Solutions

ZYF

Hengyang No.8 High School

June 10, 2016

人生的经验(life)

- 给出一个大小为c的字符集

人生的经验(life)

- 给出一个大小为c的字符集
- 求出一个包含长度为l的所有c^l个字符串的最短母串

人生的经验(life)

- 给出一个大小为c的字符集
- 求出一个包含长度为l的所有c^l个字符串的最短母串
- 数据范围：$c \leqslant 26$，输出文件大小不超过10M

算法一

对于$l = 1$的数据，直接输出字符集即可。

算法一

对于$l = 1$的数据，直接输出字符集即可。

时间复杂度：$O(c)$

期望得分：5分

算法二

显然答案的上界为$l \cdot c^l$，因为我们只要将所有的字符串顺次相接即可。

算法二

显然答案的上界为$l \cdot c^l$，因为我们只要将所有的字符串顺次相接即可。

确定上界后我们直接暴力DFS，加上一些可行性剪枝和枚举顺序上的优化后，可以通过前几个小测试点。

算法二

显然答案的上界为$l \cdot c^l$，因为我们只要将所有的字符串顺次相接即可。

确定上界后我们直接暴力DFS，加上一些可行性剪枝和枚举顺序上的优化后，可以通过前几个小测试点。

时间复杂度：$O(?)$

期望得分：15-30分

算法三

首先，我们来分析一下答案的下界。

算法三

首先，我们来分析一下答案的下界。

长度为n的母串有$n-l+1$个长度为l的子串，而最理想的情况是这$n-l+1$个子串互不相同。

算法三

首先，我们来分析一下答案的下界。

长度为n的母串有$n-l+1$个长度为l的子串，而最理想的情况是这$n-l+1$个子串互不相同。

所以$Answer-l+1 \geqslant c^l$，也就是$Answer \geqslant c^l+l-1$。

算法三

首先，我们来分析一下答案的下界。

长度为n的母串有$n-l+1$个长度为l的子串，而最理想的情况是这$n-l+1$个子串互不相同。

所以$Answer-l+1 \geqslant c^l$，也就是$Answer \geqslant c^l+l-1$。

接下来，我们考虑证明这个下界确实是可以达到的。

算法三

要想达到这个下界，我们必须要将c^l个不同的子串按一定顺序连接起来，保证相邻的两个子串能共用$l-1$个字符。

算法三

要想达到这个下界，我们必须要将c^l个不同的子串按一定顺序连接起来，保证相邻的两个子串能共用$l-1$个字符。

先考虑一个比较简单的建图。以c^l个子串为顶点，如果子串A的后$l-1$个字符与子串B的前$l-1$个字符完全相同，就从A向B连一条有向边。

算法三

要想达到这个下界，我们必须要将c^l个不同的子串按一定顺序连接起来，保证相邻的两个子串能共用$l-1$个字符。

先考虑一个比较简单的建图。以c^l个子串为顶点，如果子串A的后$l-1$个字符与子串B的前$l-1$个字符完全相同，就从A向B连一条有向边。

那么问题转化为在该图中找一条不重复经过任意顶点的路径，也就是哈密顿路径，显然这是一个NPC问题。

算法三

重新建模，我们考虑将点的信息转移到边上。

算法三

重新建模，我们考虑将点的信息转移到边上。

以c^{l-1}个长度为$l-1$的前缀串为顶点，每个顶点向外连出c条边，分别表示以这个顶点为前缀的c个长度为l的子串。

算法三

重新建模，我们考虑将点的信息转移到边上。

以c^{l-1}个长度为$l-1$的前缀串为顶点，每个顶点向外连出c条边，分别表示以这个顶点为前缀的c个长度为l的子串。

而对于所有c^l个子串，我们从它前缀串对应的顶点向它后缀串对应的顶点连一条有向边，边上记录的是这个子串的最后一个字符。

算法三

重新建模，我们考虑将点的信息转移到边上。

以c^{l-1}个长度为$l-1$的前缀串为顶点，每个顶点向外连出c条边，分别表示以这个顶点为前缀的c个长度为l的子串。

而对于所有c^l个子串，我们从它前缀串对应的顶点向它后缀串对应的顶点连一条有向边，边上记录的是这个子串的最后一个字符。

此时我们发现，问题转化为在新图中找一条不重复经过任意**边**的路径，也就是欧拉路径。

算法三

而且可以发现，图中每个顶点的入度和出度都是c，所以该图必定存在一条欧拉回路。

算法三

而且可以发现，图中每个顶点的入度和出度都是c，所以该图必定存在一条欧拉回路。

于是我们使用圈套圈算法，找出一条欧拉回路即可。

算法三

而且可以发现，图中每个顶点的入度和出度都是c，所以该图必定存在一条欧拉回路。

于是我们使用圈套圈算法，找出一条欧拉回路即可。

由于这题连边很特殊，所以我们并不需要建出实际的图，直接用顶点相对应字符串的BKDR哈希值来操作即可，这样做可以减少常数。

算法三

而且可以发现，图中每个顶点的入度和出度都是c，所以该图必定存在一条欧拉回路。

于是我们使用圈套圈算法，找出一条欧拉回路即可。

由于这题连边很特殊，所以我们并不需要建出实际的图，直接用顶点相对应字符串的BKDR哈希值来操作即可，这样做可以减少常数。

时间复杂度：$O(c^l)$

期望得分：100分

基因改造计划(gene)

- 给出一个长度为n的母串S

基因改造计划(gene)

- 给出一个长度为n的母串S
- 每次询问给出区间$[l, r]$，求出$S[l..r]$内回文子串的个数

基因改造计划(gene)

- 给出一个长度为n的母串S
- 每次询问给出区间$[l,r]$，求出$S[l..r]$内回文子串的个数
- 数据范围：$1 \leqslant n, m \leqslant 130000$

一个拿衣服的做法

对于每个询问区间，枚举区间内的所有子串，暴力判断它是不是回文串。

一个拿衣服的做法

对于每个询问区间，枚举区间内的所有子串，暴力判断它是不是回文串。

时间复杂度：$O(n^3m)$

期望得分：5分

另一个拿衣服的做法

我们不直接枚举所有子串，而是枚举区间内所有字符，暴力向两边扩展，求出以这个字符为中点的回文串有多少个。

另一个拿衣服的做法

我们不直接枚举所有子串，而是枚举区间内所有字符，暴力向两边扩展，求出以这个字符为中点的回文串有多少个。

但注意长度为偶数的回文串中点不是字符，类似处理一下。

另一个拿衣服的做法

我们不直接枚举所有子串，而是枚举区间内所有字符，暴力向两边扩展，求出以这个字符为中点的回文串有多少个。

但注意长度为偶数的回文串中点不是字符，类似处理一下。

时间复杂度：$O(n^2m)$

期望得分：10分

算法一

测试点0-3的询问数很少，我们可以直接将每个询问对应的子串拿出来，做一遍Manacher算法得出答案。

算法一

测试点0-3的询问数很少，我们可以直接将每个询问对应的子串拿出来，做一遍Manacher算法得出答案。

时间复杂度：$O(nm)$

期望得分：20分

算法二

测试点4-5的字符串长度比较短，我们考虑动态规划。

算法二

测试点4-5的字符串长度比较短，我们考虑动态规划。
设$w[i][j]$表示子串$S[i..j]$是否是回文串，那么显然有

$$w[i][j] = (S[i] = S[j]) \text{ and } w[i+1][j-1]$$

算法二

测试点4-5的字符串长度比较短，我们考虑动态规划。
设$w[i][j]$表示子串$S[i..j]$是否是回文串，那么显然有

$$w[i][j] = (S[i] = S[j]) \text{ and } w[i+1][j-1]$$

设$f[l][r]$表示区间$[l, r]$内的回文子串个数，则

$$f[l][r] = \sum_{i \geqslant l, j \leqslant r} w[i][j]$$

算法二

类似二维前缀和，我们可以得到递推式：

$$f[l][r] = f[l+1][r] + f[l][r-1] - f[l+1][r-1] + w[l][r]$$

算法二

类似二维前缀和，我们可以得到递推式：

$$f[l][r] = f[l+1][r] + f[l][r-1] - f[l+1][r-1] + w[l][r]$$

时间复杂度：$O(n^2 + m)$

期望得分：20分

（与算法一结合能够拿到30分）

算法三

由于不存在修改操作，且所有的询问都未加密直接给出，我们考虑使用莫队算法。

算法三

由于不存在修改操作，且所有的询问都未加密直接给出，我们考虑使用莫队算法。

那么现在需要思考的问题就是，如何由区间$[l, r-1]$的答案迅速得出$[l, r]$的答案。（其余三种情况做法类似）

算法三

由于不存在修改操作，且所有的询问都未加密直接给出，我们考虑使用莫队算法。

那么现在需要思考的问题就是，如何由区间$[l, r-1]$的答案迅速得出$[l, r]$的答案。（其余三种情况做法类似）

右端点从$r-1$变为r，此时多出来的答案为以r为右端点，且左端点不小于l的所有回文串。

算法三

以中点为思考方向的回文串题目我们一般使用Manacher算法解决，而以左/右端点为思考方向的话，这就是回文自动机的强项了。

算法三

以中点为思考方向的回文串题目我们一般使用Manacher算法解决，而以左/右端点为思考方向的话，这就是回文自动机的强项了。

我们通过增量法对母串建立回文自动机，同时记录每个字符S[i]在回文自动机上对应的节点Node[i]。

算法三

那么此时查询以r为右端点且左端点不小于l的回文串个数，就相当于在回文自动机的parent树上查询tree[r]的祖先中，对应回文串长度不超过$r-l+1$的节点个数。

算法三

那么此时查询以r为右端点且左端点不小于l的回文串个数，就相当于在回文自动机的parent树上查询tree[r]的祖先中，对应回文串长度不超过$r-l+1$的节点个数。

而这个问题我们通过倍增或者树链剖分可以轻松解决。（前者常数较大）

算法三

那么此时查询以r为右端点且左端点不小于l的回文串个数，就相当于在回文自动机的parent树上查询tree[r]的祖先中，对应回文串长度不超过$r-l+1$的节点个数。

而这个问题我们通过倍增或者树链剖分可以轻松解决。（前者常数较大）

时间复杂度：$O(m\sqrt{n}\log n)$

期望得分：45-75分

算法四

重新回归Manacher算法，以中点为思考方向。

算法四

重新回归Manacher算法，以中点为思考方向。

为了更加方便地处理长度为偶数的回文串，我们在原串的字符之间加入分隔符，得到一个长度为$2n+1$的新字符串。

算法四

重新回归Manacher算法，以中点为思考方向。

为了更加方便地处理长度为偶数的回文串，我们在原串的字符之间加入分隔符，得到一个长度为$2n+1$的新字符串。

对新字符串我们做一遍Manacher算法，求出数组f[k]表示以字符S[k]为中点的最长扩展长度。

算法四

重新回归Manacher算法，以中点为思考方向。

为了更加方便地处理长度为偶数的回文串，我们在原串的字符之间加入分隔符，得到一个长度为$2n+1$的新字符串。

对新字符串我们做一遍Manacher算法，求出数组f[k]表示以字符S[k]为中点的最长扩展长度。

例：abaa $\rightarrow$ |a|b|a|a|

此时f数组的值为：0,1,0,3,0,1,2,3,2

算法四

重新回归Manacher算法，以中点为思考方向。

为了更加方便地处理长度为偶数的回文串，我们在原串的字符之间加入分隔符，得到一个长度为$2n+1$的新字符串。

对新字符串我们做一遍Manacher算法，求出数组f[k]表示以字符S[k]为中点的最长扩展长度。

例：abaa $\rightarrow$ |a|b|a|a|

此时f数组的值为：0,1,0,3,0,1,2,3,2

算法四

经过观察我们可以得到，若S[k]为分隔符，则以S[k]为中点并且在原字符串中实际存在的回文串个数为$\frac{f[k]}{2}$，否则S[k]为字母，此时回文串个数为$\frac{f[k]+1}{2}$。

算法四

经过观察我们可以得到，若S[k]为分隔符，则以S[k]为中点并且在原字符串中实际存在的回文串个数为$\frac{f[k]}{2}$，否则S[k]为字母，此时回文串个数为$\frac{f[k]+1}{2}$。

对于原询问$[l, r]$，在新串中对应的新询问为$[2l-1, 2r+1]$，令$L = 2l-1, R = 2r+1$，考虑枚举回文串的中点，由于回文串的左/右端点不能超过L/R，我们可以得到

算法四

经过观察我们可以得到，若S[k]为分隔符，则以S[k]为中点并且在原字符串中实际存在的回文串个数为$\frac{f[k]}{2}$，否则S[k]为字母，此时回文串个数为$\frac{f[k]+1}{2}$。

对于原询问$[l,r]$，在新串中对应的新询问为$[2l-1,2r+1]$，令$L=2l-1,R=2r+1$，考虑枚举回文串的中点，由于回文串的左/右端点不能超过L/R，我们可以得到

$$Ans = \frac{1}{2} \cdot \left[(r-l+1) + \sum_{k=L}^{R} \min\{f[k], k-L, R-k\} \right]$$

算法四

当$k \leqslant \frac{L+R}{2} = l + r$时，$k - L \leqslant R - k$,

此时$\min\{f[k], k - L, R - k\} = \min\{f[k], k - L\}$

算法四

当$k \leqslant \frac{L+R}{2} = l + r$时，$k - L \leqslant R - k$,

此时$\min\{f[k], k - L, R - k\} = \min\{f[k], k - L\}$

那么现在我们的目标就是求出

$$\sum_{k=L}^{l+r} \min\{f[k], k - L\}$$

算法四

当$k \leqslant \frac{L+R}{2} = l + r$时，$k - L \leqslant R - k$,

此时$\min\{f[k], k - L, R - k\} = \min\{f[k], k - L\}$

那么现在我们的目标就是求出

$$\sum_{k=L}^{l+r} \min\{f[k], k - L\}$$

（当$k > \frac{L+R}{2} = l + r$时做法类似）

算法四

当$k \leqslant \frac{L+R}{2} = l + r$时，$k - L \leqslant R - k$,

此时$\min\{f[k], k - L, R - k\} = \min\{f[k], k - L\}$

那么现在我们的目标就是求出

$$\sum_{k=L}^{l+r} \min\{f[k], k - L\}$$

（当$k > \frac{L+R}{2} = l + r$时做法类似）

这是个经典问题，通过分类讨论去掉min符号后，问题转化为求二维平面矩形内点权和。

算法四

于是我们使用可持久化线段树来维护这个二维点集，便能做到单次询问$O(\log n)$了。

算法四

于是我们使用可持久化线段树来维护这个二维点集，便能做到单次询问$O(\log n)$了。

当然，由于这题可以离线，排序+扫描线的做法也是可以的（虽然有点麻烦= =）

算法四

于是我们使用可持久化线段树来维护这个二维点集，便能做到单次询问$O(\log n)$了。

当然，由于这题可以离线，排序+扫描线的做法也是可以的（虽然有点麻烦= =）

时间复杂度：$O((n+m)\log n)$

期望得分：100分

建造记者站(jhaha)

- 给出n个村庄在数轴上的坐标

建造记者站(jhaha)

- 给出n个村庄在数轴上的坐标
- 要求在这n个村庄内选择至多m个建造记者站，在村庄i建造记者站的代价为$C[i]$

建造记者站(jhaha)

- 给出n个村庄在数轴上的坐标
- 要求在这n个村庄内选择至多m个建造记者站，在村庄i建造记者站的代价为$C[i]$
- 对于村庄i，如果它左右R_i范围内都没有记者站，那么需要额外付出$P[i]$的代价

建造记者站(jhaha)

- 给出n个村庄在数轴上的坐标
- 要求在这n个村庄内选择至多m个建造记者站，在村庄i建造记者站的代价为$C[i]$
- 对于村庄i，如果它左右R_i范围内都没有记者站，那么需要额外付出$P[i]$的代价
- 求出最小总代价

建造记者站(jhaha)

- 给出n个村庄在数轴上的坐标
- 要求在这n个村庄内选择至多m个建造记者站，在村庄i建造记者站的代价为$C[i]$
- 对于村庄i，如果它左右R_i范围内都没有记者站，那么需要额外付出$P[i]$的代价
- 求出最小总代价
- 数据范围：$n \leqslant 20000, m \leqslant 100$

算法一

考虑动态规划，设$f[k][i]$表示在村庄$1\sim i$中共建造了k个记者站，且最后一个记者站在村庄i时的最小总花费，则

算法一

考虑动态规划，设$f[k][i]$表示在村庄$1\sim i$中共建造了k个记者站，且最后一个记者站在村庄i时的最小总花费，则

$$f[k][i]=C[i]+\min_{1\leqslant j<i} f[k-1][j]+cost(j,i)$$

算法一

考虑动态规划，设$f[k][i]$表示在村庄$1\sim i$中共建造了k个记者站，且最后一个记者站在村庄i时的最小总花费，则

$$f[k][i]=C[i]+\min_{1\leqslant j<i} f[k-1][j]+cost(j,i)$$

其中$cost(l,r)$表示记者站建在村庄l和村庄r，且这两个记者站之间没有其他记者站，此时村庄$l\sim r$所需要额外付出的代价。

算法一

对于$cost(j,i)$，我们直接暴力枚举$j\sim i$间的所有村庄，判断它是否能被记者站覆盖。

算法一

对于$cost(j,i)$，我们直接暴力枚举$j \sim i$间的所有村庄，判断它是否能被记者站覆盖。

时间复杂度：$O(n^3m)$

期望得分：10分

算法二

在算法一中求$cost(j,i)$很耗时，于是我们考虑先预处理出所有的$cost$值。

算法二

在算法一中求$cost(j,i)$很耗时，于是我们考虑先预处理出所有的$cost$值。

先固定左端点l，右端点r不断向右移动。如果某个村庄能被左端点覆盖，那么右端点怎么移动都没有影响，我们直接去掉这些村庄。

算法二

在算法一中求$cost(j,i)$很耗时，于是我们考虑先预处理出所有的$cost$值。

先固定左端点l，右端点r不断向右移动。如果某个村庄能被左端点覆盖，那么右端点怎么移动都没有影响，我们直接去掉这些村庄。

当右端点由r变为$r+1$时，一些村庄因为右端点的移动不再被覆盖了，同时新增了一个需要考虑的村庄$r+1$。

算法二

村庄i能被右边的记者站r覆盖，必须满足$D[i]+R[i]\geqslant D[r]$

算法二

村庄i能被右边的记者站r覆盖，必须满足$D[i]+R[i] \geqslant D[r]$

$D[r]$是单调递增的，于是我们以$D[i]+R[i]$为关键字建一个小根堆，每次判断堆顶村庄在此次移动后是否不再被覆盖。

算法二

村庄i能被右边的记者站r覆盖，必须满足$D[i] + R[i] \geqslant D[r]$

$D[r]$是单调递增的，于是我们以$D[i] + R[i]$为关键字建一个小根堆，每次判断堆顶村庄在此次移动后是否不再被覆盖。

如果未被覆盖，在堆中删去它，同时用该村庄的额外代价更新答案即可。

算法二

村庄i能被右边的记者站r覆盖，必须满足$D[i] + R[i] \geqslant D[r]$

$D[r]$是单调递增的，于是我们以$D[i] + R[i]$为关键字建一个小根堆，每次判断堆顶村庄在此次移动后是否不再被覆盖。

如果未被覆盖，在堆中删去它，同时用该村庄的额外代价更新答案即可。

时间复杂度：$O(n^2 \log n + n^2 m)$

期望得分：30分

一个直观的证明

考虑证明该DP的转移具有决策单调性。

一个直观的证明

考虑证明该DP的转移具有决策单调性。

对于当前状态$f[k][i]$，如果有两个决策满足$j_1 < j_2 < i$，且决策j_2比决策j_1优。

一个直观的证明

考虑证明该DP的转移具有决策单调性。

对于当前状态$f[k][i]$，如果有两个决策满足$j_1 < j_2 < i$，且决策j_2比决策j_1优。

那么对于任意$i' > i$，决策j_2都会比决策j_1优。因为后面的村庄离j_2更近一些，记者站建在j_2更有可能覆盖到后面的村庄。

更加严谨的证明

令$F = f[k], G = f[k-1]$，

设$F[r]$和$F[r+1]$的最优决策分别为l和l'，现在我们要证明的就是$l \leqslant l'$。

更加严谨的证明

令$F = f[k], G = f[k-1]$，

设$F[r]$和$F[r+1]$的最优决策分别为l和l'，现在我们要证明的就是$l \leqslant l'$。

考虑反证法，假设$l > l'$成立，根据最优决策的定义有：
$G[l] + cost(l, r) + C[r] \leqslant G[l'] + cost(l', r) + C[r]$

更加严谨的证明

令$F = f[k], G = f[k-1]$，

设$F[r]$和$F[r+1]$的最优决策分别为l和l'，现在我们要证明的就是$l \leqslant l'$。

考虑反证法，假设$l > l'$成立，根据最优决策的定义有：

$G[l] + cost(l, r) + C[r] \leqslant G[l'] + cost(l', r) + C[r]$

即$G[l] - G[l'] \leqslant cost(l', r) - cost(l, r)$

更加严谨的证明

令$F = f[k], G = f[k-1]$，

设$F[r]$和$F[r+1]$的最优决策分别为l和l'，现在我们要证明的就是$l \leqslant l'$。

考虑反证法，假设$l > l'$成立，根据最优决策的定义有：

$G[l] + cost(l,r) + C[r] \leqslant G[l'] + cost(l',r) + C[r]$

即$G[l] - G[l'] \leqslant cost(l',r) - cost(l,r)$

接下来我们就要证明l'不可能是$F[r+1]$的最优决策。

更加严谨的证明

不妨证明决策l比决策l'更优，也就是

$$G[l] + cost(l, r+1) + C[r+1] \leqslant G[l'] + cost(l', r+1) + C[r+1]$$

更加严谨的证明

不妨证明决策l比决策l'更优，也就是

$$G[l]+cost(l,r+1)+C[r+1]\leqslant G[l']+cost(l',r+1)+C[r+1]$$

即$G[l]-G[l']\leqslant cost(l',r+1)-cost(l,r+1)$

更加严谨的证明

不妨证明决策l比决策l'更优，也就是

$$G[l] + cost(l, r+1) + C[r+1] \leqslant G[l'] + cost(l', r+1) + C[r+1]$$

即$G[l] - G[l'] \leqslant cost(l', r+1) - cost(l, r+1)$

只需$cost(l', r) - cost(l, r) \leqslant cost(l', r+1) - cost(l, r+1)$

更加严谨的证明

不妨证明决策l比决策l'更优，也就是

$$G[l] + cost(l, r+1) + C[r+1] \leqslant G[l'] + cost(l', r+1) + C[r+1]$$

即$G[l] - G[l'] \leqslant cost(l', r+1) - cost(l, r+1)$

只需$cost(l', r) - cost(l, r) \leqslant cost(l', r+1) - cost(l, r+1)$

所以接下来我们需要证明

$$cost(l', r+1) - cost(l', r) \geqslant cost(l, r+1) - cost(l, r) \qquad (*)$$

更加严谨的证明

而由$cost$函数的定义我们可以知道：

更加严谨的证明

而由$cost$函数的定义我们可以知道：

$$cost(l', r+1) - cost(l', r) = \sum_{\substack{l' \leqslant k \leqslant r, \\ D[k]+R[k]=r}} \left[D[k] - R[k] > l' \right] \cdot P[k]$$

$$cost(l, r+1) - cost(l, r) = \sum_{\substack{l \leqslant k \leqslant r, \\ D[k]+R[k]=r}} \left[D[k] - R[k] > l \right] \cdot P[k]$$

更加严谨的证明

而由$cost$函数的定义我们可以知道：

$$cost(l', r+1) - cost(l', r) = \sum_{\substack{l' \leqslant k \leqslant r, \\ D[k]+R[k]=r}} [D[k] - R[k] > l'] \cdot P[k]$$

$$cost(l, r+1) - cost(l, r) = \sum_{\substack{l \leqslant k \leqslant r, \\ D[k]+R[k]=r}} [D[k] - R[k] > l] \cdot P[k]$$

其中符号$[cond]$表示当命题$cond$成立时返回1，否则返回0。

更加严谨的证明

因为$l' < l$，所以显然$[D[k] - R[k] > l'] \geqslant [D[k] - R[k] > l]$

更加严谨的证明

因为$l' < l$，所以显然$[D[k] - R[k] > l'] \geqslant [D[k] - R[k] > l]$

又因为$P[k] \geqslant 0$，故$(*)$式成立。

更加严谨的证明

因为$l' < l$，所以显然$[D[k] - R[k] > l'] \geqslant [D[k] - R[k] > l]$
又因为$P[k] \geqslant 0$，故$(*)$式成立。
那么决策l比决策l'更优，与l'是$G[r+1]$的最优决策矛盾。

更加严谨的证明

因为$l' < l$，所以显然$[D[k] - R[k] > l'] \geqslant [D[k] - R[k] > l]$

又因为$P[k] \geqslant 0$，故$(*)$式成立。

那么决策l比决策l'更优，与l'是$G[r+1]$的最优决策矛盾。

所以$l \leqslant l'$，DP转移的决策单调性得证。

算法三

证明了决策单调性，接下来考虑如何快速预处理出$cost$值。

算法三

证明了决策单调性，接下来考虑如何快速预处理出$cost$值。
设村庄i左边最远能覆盖它的记者站为$left[i]$。

算法三

证明了决策单调性，接下来考虑如何快速预处理出$cost$值。

设村庄i左边最远能覆盖它的记者站为$left[i]$。

将每个村庄按$D[i]+R[i]$从小到大排序放入一个队列，不断向右移动右端点r。

算法三

证明了决策单调性，接下来考虑如何快速预处理出$cost$值。

设村庄i左边最远能覆盖它的记者站为$left[i]$。

将每个村庄按$D[i]+R[i]$从小到大排序放入一个队列，不断向右移动右端点r。

如果这次移动导致一些村庄不再被右端点覆盖了，考虑它是否还能被左端点覆盖。

算法三

如果此次移动后村庄i未被覆盖，那么左端点小于$left[i]$的所有$cost$值都会增加$P[i]$，同时村庄i出队。

算法三

如果此次移动后村庄i未被覆盖，那么左端点小于$left[i]$的所有$cost$值都会增加$P[i]$，同时村庄i出队。

所以我们需要实现一个支持在上一个版本基础上进行区间加操作的数据结构。

算法三

如果此次移动后村庄i未被覆盖，那么左端点小于$left[i]$的所有$cost$值都会增加$P[i]$，同时村庄i出队。

所以我们需要实现一个支持在上一个版本基础上进行区间加操作的数据结构。

使用可持久化线段树即可。

算法三

如果此次移动后村庄i未被覆盖，那么左端点小于$left[i]$的所有$cost$值都会增加$P[i]$，同时村庄i出队。

所以我们需要实现一个支持在上一个版本基础上进行区间加操作的数据结构。

使用可持久化线段树即可。

每个村庄只会出队一次，所以总时间复杂度为$O(nlogn)$。

算法三

于是我们使用单调栈算法，每次转移在可持久化线段树中查询$cost$值，便能在$O(mn\log^2 n)$的时间内解决此题了。

算法三

于是我们使用单调栈算法，每次转移在可持久化线段树中查询$cost$值，便能在$O(mn\log^2 n)$的时间内解决此题了。

当然，由于这题F数组的转移与DP顺序无关，我们还可以使用分治算法。若在求mid的最优决策时，不是每次都重新对线段树的叶子节点进行访问，而是直接在线段树上找到对应决策区间后再递归到所有叶子节点，这样可以做到$O(mn\log n)$的复杂度。

算法三

于是我们使用单调栈算法，每次转移在可持久化线段树中查询$cost$值，便能在$O(mn\log^2 n)$的时间内解决此题了。

当然，由于这题F数组的转移与DP顺序无关，我们还可以使用分治算法。若在求mid的最优决策时，不是每次都重新对线段树的叶子节点进行访问，而是直接在线段树上找到对应决策区间后再递归到所有叶子节点，这样可以做到$O(mn\log n)$的复杂度。

期望得分：60-100分

算法三

于是我们使用单调栈算法，每次转移在可持久化线段树中查询$cost$值，便能在$O(mn\log^2 n)$的时间内解决此题了。

当然，由于这题F数组的转移与DP顺序无关，我们还可以使用分治算法。若在求mid的最优决策时，不是每次都重新对线段树的叶子节点进行访问，而是直接在线段树上找到对应决策区间后再递归到所有叶子节点，这样可以做到$O(mn\log n)$的复杂度。

期望得分：60-100分

~~真的有人写这个算法吗？~~

算法四

算法三纯属口胡，没听懂也没有关系，算法四似乎比算法三好想&好写多的样子. . .

算法四

算法三纯属口胡，没听懂也没有关系，算法四似乎比算法三好想&好写多的样子...

对于每一层的DP，我们从小到大枚举r，然后直接用线段树动态维护$f[k-1][r]+cost(l,r), 1 \leqslant l < r$的最小值。

算法四

算法三纯属口胡，没听懂也没有关系，算法四似乎比算法三好想&好写多的样子...

对于每一层的DP，我们从小到大枚举r，然后直接用线段树动态维护$f[k-1][r]+cost(l,r), 1 \leqslant l < r$的最小值。

与算法三$cost$值的预处理类似，我们将村庄按$D[i]+R[i]$为关键字排序。

算法四

算法三纯属口胡，没听懂也没有关系，算法四似乎比算法三好想&好写多的样子. . .

对于每一层的DP，我们从小到大枚举r，然后直接用线段树动态维护$f[k-1][r]+cost(l,r), 1 \leqslant l < r$的最小值。

与算法三$cost$值的预处理类似，我们将村庄按$D[i]+R[i]$为关键字排序。

当r向右移动变为$r+1$时，若村庄i不再被右端点覆盖，考虑它是否能被左端点覆盖。

算法四

算法三纯属口胡，没听懂也没有关系，算法四似乎比算法三好想&好写多的样子...

对于每一层的DP，我们从小到大枚举r，然后直接用线段树动态维护$f[k-1][r]+cost(l,r), 1 \leqslant l < r$的最小值。

与算法三$cost$值的预处理类似，我们将村庄按$D[i]+R[i]$为关键字排序。

当r向右移动变为$r+1$时，若村庄i不再被右端点覆盖，考虑它是否能被左端点覆盖。

于是村庄$1 \sim left[i]-1$在线段树里的值都会加上$P[i]$，也就是一个区间加操作，然后村庄i出队。

算法四

在每一层DP之前我们都清空线段树，每次转移的时候直接查询最小值即可。

算法四

在每一层DP之前我们都清空线段树，每次转移的时候直接查询最小值即可。

时间复杂度：$O(mn\log n)$

期望得分：100分

- Thanks!

- Thanks!
- 祝大家都能在NOI2016中取得好成绩!