，然后求出要使这一段的和大于等于K（或$K - A_1$），另一个极点的范围。这样，问题就转化成以下这样：

在n个元素中有序地选出m个，另有一个参数t。n个元素中有3类特殊元素：A,B,X。A和B分别有c_A和c_B个，必须全选，且各自必须按特定顺序选；X只有一个，若t为0，那么X必须选，且在最后一个A和最后一个B之前选；否则，X可以不选，但如果选，必须在最后一个A和最后一个B之间选。求方案数$f(c_A, c_B, t)$。

这个问题看起来很复杂，但可以发现，符合要求的选法不外乎以下几种（设最后一个A为a，最后一个B为b）：

1. X不选，AB各自有序，符合$t=1$。
2. X选，X、a、b的相对顺序为Xab，符合$t=0$。
3. X选，X、a、b的相对顺序为Xba，符合$t=0$。
4. X选，X、a、b的相对顺序为aXb，符合$t=1$。
5. X选，X、a、b的相对顺序为bXa，符合$t=1$。

其中，2和3、4和5是对称的。所以讨论1、2、4三种的求法。

1：相当于先在m个位置中选c_A+c_B个，再在c_A+c_B个选出c_A个为A，最后在$n-c_A-c_B-1$中选$m-c_A-c_B$个，方案$C_m^{c_A+c_B}C_{c_A+c_B}^{c_A}A_{n-c_A-c_B-1}^{m-c_A-c_B}$。

2：相当于先在m个位置中选c_A+c_B+1个，再在c_A+c_B个选出c_A+1个为A和X（剩下c_B-1个是除了b外的B），接着在c_A个位置中选一个插入X，然后把b放到这c_A+c_B个元素的后面，最后在$n-c_A-c_B-1$中选$m-c_A-c_B-1$个，方案数$c_AC_m^{c_A+c_B+1}C_{c_A+c_B}^{c_A+1}A_{n-c_A-c_B-1}^{m-c_A-c_B-1}$。

4：相当于先在m个位置中选c_A+c_B+1个，再在c_A+c_B个选出c_A+1个为A和X（剩下c_B-1个是除了b外的B），接着在a后面插入X，然后把b放到这c_A+c_B个元素的后面，最后在$n-c_A-c_B-1$中选$m-c_A-c_B-1$个，方案数$C_m^{c_A+c_B+1}C_{c_A+c_B}^{c_A+1}A_{n-c_A-c_B-1}^{m-c_A-c_B-1}$。

3和5同理。

然后求出大于等于K的区间的$t=1$型询问，求出大于等于$K-A_1$的区间的$t=0$型询问。

但是有一个小问题：如果直接计算以上值，那么会导致某些方案算重。归根结底，是因为以上f函数计算的是“到极左点距离大于等于c_A，到极右点距离大于等于c_B”的方案数，而我们需要的是“到极左点距离恰好等于c_A，到极右点距离大于等于c_B”（左右互换亦可）的方案数。容易发现这个方案数$g(c_A,c_B,t)$可以容斥，即$g(c_A,c_B,t)=f(c_A,c_B,t)-f(c_A+1,c_B,t)$。然后就可以直接算了。

那么，2和n都被选的情况就讨论完了。剩下还有

1. 2被选n不被选
2. n被选2不被选
3. $2,n$都不被选

的方案没有计算（不过都比较简单）。前两者对称，考虑第2和3种。

第一种中，$l=1$，可以求出对应的r，问题变为：

在$n-1$个元素（去掉n）中有序地选出m个。n个元素中有2类特殊元素：A,X。A有c_A个，必须全选，且必须按特定顺序选；X只有一个，可以不选，但如果选，必须在最后一个A之前选。求方案数$f'(c_A)$。

这就只有两种情况：

1. X选。
2. X不选。

1：相当于先在m个位置中选c_A+1个，然后把X插入c_A个位置中的一个，最后在$n-c_A-2$中选$m-c_A-1$个，方案数$c_AC_m^{c_A+1}A_{n-c_A-2}^{m-c_A-1}$。

2：相当于先在m个位置中选c_A个，然后在$n-c_A-2$中选$m-c_A$个，方案数$C_m^{c_A}A_{n-c_A-2}^{m-c_A}$。

不用容斥，直接算出$f'(c_A)$即可。

第二种同理；第三种情况则更简单，如果$A_1\geq K$，那么除了$1,2$和n随便选，方案数A_{n-3}^m。

最后把所有情况加起来就是总方案数。复杂度$O(n)$，常数略大，但跑10^5的数据还是绰绰有余的。

如果有更简洁的写法欢迎大佬打脸。