两道水题

ouuan

2019 年 3 月 22 日

皇后游戏/加工生产调度

给你两个数列长为 n 的数列 a 和 b，

$$c_i = \begin{cases} a_1 + b_1 & i = 1 \\ \max(c_{i-1}, \sum_{j=1}^{i} a_j) + b_i & 2 \le i \le n \end{cases}$$

现在你需要重新排列这两个数列（不改变 a_i 和 b_i 的对应关系），最小化 c_n。

皇后游戏/加工生产调度

给你两个数列长为 n 的数列 a 和 b，

$$c_i = \begin{cases} a_1 + b_1 & i = 1 \\ \max(c_{i-1}, \sum_{j=1}^{i} a_j) + b_i & 2 \le i \le n \end{cases}$$

现在你需要重新排列这两个数列（不改变 a_i 和 b_i 的对应关系），最小化 c_n。

- PJ组水题？

皇后游戏/加工生产调度

给你两个数列长为 n 的数列 a 和 b，

$$c_i = \begin{cases} a_1 + b_1 & i = 1 \\ \max(c_{i-1}, \sum_{j=1}^{i} a_j) + b_i & 2 \le i \le n \end{cases}$$

现在你需要重新排列这两个数列（不改变 a_i 和 b_i 的对应关系），最小化 c_n。

- PJ组水题？
- 很多人应该都A过这道题，但不一定真正理解了。

贪心

选取相邻的两个大臣 i 和 j （j 此时在 i 后一个），交换 i 和 j 对前面的大臣无影响，对后面的大臣的影响在于排在后面的那个大臣获得的奖金，需要使之尽量小。

设这两个大臣前面的所有大臣左手上的数之和为 sum ，这两个大臣的再往前一个大臣得到的奖金是 pre。

当 i 在前 j 在后时，这个值为 $\max(\max(\mathrm{pre}, \mathrm{sum} + a_i) + b_i, \mathrm{sum} + a_i + a_j) + b_j$。

当 j 在前 i 在后时，这个值为 $\max(\max(\mathrm{pre}, \mathrm{sum} + a_j) + b_j, \mathrm{sum} + a_j + a_i) + b_i$

把 $\max$ 外面的东西塞到里面去，推一推，就可以得到：

当 $\min(a_i, b_j) > \min(a_j, b_i)$ 时，需要交换 i 和 j。

一个经典错解

- 我会了!

```
sort(id+1,id+n+1,[](int x,int y){return min(a[i],b[
    j])<min(a[j],b[i]);});
```

（以 $P_{i,j} = \min(a_i, b_j) < \min(a_j, b_i)$ 为比较函数进行排序。）

hack数据

实际上，上面的做法是错误的，无法通过下面这组数据：

```
2
4
1 1
1 1
3 5
2 7
4
1 1
3 5
1 1
2 7
```

hack数据

这两组数据只有数列的初始顺序变了一下，但上面的代码输出为：

```
16
17
```

输出中间结果，可以发现，排列后的最终结果分别为：

```
1  1
1  1
2  7
3  5
```

和

```
1  1
3  5
1  1
2  7
```

hack数据

这两种排列方式都满足

$$\forall i \in [1, n), \min(a_i, b_{i+1}) \leq \min(a_{i+1}, b_i)$$

但第二种方式并不是最优解。

Strict Weak Ordering

在解释上面那种做法为什么会错之前，先来介绍一下 Strict Weak Ordering。

Strict Weak Ordering

在解释上面那种做法为什么会错之前，先来介绍一下 Strict Weak Ordering。

对于一个比较运算符（用“$<$”表示此运算符，用“$\not<$”表示不满足此运算符），若满足以下四个条件，则称其是满足 Strict Weak Ordering 的：

1. $x \not< x$ （非自反性）
2. 若 $x < y$，则 $y \not< x$ （非对称性）
3. 若 $x < y, y < z$，则 $x < z$ （传递性）
4. 若 $x \not< y, y \not< x, y \not< z, z \not< y$，则 $x \not< z, z \not< x$ （不可比性的传递性）

Strict Weak Ordering

在解释上面那种做法为什么会错之前，先来介绍一下 Strict Weak Ordering。

对于一个比较运算符（用“$<$”表示此运算符，用“$\not<$”表示不满足此运算符），若满足以下四个条件，则称其是满足 Strict Weak Ordering 的：

1. $x \not< x$（非自反性）
2. 若 $x < y$，则 $y \not< x$（非对称性）
3. 若 $x < y, y < z$，则 $x < z$（传递性）
4. 若 $x \not< y, y \not< x, y \not< z, z \not< y$，则 $x \not< z, z \not< x$（不可比性的传递性）

STL 中的比较函数（sort, priority_queue, set 等）必须满足 Strict Weak Ordering。

之前提到的错解，具有传递性而不具有不可比性的传递性。（某些题解中说这个做法错误的原因是不具有传递性，那是错误的。）

具有传递性的证明

数学证明比较繁琐，有兴趣的话可以到我博客去看..

（洛谷P2123我的题解里也给出了证明）

（大概长这样：）

满足传递性的证明

命题：$\forall \begin{cases} \min(a_i,b_j) < \min(a_j,b_i) \\ \min(a_j,b_k) < \min(a_k,b_j) \end{cases}$，有 $\min(a_i,b_k) < \min(a_k,b_i)$。

将上式拆解成逻辑式，即证：

$\forall \begin{cases} (a_i < a_j \vee b_j < a_j) \wedge (a_i < b_i \vee b_j < b_i) \\ (a_j < a_k \vee b_k < a_k) \wedge (a_j < b_j \vee b_k < b_j) \end{cases}$，有 $(a_i < a_k \vee b_k < a_k) \wedge (a_i < b_i \vee b_k < b_i)$。

假设原命题不成立，即 $\exists \begin{cases} (a_i < a_j \vee b_j < a_j) \wedge (a_i < b_i \vee b_j < b_i) & (1) \\ (a_j < a_k \vee b_k < a_k) \wedge (a_j < b_j \vee b_k < b_j) & (2) \\ (a_i \geq a_k \wedge b_k \geq a_k) \vee (a_i \geq b_i \wedge b_k \geq b_i) & (3) \end{cases}$

分别讨论 (3) 式成立的两种情况：

若 $a_i \geq a_k \wedge b_k \geq a_k$，由 (2) 式得 $a_j < a_k$，进而推出 $a_j < a_i$，再由 (1) 式得 $b_j < a_j$，再由 (2) 式得到 $b_k < b_j$，所以 $b_k < b_j < a_j < a_k$，与 $b_k \geq a_k$ 矛盾，不成立。

若 $a_i \geq b_i \wedge b_k \geq b_i$，与上面类似，由 (1) 式得 $b_j < b_i$，进而推出 $b_j < b_k$，再由 (2) 式得到 $a_j < b_j$，再由 (1) 式得到 $a_i < a_j$，所以 $a_i < a_j < b_j < b_i$，与 $a_i \geq b_i$ 矛盾，不成立。

综上所述，假设不成立。

所以，$P_{i,j} = \min(a_i,b_j) < \min(a_j,b_i)$ 具有传递性。

还有一种证明方式是枚举四个数之间的大小关系，也可以证明。

不具有不可比性传递性的反例

	a	b
i	3	5
j	1	1
k	2	7

$\begin{cases} \min(3,1) = \min(1,5) \\ \min(1,7) = \min(2,1) \end{cases}$，但 $\min(3,7) \neq \min(2,5)$。

这样的反例还有很多，所以，

$$P(i,j) = \min(a_i, b_j) < \min(a_j, b_i)$$

不具有不可比性的传递性。

为何会错

简单地说，不满足 Strict Weak Ordering 的排序方式不能作为 STL 的比较函数。

为何会错

简单地说，不满足 Strict Weak Ordering 的排序方式不能作为 STL 的比较函数。

究其原因，“不具有不可比性的传递性”意味着：将序列中若干对不可比的相邻元素互换后，可能会出现前面的元素“大于”后面的元素，从而使得原先的排列方式不是最优的。

为何会错

简单地说，不满足 Strict Weak Ordering 的排序方式不能作为 STL 的比较函数。

究其原因，“不具有不可比性的传递性”意味着：将序列中若干对不可比的相邻元素互换后，可能会出现前面的元素“大于”后面的元素，从而使得原先的排列方式不是最优的。

举个栗子，$P(x, y) = x + 1 < y$ 就是一个满足非自反性、非对称性和传递性，却不满足不可比性的传递性的偏序关系。

为何会错

简单地说，不满足 Strict Weak Ordering 的排序方式不能作为 STL 的比较函数。

究其原因，“不具有不可比性的传递性”意味着：将序列中若干对不可比的相邻元素互换后，可能会出现前面的元素“大于”后面的元素，从而使得原先的排列方式不是最优的。

举个栗子，$P(x, y) = x + 1 < y$ 就是一个满足非自反性、非对称性和传递性，却不满足不可比性的传递性的偏序关系。

在这个偏序关系下，3 2 1 是一个相邻两项均不可比的数列，而将前两项互换后得到 2 3 1，此时 $P(1, 3) = true$（出现了后一项“小于”前一项）。

为何会错

简单地说，不满足 Strict Weak Ordering 的排序方式不能作为 STL 的比较函数。

究其原因，“不具有不可比性的传递性”意味着：将序列中若干对不可比的相邻元素互换后，可能会出现前面的元素“大于”后面的元素，从而使得原先的排列方式不是最优的。

举个栗子，$P(x, y) = x + 1 < y$ 就是一个满足非自反性、非对称性和传递性，却不满足不可比性的传递性的偏序关系。

在这个偏序关系下，3 2 1 是一个相邻两项均不可比的数列，而将前两项互换后得到 2 3 1，此时 $P(1, 3) = true$（出现了后一项“小于”前一项）。

对于一个 Strong Weak Ordering，每相邻两项都有前一项“小于”后一项或相邻两项不可比（互不“小于”）即意味着排序的完成，而此时任意交换不可比的相邻两项数列仍应有序。不具有不可比性的传递性则无法保证这一点。

正确解法

我在网上看到的解法几乎都是分组排序..我认为那种做法很不直观，难以想到。

正确解法

我在网上看到的解法几乎都是分组排序..我认为那种做法很不直观，难以想到。

一个简单的贪心思路： a 的前缀和对答案是有影响的，所以应当将较小的 a 放在前面。

$$P(i,j)=\begin{cases} a_i < a_j & \min(a_i,b_j)=\min(a_j,b_i) \\ \min(a_i,b_j) < \min(a_j,b_i) & otherwise \end{cases}$$

正确解法

我在网上看到的解法几乎都是分组排序..我认为那种做法很不直观，难以想到。

一个简单的贪心思路： a 的前缀和对答案是有影响的，所以应当将较小的 a 放在前面。

$$P(i,j) = \begin{cases} a_i < a_j & \min(a_i, b_j) = \min(a_j, b_i) \\ \min(a_i, b_j) < \min(a_j, b_i) & otherwise \end{cases}$$

一个解法正确的充分条件是：

1. 当 $P(i,j) = true$, 有 $\min(a_i, b_j) \leq \min(a_j, b_i)$。
2. 满足 Strict Weak Ordering。

正确解法

我在网上看到的解法几乎都是分组排序..我认为那种做法很不直观，难以想到。

一个简单的贪心思路： a 的前缀和对答案是有影响的，所以应当将较小的 a 放在前面。

$$P(i,j)=\begin{cases} a_i < a_j & \min(a_i,b_j)=\min(a_j,b_i) \\ \min(a_i,b_j)<\min(a_j,b_i) & otherwise \end{cases}$$

一个解法正确的充分条件是：

1. 当 $P(i,j)=true$, 有 $\min(a_i,b_j)\le\min(a_j,b_i)$。
2. 满足 Strict Weak Ordering。

这个做法显然满足第一点，第二点可以通过枚举证明，具体可以看我博客里的代码（大约就是枚举 6 个数之间的大小关系）。

由乃与大母神原型和偶像崇拜

这是第二道题，~~对轻拍/大母神有兴趣的可以自行去看lxl写的题面~~。

一个数列，两种操作：

1. 修改一个位置的值。
2. 查询一个区间是否是连续段。

其中，连续段指这个区间包含从最小值到最大值的每个数恰好一次。即 $max - min = r - l$ 且无重复元素。

数列长度、操作个数 5×10^5，值域 10^9。

几种经典做法

- 我会维护平方和!

几种经典做法

- 我会维护平方和!

几种经典做法

- 我会维护平方和!

  平方和是不是有问题

  In P3792 由乃与大母神原型和偶像崇拜 @2018-07-30 17:35

- 我会维护立方和!

几种经典做法

- 我会维护平方和!

  平方和是不是有问题

  In P3792 由乃与大母神原型和偶像崇拜 @2018-07-30 17:35

- 我会维护立方和!

  立方和是不是有问题?

  In P3792 由乃与大母神原型和偶像崇拜 @2018-10-31 20:46

几种经典做法

- 我会维护平方和!

  平方和是不是有问题

  In P3792 由乃与大母神原型和偶像崇拜 @2018-07-30 17:35

- 我会维护立方和!

  立方和是不是有问题?

  In P3792 由乃与大母神原型和偶像崇拜 @2018-10-31 20:46

- 我会线段树+平衡树/set维护前驱后继!

几种经典做法

- 我会维护平方和!

  平方和是不是有问题

  In P3792 由乃与大母神原型和偶像崇拜 @2018-07-30 17:35

- 我会维护立方和!

  立方和是不是有问题?

  In P3792 由乃与大母神原型和偶像崇拜 @2018-10-31 20:46

- 我会线段树+平衡树/set维护前驱后继!

  作者: wyx150137 更新时间: 2017-06-12 22:26 在Ta的博客查看

  首先我先轻轻的斥责一下这题卡一个 $nlog(n)$ 算法常数的暴行.....

一种新的做法

把每个数映射到一个随机数上，然后维护异或和来判断是否是连续段。

一种新的做法

把每个数映射到一个随机数上，然后维护异或和来判断是否是连续段。

我们先不管值域 10^9，假设值域 10^6，我们就可以把 1 - 10^6 的每个数映射到一个随机数上（随机生成一个长为 10^6 的数列即可），然后预处理出随机数列前 i 项的异或和 $pre[i]$。

一种新的做法

把每个数映射到一个随机数上，然后维护异或和来判断是否是连续段。

我们先不管值域 10^9，假设值域 10^6，我们就可以把 1 - 10^6 的每个数映射到一个随机数上（随机生成一个长为 10^6 的数列即可），然后预处理出随机数列前 i 项的异或和 $pre[i]$。

接着，我们用两个树状数组，分别维护和和~~异或和~~原数列的加法和以及映射到的随机数的异或和。查询一个区间时，先根据加法和以及区间长度判断如果这是一个连续段它应该包含哪些数，根据预处理出的前缀异或和 pre 就可以得到如果它是一个连续段异或和应该是多少，和实际的异或和进行比对，就可以判断它是不是一个连续段。

一种新的做法

把每个数映射到一个随机数上，然后维护异或和来判断是否是连续段。

我们先不管值域 10^9，假设值域 10^6，我们就可以把 1 - 10^6 的每个数映射到一个随机数上（随机生成一个长为 10^6 的数列即可），然后预处理出随机数列前 i 项的异或和 $pre[i]$。

接着，我们用两个树状数组，分别维护和和~~异或和~~原数列的加法和以及映射到的随机数的异或和。查询一个区间时，先根据加法和以及区间长度判断如果这是一个连续段它应该包含哪些数，根据预处理出的前缀异或和 pre 就可以得到如果它是一个连续段异或和应该是多少，和实际的异或和进行比对，就可以判断它是不是一个连续段。

举个栗子，如果询问区间是 $[2,4]$，$[2,4]$ 这段区间的加法和是 15，那么如果这个区间是连续段，就应该包含 $4,5,6$，那么这个区间对应的随机数的异或和就应该是 $pre[6]\ xor\ pre[3]$，与实际的异或和比较就可以判断是否是连续段。

一种新的做法

把每个数映射到一个随机数上，然后维护异或和来判断是否是连续段。

我们先不管值域 10^9，假设值域 10^6，我们就可以把 1 - 10^6 的每个数映射到一个随机数上（随机生成一个长为 10^6 的数列即可），然后预处理出随机数列前 i 项的异或和 $pre[i]$。

接着，我们用两个树状数组，分别维护和和~~异或和~~原数列的加法和以及映射到的随机数的异或和。查询一个区间时，先根据加法和以及区间长度判断如果这是一个连续段它应该包含哪些数，根据预处理出的前缀异或和 pre 就可以得到如果它是一个连续段异或和应该是多少，和实际的异或和进行比对，就可以判断它是不是一个连续段。

举个栗子，如果询问区间是 $[2,4]$，$[2,4]$ 这段区间的加法和是 15，那么如果这个区间是连续段，就应该包含 $4,5,6$，那么这个区间对应的随机数的异或和就应该是 $pre[6]\ xor\ pre[3]$，与实际的异或和比较就可以判断是否是连续段。

这虽然是一个非确定性算法，但如果你映射到的随机数是 unsigned long long，几乎没法卡掉。当然还可以映射到多个随机数（类似双哈希）来进一步减少错误概率。

值域 10^9？

值域 10^9..不是离散化就好了吗？

值域 10^9？

值域 10^9..不是离散化就好了吗？

直接离散化：$2\ 3\ 6\ 7 \to 1\ 2\ 3\ 4$，值域不连续$\to$值域连续。

值域 10^9？

值域 10^9..不是离散化就好了吗？

直接离散化：$2\ 3\ 6\ 7 \to 1\ 2\ 3\ 4$，值域不连续$\to$值域连续。

一个简单的方法：离散化时把每个数加一也进行离散化。（把 $2, 3, 4, 6, 7, 8$ 都扔进离散化数组里。）

$2\ 3\ 6\ 7 \to 1\ 2\ 4\ 5$，解决!

讲完了

之前听大家的神（du）仙（liu）题选（qiangzhi）讲（diaoxian），感觉自己没什么题可以拿来讲，所以就选了两道水题，分享了一些与经典做法不同的做法，希望能对大家有所帮助。

感谢 YALI 能给我这样一次展（gong）示（kai）自（chu）己（xing）的机会。

感谢各位同学能听我讲完这样两道无聊的题目。

感谢 TeX 让我有了这样一个制作 slide 的平台。

~~我和几位神仙最近在准备一场CF，欢迎大家到时候来捧（nue）场！~~