# disk

- 离散后用栈模拟 O(nlgn)
-
- 倒序用栈模拟 O(n)

# sequence

根据数据大小，考虑dp。

先设个fi,j,ii表示dp到第i位，已用j个数，cnt等于ii

考虑第i位放几个数，然后发现不能确定第i位最终是0还是1，因为存在之前的进位。

于是加1维以避免后效性，jj表示从第i位的进位次数。

则可以进行转移，设第i位放a个数，有：

$$f_{i+1,\,j+a,\,ii+(\lfloor \frac{jj}{2} \rfloor + a)\&1,\,(\lfloor \frac{jj}{2} \rfloor + a)} \leftarrow f_{i,j,ii,jj} \times \overset{\text{权值}}{v_i^a} \times \overset{\text{方法数}}{\binom{n-j}{a}}$$

# rollcall

- **堆**
- 维护前k-1大的大根堆 A
- 维护其余的小根堆 B
- 插入：A.insert() B.insert(A.pop())
- 询问：answer := B.top(); A.insert(B.pop())
- 离散化+树状数组
- 离线+链表/并查集

# tree

先考虑暴力的做法。

还是贪心的逐位确定，逐位确定判有没有解，相当于下面的问题：
树上有一些路径，一条路径表示要把$x$的数字换到$y$去，问有没有解。

对于一条路径$p[1], p[2], ..., p[m]$，限制如下：

1.$(p[1], p[2])$是$p[1]$的所有相邻边中时间最小的。
2.$(p[[m-1], p[m])$是$p[m]$的所有相邻边中时间最大的。
3.$\forall i \in [1, m-2]$，$(p[i], p[i+1])$和$(p[i+1], p[i+2])$在$p[i+1]$的所有相邻边中时间是相邻的（前小于后）。

发现所有限制都是对于一个点的相邻边的，因为是树，所以不同点之间的相临边限制不会影响。

那么只看每个点的相邻边是否有满足条件的解。

暴力的做法就是先把3限制的缩成若干段段（段内要合法），然后若$T(i) < T(j)$，则$i->j$连边，看有没有环就好了。

仔细思考，除了3限制就只有最小最大限制，那么可以总结为以下几个限制：
1.每一段是合法的（不能有反向边、跨越边）
2.min前面不能有小于它的
3.max后面不能有大于它的
4.min、max若处于一段，则这一段的长度必须是中转点的度数

复杂度没怎么变，一共要check$O(n^2)$次，每次$O(n)$。

考虑对于每一位，不去枚举它选什么，而是先求它能选什么，再从中选最小的。

假设第?位的起点是$x$，以它为根，dfs一遍，处理上面四条限制，就可以求出每个点能不能作为终点了！

需要并查集维护段，时间复杂度：$O(T * n^2\alpha)$