# MAX

考虑达到型dp
记dp[i][j]，前i个事件过后，能否存在一个方案使得当前值为j
能：dp值为1　不能：dp值为0
时间复杂度O(n*max)

# NUM

稍微分了点部分分

最开始的10分：.......

接下来的30分：dp[i][j][k]表示，在k时刻，x坐标是I,y坐标是j，能收集到的最大金币数

最后的60分：dp[i]表示，最后收集了第i个金币能收集到的最大的金币数

转移很简单呢，可以直接枚举前一个金币

如果 time[i]-time[j]>=abs(x[i]-x[j])+abs(y[i]-y[j])

dp[i]=max(dp[i],dp[j]+1)

时间复杂度O(n^2)

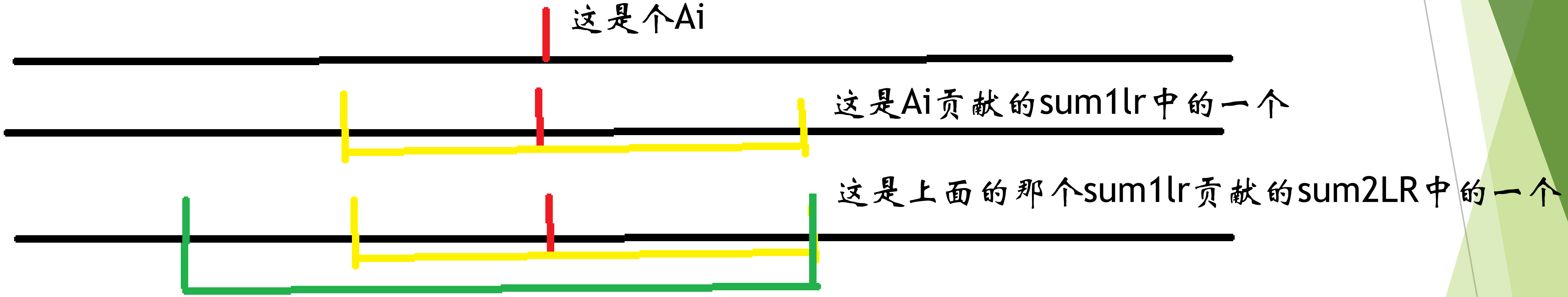
# SUM

前三十分随便照着题意模拟一下就有了，算是个读题分

考虑正解（?　　我们先枚举所求的r，也就是定了个边界

然后我们考虑每一个Ai的贡献

我们抽丝剥茧，一层一层的考虑




我们可以发现，这是一个嵌套的结构，所以Ai对答案的贡献就是不同的嵌套的个数*Ai

所以我们可以分别考虑Ai的左边和右边

对于每一边，实际就是有顺序的选取(k-1)个下标，且后一个的下标>=前一个下标

容易发现可以用组合数计算 $sum_{k1r} = \sum_1^r a[i] * C_{i+k-2}^{k-1} * C_{r-i+k-1}^{k-1}$

关于ntt,我还是提一句。对于每一个r，发现组合数的上指标相加等于k*2+r-3，是一个定值

所以是一个很显然的加法卷积，使用ntt优化即可实现nlogn

# CNT

我不会n^2的暴力……如果有会的人我只能说您吊打我
我们发现题面的两个限制非常强，所以我们可以直接模拟插入删除的操作
我们真的去维护两个集合A和B
对于add x
　　如果A集合最大的数>x，我们就把他放进A
　　如果B集合最小的数<x，我们就把他放进B
　　上面两种情况，我们发现x的去向都是确定的
　　但是如果不符合上面的情况呢？？
　　！我们可以引入一个中间区域，把x放进这个缓冲区
对于accept x
　　如果x出现在A集合，如果是最大值，直接删除，否则直接输出0
　　如果x出现在B集合，如果是最小值，直接删除，否则直接输出0
　　但是如果x在中间的缓冲区呢？？
　　我们可以发现，为了使x合法，对于比x先进入缓冲区的元素，如果比x小，就一定要进入A，如果比x大，就必须进入B，正确性显然
　　清除完比x先进入缓冲区的元素后，也就是我们该讨论x的去向了，我们发现x去A或者B都可以，所以答案*2
　对于所有操作完成，这时候缓冲区可能还剩下一些元素，不知道去A还是B，这时我们可以随便钦定前k小去A，剩下的去B，所以使答案乘(cnt+1)