FOI2022 算法夏令营上机练习（三）

试题讲评

Jul 18, 2022

Contents

章节目录

题目简述

给定平面上 n 个整点的坐标，请在直角坐标系上绘制出这些点。具体要求如下：

1. 要求绘制的点用 * 标出，其它点用 . 标出。

2. 对于没有要求绘制的点，x 轴上的点用 - 标出，y 轴上的点用 | 标出。特别地，原点用 + 标出。

3. 所绘制的坐标系的边缘上必须有要绘制的点或坐标轴。

算法一

算法流程

1. 确定边界：用所有要绘制的点的横坐标和 $x = 0$ 确定横坐标的上下限，用所有要绘制的点的纵坐标和 $y = 0$ 确定纵坐标的上下限。

算法一

算法流程

1. 确定边界：用所有要绘制的点的横坐标和 $x = 0$ 确定横坐标的上下限，用所有要绘制的点的纵坐标和 $y = 0$ 确定纵坐标的上下限。
2. 由于优先级满足. $<$ -,| $<$ + $<$ *，所以可以按照这个顺序不断覆盖（即先全部变成.，然后将坐标轴上的点变成 - 或 |，再用 + 标注原点，最后标出要绘制的点。

算法一

算法流程

1. 确定边界：用所有要绘制的点的横坐标和 $x = 0$ 确定横坐标的上下限，用所有要绘制的点的纵坐标和 $y = 0$ 确定纵坐标的上下限。
2. 由于优先级满足. < -,| < + < *，所以可以按照这个顺序不断覆盖（即先全部变成.，然后将坐标轴上的点变成 - 或 |，再用 + 标注原点，最后标出要绘制的点。

实现技巧

1. 对于绘制图形（特别是多个图层）的题目，建议先在一个字符数组中完成图形的编辑（因为这样可以方便地完成修改），然后再一次性输出。

算法一

算法流程

1. 确定边界：用所有要绘制的点的横坐标和 $x = 0$ 确定横坐标的上下限，用所有要绘制的点的纵坐标和 $y = 0$ 确定纵坐标的上下限。
2. 由于优先级满足. < -,| < + < *，所以可以按照这个顺序不断覆盖（即先全部变成.，然后将坐标轴上的点变成 - 或 |，再用 + 标注原点，最后标出要绘制的点。

实现技巧

1. 对于绘制图形（特别是多个图层）的题目，建议先在一个字符数组中完成图形的编辑（因为这样可以方便地完成修改），然后再一次性输出。
2. 对于坐标为负数的情况，可以将所有的点的坐标全部加上一个数，使得它们都大于等于 0，这样就可以直接对应到数组中的坐标了。

算法一

算法流程

1. 确定边界：用所有要绘制的点的横坐标和 $x = 0$ 确定横坐标的上下限，用所有要绘制的点的纵坐标和 $y = 0$ 确定纵坐标的上下限。
2. 由于优先级满足. < -,| < + < *，所以可以按照这个顺序不断覆盖（即先全部变成.，然后将坐标轴上的点变成 - 或 |，再用 + 标注原点，最后标出要绘制的点。

实现技巧

1. 对于绘制图形（特别是多个图层）的题目，建议先在一个字符数组中完成图形的编辑（因为这样可以方便地完成修改），然后再一次性输出。
2. 对于坐标为负数的情况，可以将所有的点的坐标全部加上一个数，使得它们都大于等于 0，这样就可以直接对应到数组中的坐标了。**注意**：此时原点和坐标轴的位置也要相应移动。

章节目录

题目简述

有一棵二叉树，每条边有边权，第一次到达一个点的时候可以获得 a_i 的收益。要求你选择一条从点 1 出发并最终回到点 1 的路径，满足边权之和不超过 w，求这条路径的最大收益。

算法一

数据范围：$n \leq 10, w \leq 20$。

算法一

数据范围：$n \leq 10, w \leq 20$。
由于路线从点 1 出发并最终回到点 1，所以每一条边一旦经过就一定会经过恰好 2 次。

算法一

数据范围：$n \le 10, w \le 20$。

由于路线从点 1 出发并最终回到点 1，所以每一条边一旦经过就一定会经过恰好 2 次。

根据这个性质，我们可以对整棵树进行搜索，令 $dfs(u, w)$ 表示到点 u 后还剩 w 元时的最优方案，此时有三种情况：

算法一

数据范围： $n \le 10, w \le 20$。

由于路线从点 1 出发并最终回到点 1，所以每一条边一旦经过就一定会经过恰好 2 次。

根据这个性质，我们可以对整棵树进行搜索，令 $dfs(u, w)$ 表示到点 u 后还剩 w 元时的最优方案，此时有三种情况：

1. 不再走子树，则收益为 a_u。

算法一

数据范围：$n \le 10, w \le 20$。

由于路线从点 1 出发并最终回到点 1，所以每一条边一旦经过就一定会经过恰好 2 次。

根据这个性质，我们可以对整棵树进行搜索，令 $dfs(u, w)$ 表示到点 u 后还剩 w 元时的最优方案，此时有三种情况：

1. 不再走子树，则收益为 a_u。
2. 只去一个子树，且去这个子树的边权为 w_1，那么收益为 $a_u + dfs(v_1, w - 2w_1)$。

算法一

数据范围：$n \le 10, w \le 20$。

由于路线从点 1 出发并最终回到点 1，所以每一条边一旦经过就一定会经过恰好 2 次。

根据这个性质，我们可以对整棵树进行搜索，令 $dfs(u, w)$ 表示到点 u 后还剩 w 元时的最优方案，此时有三种情况：

1. 不再走子树，则收益为 a_u。
2. 只去一个子树，且去这个子树的边权为 w_1，那么收益为 $a_u + dfs(v_1, w - 2w_1)$。
3. 两个子树都去，则收益为 $a_u + \max_j\{dfs(v_1, j - 2w_1) + dfs(v_2, w - j - 2w_2)\}$。

算法二

数据范围：$n \le 50, w \le 100$。

算法二

数据范围：$n \le 50, w \le 100$。

注意到 $dfs(u, w)$ 的值和这个函数再程序中被调用的位置无关，所以可以加上记忆化搜索或直接使用树形 DP 即可通过本题。

算法二

数据范围：$n \le 50, w \le 100$。

注意到 $dfs(u, w)$ 的值和这个函数再程序中被调用的位置无关，所以可以加上记忆化搜索或直接使用树形 DP 即可通过本题。

时间复杂度 $\mathbb{O}(nw)$。

章节目录

题目简述

有一个 n 位的正整数，要求从中删去 k 个数位，求剩下的数字拼成的数的最大值。

算法一

数据范围：$n \le 10$。

算法一

数据范围：$n \leq 10$。
枚举每一位数字是否擦除，然后将剩下的数字拼成一个数后比较得出答案。

算法一

数据范围：$n \leq 10$。
枚举每一位数字是否擦除，然后将剩下的数字拼成一个数后比较得出答案。
时间复杂度 $\mathbb{O}(2^n)$。

算法二

数据范围：$n \le 1\ 000$。

算法二

数据范围：$n \le 1\ 000$。
由于最后剩下的数中越靠前的数字对于整个数的大小影响越大，所以可以倒着 DP。

算法二

数据范围：$n \leq 1\ 000$。

由于最后剩下的数中越靠前的数字对于整个数的大小影响越大，所以可以倒着 DP。

记 $f[i][j]$ 表示在后 i 位中删掉 j 位后剩下的数字组成的最大的数的第一位在原数中的位置，$val(i,j)$ 表示这个最大的数的值。

算法二

数据范围：$n \leq 1\ 000$。

由于最后剩下的数中越靠前的数字对于整个数的大小影响越大，所以可以倒着 DP。

记 $f[i][j]$ 表示在后 i 位中删掉 j 位后剩下的数字组成的最大的数的第一位在原数中的位置，$val(i,j)$ 表示这个最大的数的值。

那么有：

$$f[i][j] = \begin{cases} f[i+1][j], val(i+1,j) > \overline{a[i](val(i+1,j-1))} \\ i, val(i+1,j) \leq \overline{a[i](val(i+1,j-1))} \end{cases} \tag{1}$$

算法二

数据范围：$n \le 1\ 000$。

由于最后剩下的数中越靠前的数字对于整个数的大小影响越大，所以可以倒着 DP。

记 $f[i][j]$ 表示在后 i 位中删掉 j 位后剩下的数字组成的最大的数的第一位在原数中的位置，$val(i,j)$ 表示这个最大的数的值。

那么有：

$$f[i][j] = \begin{cases} f[i+1][j], val(i+1,j) > \overline{a[i](val(i+1,j-1))} \\ i, val(i+1,j) \le \overline{a[i](val(i+1,j-1)} \end{cases} \tag{1}$$

最后根据 f 的值一步步往后跳即可恢复出整个数。

算法二

数据范围：$n \le 1\ 000$。

由于最后剩下的数中越靠前的数字对于整个数的大小影响越大，所以可以倒着 DP。

记 $f[i][j]$ 表示在后 i 位中删掉 j 位后剩下的数字组成的最大的数的第一位在原数中的位置，$val(i,j)$ 表示这个最大的数的值。

那么有：

$$f[i][j] = \begin{cases} f[i+1][j], val(i+1,j) > \overline{a[i](val(i+1,j-1))} \\ i, val(i+1,j) \le \overline{a[i](val(i+1,j-1)} \end{cases} \quad (1)$$

最后根据 f 的值一步步往后跳即可恢复出整个数。

时间复杂度 $\mathbb{O}(n^2)$。

算法三

数据范围：$n \leq 10^6$。

算法三

数据范围：$n \le 10^6$。

如果不考虑 k 的限制，单从字典序的角度来讲，最优的策略是先选出所有的 9，然后选出最后一个 9 之后的所有的 8，再选最后 8 之后的所有的 7，以此类推。换句话说，我们想让这个序列变成一个单调递减序列。

算法三

数据范围：$n \le 10^6$。

如果不考虑 k 的限制，单从字典序的角度来讲，最优的策略是先选出所有的 9，然后选出最后一个 9 之后的所有的 8，再选最后 8 之后的所有的 7，以此类推。换句话说，我们想让这个序列变成一个单调递减序列。

如果加上了 k 的限制，那么在删数的过程中可能无法将所有我们想删除的数都删掉。不过由于前面数位的重要性高于后面的数位的重要性，所以此时的最优策略为先用单调栈让前面的数列递减，在删满 k 个数之后将所有的剩下的数保留。

算法三

数据范围：$n \le 10^6$。

如果不考虑 k 的限制，单从字典序的角度来讲，最优的策略是先选出所有的 9，然后选出最后一个 9 之后的所有的 8，再选最后 8 之后的所有的 7，以此类推。换句话说，我们想让这个序列变成一个单调递减序列。

如果加上了 k 的限制，那么在删数的过程中可能无法将所有我们想删除的数都删掉。不过由于前面数位的重要性高于后面的数位的重要性，所以此时的最优策略为先用单调栈让前面的数列递减，在删满 k 个数之后将所有的剩下的数保留。

时间复杂度 $\mathbb{O}(n)$。

章节目录

题目简述

一个 01 数列由 x_1 个 1、x_2 个 0、x_3 个 1、x_4 个 0、x_5 个 1 构成，有若干种可选的操作，每个操作形如用 $r-l+1$ 的时间将 $[l,r]$ 的所有数取反，求最小的操作时间使得所有数都变成 1。

算法一

数据范围：$m \le 10$。

算法一

数据范围：$m \leq 10$。

由于每个操作最多执行一次（执行两次等于没有执行），所以我们可以枚举每一个操作是否执行了，并在所有满足条件的操作组合种选出一个花费时间最少的即可。

算法一

数据范围：$m \le 10$。

由于每个操作最多执行一次（执行两次等于没有执行），所以我们可以枚举每一个操作是否执行了，并在所有满足条件的操作组合种选出一个花费时间最少的即可。

时间复杂度 $\mathbb{O}(2^m \times mn)$。

算法二

数据范围：$1 \le x_1, x_2, x_3, x_4, x_5, m \le 10^5$。

算法二

数据范围：$1 \le x_1, x_2, x_3, x_4, x_5, m \le 10^5$。

在正常情况下，如果要将某一个 01 序列中的某一段数取反，可以考虑使用线段树等数据结构解决，但是本题中操作的随意性很大（不确定每个操作是否执行），所以我们需要找到取反操作的一种更加直观的表达形式，为之后套用其它算法留下空间。

算法二

数据范围：$1 \le x_1, x_2, x_3, x_4, x_5, m \le 10^5$。

在正常情况下，如果要将某一个 01 序列中的某一段数取反，可以考虑使用线段树等数据结构解决，但是本题中操作的随意性很大（不确定每个操作是否执行），所以我们需要找到取反操作的一种更加直观的表达形式，为之后套用其它算法留下空间。

考虑类比差分，对于原来的 01 序列 c_i，若第 i 位与上一位不同则 $d_i = 1$，否则 $d_i = 0$。初始时有 $1, x_1 + 1, x_1 + x_2 + 1, x_1 + x_2 + x_3 + 1, x_1 + x_2 + x_3 + x_4 + 1$ 这几个位置上的 $d = 1$，而末状态除了 $d_1 = 1$ 外，其它所有的 $d = 0$。

算法二

数据范围：$1 \le x_1, x_2, x_3, x_4, x_5, m \le 10^5$。

在正常情况下，如果要将某一个 01 序列中的某一段数取反，可以考虑使用线段树等数据结构解决，但是本题中操作的随意性很大（不确定每个操作是否执行），所以我们需要找到取反操作的一种更加直观的表达形式，为之后套用其它算法留下空间。

考虑类比差分，对于原来的 01 序列 c_i，若第 i 位与上一位不同则 $d_i = 1$，否则 $d_i = 0$。初始时有 $1, x_1 + 1, x_1 + x_2 + 1, x_1 + x_2 + x_3 + 1, x_1 + x_2 + x_3 + x_4 + 1$ 这几个位置上的 $d = 1$，而末状态除了 $d_1 = 1$ 外，其它所有的 $d = 0$。

对于原序列 $[l, r]$ 区间的取反操作，不难发现它等价于对 d 的 l 和 $r + 1$ 两个位置取反。

算法二

根据上面的分析，除了 d 的 $x_1+1, x_1+x_2+1, x_1+x_2+x_3+1, x_1+x_2+x_3+x_4+1$ 这几个位置需要被取反奇数次以外，其它位置都要被取反偶数次。

算法二

根据上面的分析，除了 d 的 $x_1+1, x_1+x_2+1, x_1+x_2+x_3+1, x_1+x_2+x_3+x_4+1$ 这几个位置需要被取反奇数次以外，其它位置都要被取反偶数次。

于是可以类比得出下面这种图论模型：

对于 d 的每一个位置建一个点，如果存在一种操作可以对 $[l,r]$ 取反，那么就将 l 与 $r+1$ 练边权为 $r-l$ 的边。

算法二

根据上面的分析，除了 d 的 $x_1+1, x_1+x_2+1, x_1+x_2+x_3+1, x_1+x_2+x_3+x_4+1$ 这几个位置需要被取反奇数次以外，其它位置都要被取反偶数次。

于是可以类比得出下面这种图论模型：

对于 d 的每一个位置建一个点，如果存在一种操作可以对 $[l,r]$ 取反，那么就将 l 与 $r+1$ 练边权为 $r-l$ 的边。

此时如果我们用最短的时间让两个位于位置 s 和 t 的 1 消失，只需要求出从 s 到 t 的最短路即可。

算法二

根据上面的分析，除了 d 的 $x_1+1, x_1+x_2+1, x_1+x_2+x_3+1, x_1+x_2+x_3+x_4+1$ 这几个位置需要被取反奇数次以外，其它位置都要被取反偶数次。

于是可以类比得出下面这种图论模型：

对于 d 的每一个位置建一个点，如果存在一种操作可以对 $[l,r]$ 取反，那么就将 l 与 $r+1$ 练边权为 $r-l$ 的边。

此时如果我们用最短的时间让两个位于位置 s 和 t 的 1 消失，只需要求出从 s 到 t 的最短路即可。

于是我们在构建出图论模型后，枚举 $x_1+1, x_1+x_2+1, x_1+x_2+x_3+1,$ $x_1+x_2+x_3+x_4+1$ 的两两配对情况并分别计算最短路即可。

刺激的开分环节