**BÁO CÁO ĐỒ ÁN MÔN HỌC**

**(Đồ án tìm hiểu CTDL/Giải thuật)**

**Lớp: IT003.O21.CNTT**

**SINH VIÊN THỰC HIỆN**

**Mã sinh viên:23520717 Họ và tên: Hoàng Ngọc Khánh**

**TÊN ĐỀ TÀI:**

**CẤU TRÚC DỮ LIỆU VÀ THUẬT TOÁN KD-TREE**

CÁC NỘI DUNG CẦN BÁO CÁO: kết quả tìm hiểu được về CTDL/GT đã chọn

1. **Giới thiệu đồ án:**
   1. **Lý do chọn đề tài**

Nó phổ biến và hữu ích trong lĩnh vực tìm kiếm không gian và truy vấn dữ liệu không gian k chiều. Nó phù hợp với nhiều bài toán. Thú vị để nghiên cứu và có nhiều khía cạnh để khám phá bao gồm các phương pháp xây dựng cây, cách thức tìm kiếm hiệu quả, ảnh hưởng của dữ liệu đối với hiệu suất của cây, và nhiều hơn nữa. Nó cũng có thể giúp tôi biết thêm sâu rộng về thuật toán, cấu trúc dữ liệu và phân tích không gian dữ liệu.

* 1. **Giới thiệu**
* Trong khoa học máy tính và hình học tính toán, KD -Trees đã trở thành một cấu trúc dữ liệu phổ biến được sử dụng để tổ chức các điểm trong không gian K chiều, trong đó K thường là một số rất lớn.
  1. **Mục tiêu chính**
* **Hiểu được cơ bản và sâu về cấu trúc dữ liệu KD-tree.**
* **Biết cách cài đặt cơ bản KD-tree.**
* **Biết mức độ quan trọng và ứng dụng của nó vào thị trường hiện nay.**
* **Điều này là do các cấu trúc này cho phép tìm kiếm rất hiệu quả các điểm trong không gian đa chiều, bao gồm các tìm kiếm lân cận gần nhất và tìm kiếm phạm vi.**
  1. **Ứng dụng phổ biến**
* Nêu các ứng dụng phổ biến của KD-tree trong lĩnh vực như computer graphics, computational geometry, machine learning algorithms và bất

kỳ lĩnh vực nào cần tìm kiếm điểm gần nhất.

* Ứng dụng tìm kiếm nhanh.
* Phân cụm.
* Xử lý ảnh, video.
  1. **Tổ chức bài viết**
* Đưa ra được cái nhìn tổng quan về KD-tree.
* Trình bày lý thuyết cơ bản.
* Đi sâu vào thuật toán và phương pháp cụ thể.
* Kết luận.

1. **Quá trình thực hiện:**
   1. **Tuần 1: Tìm hiểu lý thuyết**

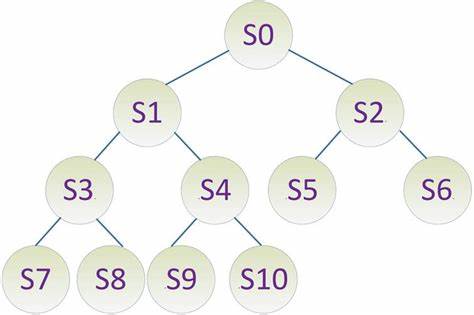
* Tìm hiểu khái niệm.
* Cách hoạt động của thuật toán KD-TREE.
* Phân tích KD-TREE.
* Mô tả KD-TREE tích hợp vào các ứng dụng cụ thể và lợi ích của nó mang lại.
* Trường hợp phổ biến nhất tìm kiếm gần nhất.
* Các biến thể và tối ưu hóa.

Link minh chứng : [IT003\_DOAN\_KD-TREE/kd-tree.docx at main · Khanh0110/IT003\_DOAN\_KD-TREE (github.com)](https://github.com/Khanh0110/IT003_DOAN_KD-TREE/blob/main/kd-tree.docx)

* 1. **Tuần 2: Cài đặt thuật toán**
* Triển khai thuật toán KD-TREE ứng dụng vào tìm kiếm gần nhất.
* Thêm bớt điều chỉnh lại cho phù hợp.
* Chốt ý, hoàn thiện lại mẫu báo cáo.

1. **Kết quả đạt được** 
   1. **Các định nghĩa/khái niệm cơ bản**

* KD-tree là một cấu trúc dữ liệu phân vùng không gian để tổ chức các điểm trong không gian k chiều.
* KD-tree là một trường hợp đặc biệt của cây phân vùng không gian nhị phân.
* Là một cây nhị phân trong đó mỗi nút là một điểm k chiều. Mỗi nút không phải lá có thể được coi là ngầm tạo ra một siêu mặt phẳng tách chia không gian thành hai phần, được gọi là nửa không gian. Các điểm ở bên trái của siêu mặt phẳng này được biểu diễn bằng cây con bên trái của nút đó và các điểm ở bên phải của siêu mặt phẳng được biểu thị bằng cây con bên phải.



* 1. **Các đặc trưng nổi bậc**
  2. **Thời gian xây dựng** 
     + Một số yếu tố chính ảnh hưởng đến thời gian xây dựng
* Số lượng điểm dữ liệu(n): Thời gian xây dựng KD-tree tăng tuyến tính với số lượng điểm dữ liệu.
* Số chiều của không gian dữ liệu(k): phụ thuộc vào số chiều của không gian dữ liệu. Tròn không gian có nhiều chiều, quá trình phân chia không gian trở nên phức tạp hơn, do đó thời gian xây dựng KD-tree có thể tăng lên.
* Cách chọn trục phân chia: Nếu không chọn trục phân chia một cách cân nhắc, có thể dẫn đến cây cân bằng và tăng thời gian xây dựng.
  1. **Thời gian tìm kiếm**
* Cấu trúc của cây: Nếu cây KD-tree có cấu trúc cân bằng, thì thời gina tìm kiếm sẽ được giảm đáng kể. Có thể thời gian tìm kiếm là O(log n)
* **Số lượng điểm dữ liệu (n)**: Thời gian tìm kiếm tăng tuyến tính với số lượng điểm dữ liệu. Tuy nhiên, với cây KD-tree, thời gian tìm kiếm thường được giảm đáng kể so với tìm kiếm tuyến tính.
* **Số chiều của không gian dữ liệu (k):** Thời gian tìm kiếm trong KD-tree phụ thuộc vào số chiều của không gian dữ liệu. Trong không gian có nhiều chiều, việc tìm kiếm có thể trở nên phức tạp hơn do hiệu quả giảm.
  1. **Độ phân chia của không gian**

\* thường được chia theo 3 cách sau:

**- Phương sai lớn nhất (Max Variance)**: Chọn trục phân chia là trục có phương sai lớn nhất trên tập dữ liệu.

**- Trục xen kẽ (Alternating Axes)**: Lần lượt lựa chọn các trục để phân chia không gian xen kẽ (ví dụ: lần lượt sử dụng trục x và y trong không gian hai chiều).

**- Trục ngẫu nhiên (Random Axes)**: Lựa chọn trục phân chia ngẫu nhiên từ giữa tất cả các trục có sẵn.

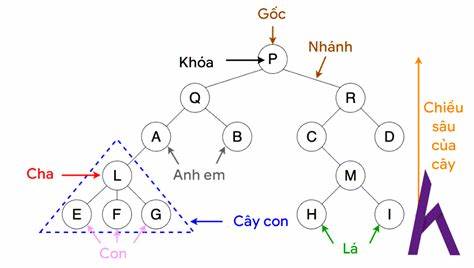
* 1. **Độ sâu của cây**
* Từ nút gốc đến các nút lá.
* Nó thể hiện mức độ phân cấp của cây và ảnh hưởng đến hiệu suất của việc tìm kiếm và lưu trữ dữ liệu.
  1. **Ưu điểm và nhược điểm:**

**Ưu điểm:**

* **Tìm kiếm nhanh chóng**: Trong trường hợp tốt nhất, thời gian tìm kiếm trong kd-tree có thể là O(log n), nơi n là số lượng điểm trong cây. Điều này là do cây kd được cắt ngang theo mỗi chiều để tạo ra các nút lá, giảm điểm dữ liệu cần phải duyệt qua.
* **Hiệu suất tốt trong không gian nhiều chiều**: Kd-tree hiệu quả trong việc tìm kiếm và truy vấn dữ liệu trong không gian nhiều chiều, nơi các phép toán tìm kiếm có thể trở nên khó khăn với các phương pháp tìm kiếm tuyến tính.
* **Tiết kiệm không gian**: Kd-tree tiết kiệm không gian so với các cấu trúc dữ liệu khác như cây quan hệ (quadtree, octree) vì nó chỉ lưu trữ điểm trên các nút lá.
* **Dễ dàng cài đặt**: Kd-tree là một cấu trúc dữ liệu đơn giản để triển khai và cài đặt.

**Nhược điểm:**

* **Phụ thuộc vào phân chia dữ liệu**: Hiệu suất của kd-tree có thể bị ảnh hưởng bởi cách chọn các trục chia và điểm chia. Trong trường hợp xấu nhất, thời gian tìm kiếm có thể trở thành O(n), khi cây không cân bằng hoặc dữ liệu phân phối không đồng đều.
* **Cập nhật chậm hơn**: Thêm hoặc xóa các điểm dữ liệu từ kd-tree có thể đòi hỏi việc cập nhật lại cấu trúc cây, điều này có thể làm tăng thời gian cần thiết so với cấu trúc dữ liệu tìm kiếm không gian không cần cập nhật lại cây.
* **Không hiệu quả với dữ liệu có chiều cao**: Kd-tree không hiệu quả khi dữ liệu có số chiều lớn đối với số lượng điểm nhỏ, vì cây có thể trở nên quá sâu và không cân bằng.
* **Không phù hợp với dữ liệu biến động**: Kd-tree không phù hợp với các tập dữ liệu có thể thay đổi thường xuyên, vì việc cập nhật cây có thể đòi hỏi thời gian và công sức đáng kể.
  1. **Các thao tác cơ sở (nếu là CTDL)**
     + **Xây dựng cây**
* Bắt đầu với tất cả điểm dữ liệu trong không gian K chiều.
* Chọn trục chính để phân chia không gian. Thường là chọn trục có phương sai lớn nhất.
* Chia không gian thành hai phần con dọc theo trục được chọn.
* Lặp lại quá trình này cho các phần tử con, tạo ra các nút con đến khi một điều kiện dừng được đáp ứng.
  + - **Chèn điểm**
* Duyệt cây: bắt đầu từ gốc, duyệt xuống cây theo chiều được chọn tại mỗi mức cho đến khi tìm thấy vị trí thích hợp
* Thêm nút mới: Chèn điểm mới vào vị trí lá thích hợp theo chiều phân chia tại mức đó.
  + - **Xóa điểm**
* Duyệt cây: tìm điểm cần xóa bằng cách duyệt cây từ gốc xuống như khi chèn.
* Xóa điểm: nếu điểm cần xóa là lá, xóa trực tiếp. Nếu là nút không phải lá, thay thế điểm đó bằng một điểm khác thích hợp từ các nút con.
  + - **Tìm kiếm gần nhất**



* Bắt đầu từ nút gốc.
* Di chuyển xuống cây, lựa chọn nhánh mà có khả năng chứa điểm gần nhất.
* Khi di chuyển xuống, duyệt lần lượt các nút, tiến hành kiểm tra khoảng cách từ điểm tìm kiếm đến điểm phân chia của nút đó.
* Cập nhật điểm gần nhất nếu cần và tiếp tục duyệt đến khi không còn nút nào cần duyệt.
* Lợi ích giảm thiểu số lượng phép toán so với tìm kiếm tuyến tính trong không gian đa chiều, giúp tăng tốc độ và hiệu suất.
  + - **Truy vấn vùng**
* Duyệt cây: Gốc -> duyệt xuống cây, kiểm tra từng nút để xác định xem nó có nằm trong vùng truy vấn hay không.
* Sử dụng đệ quy để kiểm tra: nếu một vùng con của nút đang xét giao với vùng truy vấn, tiếp tục đệ quy kiểm tra các nút con tròn vùng đó
  + - **Tái cân bằng cây**
* Xây dựng lại cây khi mất cân bằng, thu thập tất cả các điểm sau đó xây dựng như ban đầu.
  1. **Các thư viện hỗ trợ (C++/Python)**
* <iostream>
* <vector>
* <cmath>
* <algorithm>
* FLANN và NANOFLANN: là các thư viện nhẹ và dễ sử dụng cho các tác vụ tìm kiếm lân cận gần đúng.
* CGAL cung cấp một loạt các thuật toán hình học và cấu trúc dữ liệu, bao gồm KD-tree, với khả năng mở rộng và tùy biến cao.
  1. **Vận dụng thực tế**

1. **Tìm kiếm lân cận gần nhất (Nearest neighbor search)**
   * + Hệ thống gợi ý sản phẩm: Trong các hệ thống thương mại điện tử, KD-tree được sử dụng để tìm kiếm các sản phẩm tương tự dựa trên đặc điểm sản phẩm hoặc lịch sử mua hàng của người dùng.
     + Nhận diện khuôn mặt: Tìm kiếm ảnh khuôn mặt trong cơ sở dữ liệu lớn, KD-tree giúp tìm kiếm nhanh các ảnh có khuôn mặt tương tự.
2. **Truy vấn vùng(Range query)**
   * + Hệ thống thông tin địa lí(GIS): KD-tree được sử dụng để tìm kiếm các đối tượng nằm trong một vùng địa lý cụ thể, như các của hàng trong một bán kính nhất định từ vị trí hiện tại của người dùng.
     + Cơ sở dữ liệu không gian: Truy vấn các đói tượng trong một không gian 3D như các điểm trong một khối hộp hoặc vùng cầu.
3. **Phân tích và xử lý dữ liệu lớn** 
   * + Phân tích dữ liệu đa chiều: Trong phân tích dữ liệu, KD-tree giúp tìm kiếm và truy vấn dữ liệu nhanh chóng trong không gian nhiều chiều, như các dữ liệu có thuộc tính tương tự.
     + Clustering: Các thuật toán phân cụm như K-means có thể sử dụng KD-tree để tăng tốc độ tìm kiếm các điểm gần nhất trong quá trình cập nhật tâm cụm.
4. **Đồ họa máy tính và phát triển trò chơi (computer graphics and game development)**
   * + Xử lý va chạm: KD-tree được sử dụng để phát hiện va chạm trong các ứng dụng đồ họa 3D và game, giúp xác định các đối tượng nào có khả năng va chạm với nhau.
     + Tăng tốc hiển thị cảnh: KD-tree giúp quản lý và tối ưu hóa qua trình hiển thị các đối tượng trong cảnh 3D, đặc biệt là trong việc cắt bớt các đối tượng không nằm trong tầm nhìn của camera.
5. **Robot navigation và path planning**
   * + Lập kế hoạch đường đi: KD-tree được sử dụng để tìm kiêm và lập kế hoạch đường đi tối ưu cho robot trong không gian 2D hoặc 3D, thánh các chướng ngại vật
     + Localization: Trong các hệ thống định vị theo dõi, KD-tree giúp nhanh chống xác định vị trí hiện tại của robot hoặc thiết bị dựa trên các điểm mốc hoặc các đặc điểm môi trường xung quanh.
   1. **Các biến thể và tối ưu hóa**
      * **Balanced KD-tree (KD-tree cân bằng):**

* Mục tiêu của KD-tree cân bằng là đảm bảo rằng cây được chia đều và mỗi nhánh có số lượng điểm gần bằng nhau.
* Các phương pháp cân bằng lại KD-tree bao gồm việc sử dụng các phương pháp chia cân bằng hoặc chọn các chiều chia tốt nhất để cân bằng các nhánh.
* Các kỹ thuật như "median-of-medians" có thể được sử dụng để chọn điểm chia mà đảm bảo cân bằng các nhánh.
  + - **KD-tree đa chiều (Multi-dimensional KD-tree):**
* KD-tree gốc thường chỉ áp dụng cho không gian hai hoặc ba chiều. KD-tree đa-chiều mở rộng cấu trúc này để xử lý không gian nhiều chiều.
* Trong KD-tree đa-chiều, việc chọn chiều để chia dữ liệu có thể trở nên phức tạp hơn do sự tăng chiều số.
  + - **KD-tree phân tán (Distributed KD-tree):**
* KD-tree phân tán được sử dụng khi dữ liệu lớn và không thể fit vào bộ nhớ của một máy tính duy nhất.
* Dữ liệu được phân chia thành các phần và mỗi phần được xử lý bởi một máy tính riêng biệt.
* Các máy tính có thể hoạt động độc lập để xây dựng và duy trì cây cho phần dữ liệu của mình.
  + - **KD-tree phiên bản song song (Parallel KD-tree):**
* KD-tree phiên bản song song sử dụng đa luồng hoặc vi xử lý song song để tăng tốc độ xây dựng cây và thực hiện các phép tìm kiếm.
* Các phép toán được thực hiện song song trên nhiều luồng hoặc núcleo của máy tính, cung cấp hiệu suất tốt hơn cho các ứng dụng có khả năng song song hóa.
  + - **KD-tree có cấu trúc tổ chức (Structured KD-tree):**
* Các biến thể này tổ chức dữ liệu một cách cụ thể để tối ưu hóa việc truy cập.
* Ví dụ, Octree là một biến thể của KD-tree được sử dụng trong việc tổ chức không gian ba chiều thành các ô lập phẳng, giúp tối ưu hóa việc truy cập dữ liệu trong không gian ba chiều.

1. **Tài liệu tham khảo**

* [Thuật toán Ball Tree và KD Tree - GeeksforGeeks](https://www.geeksforgeeks.org/ball-tree-and-kd-tree-algorithms/)
* <https://chat.openai.com/>
* [k-d tree - Wikipedia](https://en.wikipedia.org/wiki/K-d_tree)
* [15.4. KD Trees — CS3 Data Structures & Algorithms (vt.edu)](https://opendsa-server.cs.vt.edu/ODSA/Books/CS3/html/KDtree.html)
* [Chương 6: TREES - 1. Tree là gì? Lý thuyết về Binary Tree. (viblo.asia)](https://viblo.asia/p/chuong-6-trees-1-tree-la-gi-ly-thuyet-ve-binary-tree-obA46PM9LKv)
* [Introduction to K-D Trees | Baeldung on Computer Science](https://www.baeldung.com/cs/k-d-trees)

1. **Phụ lục 1: Chương trình minh họa**

Input:

Dòng 1: nhập n số điểm

Dòng 2: nhập n điểm

Dòng 3: nhập điểm cần tìm gần nhất

Output: xuất ra điểm gần nhất