A

注意到整个矩形的最小值 L 和最大值 R 一定在同一侧。

不失一般性，令其为 A 侧，则代价可以写成 $\max\limits_{y\in B}\max\limits_{x\in A}|x-y|=\max\limits_{y\in B}\max\{|y-L|,|y-R|\}$。

所以，从 B 中移除一个元素不会变的更劣，而 B 一定包含角上的那个数。

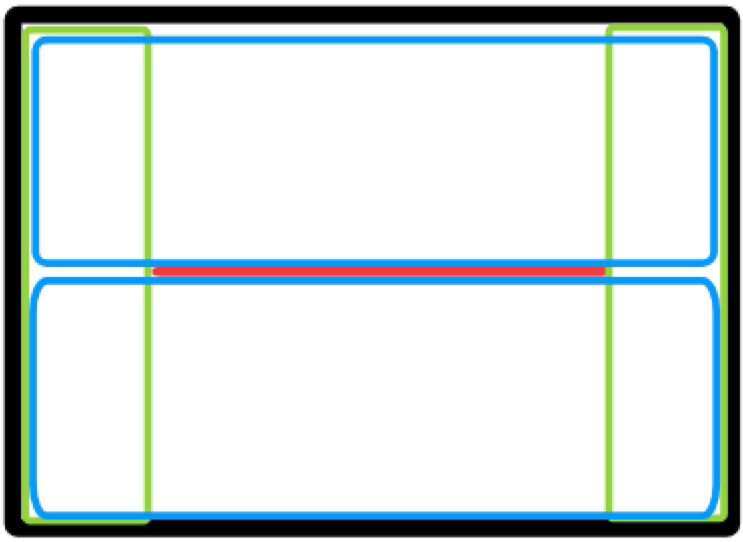
综上所述，答案只有两种情况：A 只包含右上角，B 只包含左下角。两种情况取最优即可。

B

预处理出每个位置往上，往下，往左，往右的最大延伸长度，然后就可以对于每个 i，计算出前 i 行，后 i 行，前 i 列，后 i 列，这四个子矩形内部的，选取一个长条的最大长度。

考虑二分答案。

枚举起点和方向，调用之前提到的这四个预处理结果（如图所示，两个蓝框和两个绿框）即可。



C

观察到一个性质，每一步往后跳的时候，一定是选能成为 m 个数列中可达的最靠前的一个（最后一步除外），即每个点只有 m 个后继状态，考虑倍增。

令 $f(2^k,i,j)$ 表示，从数 i 开始跳 2^k 步落在第 j 个序列上，能跳到的最小的下标是多少，合并是容易的，枚举前 2^{k-1} 步跳到哪个序列上即可。

查询的时候从大到小枚举 k，同样维护跳目前的步数在 m 个序列中可达的最前位置，合并类似，算出可以在接下来的一步跳到目标的最小代价，注意特判 $ans=1$ 和无解，复杂度 $O((n+q)m^2\log n)$。

D

一个非常直观的结论：换乘一定是从 B_i 大的换到 B_i 小的，也就是说经过的所有点（除了终点），B_i 是递减的。

将所有 i 按 B_i 从大到小排序，令 $dp(u)$ 表示到达点 u 所需的最小代价，转移：
$dp(u)=\min_{v|B_v>B_u}\{dp(v)+A_v+B_v\cdot dist(u,v)\}$。

将 $dp(v)+A_v+B_v\cdot dist(u,v)$ 中，B_v 看成斜率，$A_v+dp(v)$ 看成截距，$dist(u,v)$ 看成自变量 x。考虑点分树，在点分树上的所有点上维护一个凸包，计算完 $dp(v)$ 之后，在点分树上所有 v 的祖先 fa 上的凸包，加入直线 $y=B_vx+(B_v\cdot dist(v,fa)+A_v+dp(v))$，因为 B 是递减的，用单调栈维护凸包，查询时二分查询 $x=dist(u,fa)$ 处的 y，复杂度 $O(n\log^2 n)$。

有个小细节，路径上经过的所有点中，终点不保证是 B_i 递减的，需要额外计算最后一步，可以用一样的方法计算。