1. 最大值(max)

Source: CSA Expected Max

注意到m比较小，不妨分开考虑每个元素在哪些操作中被加了.

考虑状压DP，先计算$f(i,S,j)$表示A_i在进行S集合的操作之后，值为j的概率.

设$dp(i,j,S)$表示前i个元素的最大值为j，用掉了S集合的操作的概率. 转移时枚举当前元素用掉了哪些操作，以及进行这些操作后的值. 即：

$$dp(i-1,j,S_1)\times f(i,S_2,k)\to dp(i,max(j,k),S_1+S_2)$$

$O(n(cm)^2 3^m)$.

2. 染色(paint)

Source: AtCoder Regular Contest 063 F : Snuke's Coloring 2

题目实际上是要找一个周长最大的矩形，内部不包含任何关键点.

可以发现一个小性质：答案的下界为$2\times(max(w,h)+1)$，因此这个矩形一定会经过$x=\frac{w}{2}$或$y=\frac{h}{2}$. 先考虑经过$x=\frac{w}{2}$的情况，另一种情况是一样的.

先将坐标离散化. 枚举矩形的上边界y_R，对于每一个下边界y_L，我们可以计算出矩形的最优左边界$x_L=\min\{X_i|Y_i\in[y_L,y_R],X_i>\frac{w}{2}\}$，以及右边界$x_R=\max\{X_i|Y_i\in[y_L,y_R],X_i\le\frac{w}{2}\}$，此时可以找到一个周长为$2\times(x_R-x_L+y_R-y_L)$的矩形.

直接做是$O(n^2)$的，但该算法可以用线段树优化，在将上边界往上移的过程中动态维护每个位置的x_L,x_R，并维护全局最小值，不难发现只需要左右各开一个单调栈，在更新单调栈时在线段树树上进行区间加减即可. $O(n\log n)$.

就算没有观察到上面的小性质，也可以多套一层分治解决，复杂度多一个log.

3. 剖分(decompose)

3.1 Chain

方便起见先考虑链上$L=2$的情况，不难想到DP:

$$dp(i,2)=dp(i+1,1)+w[i][2]$$
$$dp(i,1)=\max\{dp(i+1,1),dp(i+1,2)\}+w[i][1]$$

我们可以将这个转移写成矩阵，注意这里的矩阵乘法是重新定义的–乘法变为加法，加法变为取max. 不难发现此时它仍满足结合律（与Floyd算法类似）:

$$\begin{pmatrix} dp(i+1,1) & dp(i+1,2) \end{pmatrix} \times \begin{pmatrix} w[i][1] & w[i][2] \\ w[i][1] & -\infty \end{pmatrix} = \begin{pmatrix} dp(i,1) & dp(i,2) \end{pmatrix}$$

$L>2$的情况是类似的，用线段树维护一个区间的转移矩阵的乘积，每次只需要单点修改一个矩阵，$O(q\log nL^3)$.

3.2 Tree

考虑树链剖分，重链上的转移与链上的情况是类似的，只是此时还要加上轻儿子信息. 设$f(i)=\max_{j=1}^{L} dp(i,j)$，下面仍以$L=2$为例.

$$dp(i,2)=\max_{c\in child(i)}(dp(c,1)-f(c))+\sum_{c\in child(i)} f(c)+w[i][2]$$
$$dp(i,1)=\sum_{c\in child(i)} f(c)+w[i][1]$$

设重儿子为s, 记轻儿子信息

$$sumc(i)=\sum_{c\in child(i),c\neq s} f(c)$$
$$maxc(i)=\max_{c\in child(i),c\neq s}(dp(c,1)-f(c))$$

.

可以写出矩阵:

$$\begin{pmatrix} dp(s,1) & dp(s,2) \end{pmatrix} \times \begin{pmatrix} w[i][1]+sumc(i) & w[i][2]+\max\{maxc(i),0\}+sumc(i) \\ w[i][1]+sumc(i) & w[i][2]+maxc(i)+sumc(i) \end{pmatrix} = \begin{pmatrix} dp(i,1) & dp(i,2) \end{pmatrix}$$

将轻儿子信息视为常数，我们就可以快速弄出一条重链的转移矩阵，然后对于一条重链的父亲，我们需要更改它的轻儿子信息，其中$maxc(i)$需要开一个set维护.

$L>2$的情况是类似的，$O(q\log^2 nL^3+nL^3)$