Solution

ExfJoe

福建省长乐第一中学

March 2, 2017

Outline

冒泡排序

冒泡排序

- 小的数对大的数没有影响

冒泡排序

- 小的数对大的数没有影响
- 统计每个数前面有多少个数比自己大

冒泡排序

- 小的数对大的数没有影响
- 统计每个数前面有多少个数比自己大
- $$counter = \max\{i - A_i\}$$

冒泡排序

- 小的数对大的数没有影响
- 统计每个数前面有多少个数比自己大
-
$$counter = \max\{i - A_i\}$$
- 时空复杂度 $O(n)$

Outline

字符串匹配

字符串匹配

- $C \leq 40$：字符集很小，考虑对每种字符出现的位置进行哈希，判断匹配时将所有字符串哈希值排序进行一一比较

字符串匹配

- $C \leq 40$：字符集很小，考虑对每种字符出现的位置进行哈希，判断匹配时将所有字符串哈希值排序进行一一比较
- 考虑类似 KMP 一样进行匹配处理

字符串匹配

- $C \le 40$：字符集很小，考虑对每种字符出现的位置进行哈希，判断匹配时将所有字符串哈希值排序进行一一比较
- 考虑类似 KMP 一样进行匹配处理
- 普通的 KMP 匹配的要求是两个字符完全相同

字符串匹配

- $C \leq 40$：字符集很小，考虑对每种字符出现的位置进行哈希，判断匹配时将所有字符串哈希值排序进行一一比较
- 考虑类似 KMP 一样进行匹配处理
- 普通的 KMP 匹配的要求是两个字符完全相同
- 将要求改为两个字符上一次出现的相对位置相同 (即位置差值) 即可

字符串匹配

- $C \le 40$：字符集很小，考虑对每种字符出现的位置进行哈希，判断匹配时将所有字符串哈希值排序进行一一比较
- 考虑类似 KMP 一样进行匹配处理
- 普通的 KMP 匹配的要求是两个字符完全相同
- 将要求改为两个字符上一次出现的相对位置相同 (即位置差值) 即可
- 也可以用哈希实现

字符串匹配

- $C \leq 40$：字符集很小，考虑对每种字符出现的位置进行哈希，判断匹配时将所有字符串哈希值排序进行一一比较
- 考虑类似 KMP 一样进行匹配处理
- 普通的 KMP 匹配的要求是两个字符完全相同
- 将要求改为两个字符上一次出现的相对位置相同 (即位置差值) 即可
- 也可以用哈希实现
- 时空复杂度 $O(n)$

Outline

阅读

阅读

- 显然我们只关心那些有额外收益的点，为了方便，我们将起点终点当做收益为 0 的点

阅读

- 显然我们只关心那些有额外收益的点，为了方便，我们将起点终点当做收益为 0 的点
- 我们可以将路径用经过的有收益的点来表示

阅读

- 显然我们只关心那些有额外收益的点，为了方便，我们将起点终点当做收益为 0 的点
- 我们可以将路径用经过的有收益的点来表示
- 相邻两个经过的收益点间肯定是不停跳 D 来到达的

阅读

- 显然我们只关心那些有额外收益的点，为了方便，我们将起点终点当做收益为 0 的点
- 我们可以将路径用经过的有收益的点来表示
- 相邻两个经过的收益点间肯定是不停跳 D 来到达的
- $N \le 1000$：f_i 表示到达第 i 个收益点的最大愉悦度

阅读

- 显然我们只关心那些有额外收益的点，为了方便，我们将起点终点当做收益为 0 的点
- 我们可以将路径用经过的有收益的点来表示
- 相邻两个经过的收益点间肯定是不停跳 D 来到达的
- $N \leq 1000$：f_i 表示到达第 i 个收益点的最大愉悦度
-
$$F_i = \min_{0 \leq j < i} \{F_j - \lceil \frac{(T_i - T_j)}{D} \rceil * A\} + B_i$$

阅读

- 显然我们只关心那些有额外收益的点，为了方便，我们将起点终点当做收益为 0 的点
- 我们可以将路径用经过的有收益的点来表示
- 相邻两个经过的收益点间肯定是不停跳 D 来到达的
- $N \leq 1000$：f_i 表示到达第 i 个收益点的最大愉悦度
- $$F_i = \min_{0 \leq j < i} \{F_j - \lceil \frac{(T_i - T_j)}{D} \rceil * A\} + B_i$$
- 时间复杂度 $O(N^2)$

阅读

阅读

- $D \leq 100$：若有两个点到当前点的距离模 D 同余，则我们只需考虑后一个点

阅读

- $D \leq 100$：若有两个点到当前点的距离模 D 同余，则我们只需考虑后一个点
- 将点按 $T_i \bmod D$ 的值进行分类，这样 DP 的复杂度降为 $O(N \times D)$

阅读

- $D \le 100$：若有两个点到当前点的距离模 D 同余，则我们只需考虑后一个点
- 将点按 $T_i \bmod D$ 的值进行分类，这样 DP 的复杂度降为 $O(N \times D)$
- 将前两个 DP 方法结合起来

阅读

- $D \leq 100$：若有两个点到当前点的距离模 D 同余，则我们只需考虑后一个点
- 将点按 $T_i \bmod D$ 的值进行分类，这样 DP 的复杂度降为 $O(N \times D)$
- 将前两个 DP 方法结合起来
- 考虑从第 j 个点走到第 i 个点的代价 $\lceil \frac{(T_i - T_j)}{D} \rceil * A$

阅读

- $D \leq 100$：若有两个点到当前点的距离模 D 同余，则我们只需考虑后一个点
- 将点按 $T_i \bmod D$ 的值进行分类，这样 DP 的复杂度降为 $O(N \times D)$
- 将前两个 DP 方法结合起来
- 考虑从第 j 个点走到第 i 个点的代价 $\lceil \frac{(T_i - T_j)}{D} \rceil * A$
- 我们只关心 $\lceil \frac{(T_i - T_j)}{D} \rceil$，它与 $\lfloor \frac{T_i}{D} \rfloor - \lfloor \frac{T_j}{D} \rfloor$ 的值最多只会相差 1

阅读

- $D \leq 100$：若有两个点到当前点的距离模 D 同余，则我们只需考虑后一个点
- 将点按 $T_i \bmod D$ 的值进行分类，这样 DP 的复杂度降为 $O(N \times D)$
- 将前两个 DP 方法结合起来
- 考虑从第 j 个点走到第 i 个点的代价 $\lceil \frac{(T_i - T_j)}{D} \rceil * A$
- 我们只关心 $\lceil \frac{(T_i - T_j)}{D} \rceil$，它与 $\lfloor \frac{T_i}{D} \rfloor - \lfloor \frac{T_j}{D} \rfloor$ 的值最多只会相差 1
- 根据模 $T_i \bmod D$ 与 $T_j \bmod D$ 的值的大小我们可以判断出是否相差 1

阅读

- $D \leq 100$：若有两个点到当前点的距离模 D 同余，则我们只需考虑后一个点
- 将点按 $T_i \bmod D$ 的值进行分类，这样 DP 的复杂度降为 $O(N \times D)$
- 将前两个 DP 方法结合起来
- 考虑从第 j 个点走到第 i 个点的代价 $\lceil \frac{(T_i - T_j)}{D} \rceil * A$
- 我们只关心 $\lceil \frac{(T_i - T_j)}{D} \rceil$，它与 $\lfloor \frac{T_i}{D} \rfloor - \lfloor \frac{T_j}{D} \rfloor$ 的值最多只会相差 1
- 根据模 $T_i \bmod D$ 与 $T_j \bmod D$ 的值的大小我们可以判断出是否相差 1
- 路径上的 $\lfloor \frac{T_i}{D} \rfloor - \lfloor \frac{T_j}{D} \rfloor$ 相加最后等于 $\lfloor \frac{M}{D} \rfloor - \lfloor \frac{K}{D} \rfloor$

阅读

- $D \le 100$：若有两个点到当前点的距离模 D 同余，则我们只需考虑后一个点
- 将点按 $T_i \bmod D$ 的值进行分类，这样 DP 的复杂度降为 $O(N \times D)$
- 将前两个 DP 方法结合起来
- 考虑从第 j 个点走到第 i 个点的代价 $\lceil \frac{(T_i - T_j)}{D} \rceil * A$
- 我们只关心 $\lceil \frac{(T_i - T_j)}{D} \rceil$，它与 $\lfloor \frac{T_i}{D} \rfloor - \lfloor \frac{T_j}{D} \rfloor$ 的值最多只会相差 1
- 根据模 $T_i \bmod D$ 与 $T_j \bmod D$ 的值的大小我们可以判断出是否相差 1
- 路径上的 $\lfloor \frac{T_i}{D} \rfloor - \lfloor \frac{T_j}{D} \rfloor$ 相加最后等于 $\lfloor \frac{M}{D} \rfloor - \lfloor \frac{K}{D} \rfloor$
- 利用数据结构优化转移，时间复杂度 $O(n \log n)$