2025/11/29 22:26 1/2 KD Tree

KD Tree

算法简介

一种特殊的二叉树,主要用于多维空间关键数据的搜索。

空间复杂度 \$O(n)\$□单次插入时间复杂度 \$O(\log n)\$□查询时间复杂度 \$O\left(k\sqrt[1-\frac 1k]n\right)\$□ 其中 \$k\$ 表示空间维数。

算法实现

为方便理解,这里仅讲解 2D_Tree 高维 KD_Tree 可以类推。实际上高维 KD_Tree 时间复杂度难以承受,算法竞赛中通常只涉及 2D Tree

先考虑建树过程。

二维空间的点无法直接比较大小,但如果将某个维度作为主要关键字,另一个维度作为次要关键字就可以 使得比较大小成为可能。

每层都选用一个维度,对结点进行排序,取中间结点的该维度数值作为分割面,将该结点左边结点加入左子树,该结点右边结点加入右子树。

\$\text{algorithm}\$ 库里有个叫 \$\text{nth_element}\$的神奇函数,可以 \$O(n)\$完成上述操作。

不断重复上述过程,便可以完成建树,而且可以使得该树高度平衡,时间复杂度 \$O(n\log n)\$□

建树过程实际上将整个二维空间分割成了若干部分。为了方便后面查询操作的剪枝,需要维护每个结点的子树的最小覆盖矩阵。

为使空间分割尽量均匀,需要选择合适的关键字。

比较优秀的关键字选择方法为求每个维度方差,选取方差大的维度作为主要关键字。

然而上述方法代码复杂,算法竞赛一般考虑轮换的方法选取主要关键字。

接下来是插入操作,插入操作会破坏原本树的平衡性,这个问题可以用替罪羊树的重构解决。

最后是查询操作,查询其实就是个暴力查询,但可以利用最小覆盖矩阵的剪枝控制时间复杂度,详细见例题。

代码模板

洛谷p4169

模板题

题意

二维空间,一开始 \$n\$ 个点□ \$m\$ 个操作。

Last

 $\begin{array}{l} \text{upgate:} \\ 2020/06/24 \end{array} \\ 2020-2021: \text{teams:legal_string:jxm2001:kd_tree https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal_string:jxm2001:kd_tree\&rev=1592969522\\ 11:32 \end{array}$

操作1:加入点 \$(x,y)\$[

操作2:询问当前点集中到给定点 \$(x,y)\$ 的最小哈密顿距离。

题解

建树、插入操作不再赘述。关于查询操作,将当前查询结果 \$\text{ans}\$ 设为全局变量,从根结点开始 遍历 KD_Tree_|假设当前访问结点为 \$\text{pos}\$|

首先用 \$\text{pos}\$ 到给定点 \$(x,y)\$ 的哈密顿距离更新 \$\text{ans}\$\|

计算给定点 \$(x,y)\$ 到 $\$\text{text}\{pos\}\$$ 的两个儿子结点的最小覆盖矩阵的最小哈密顿距离,记为 \$d 1\$ \parallel \$d 2\$ \parallel

优先遍历 \$d_i\$ 较小的结点。若 \$d_i\gt ans\$□立刻剪枝。

算法	习	颕
7 T / C	- J	NEX.

习题一

洛谷p6514

习题二

洛谷p3810

From:

https://wiki.cvbbacm.com/ - CVBB ACM Team

Permanent link:

https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal_string:jxm2001:kd_tree&rev=1592969522

Last update: 2020/06/24 11:32



https://wiki.cvbbacm.com/ Printed on 2025/11/29 22:26