**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG**

**BÁO CÁO**

**NGHIÊN CỨU CẤU TRÚC DỮ LIỆU VÀ THUẬT TOÁN**

**ĐỀ TÀI:**

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

A cube with lines and dots

Description automatically generated

**MSSV:** 23520717

**Sinh viên thực hiện:** Hoàng Ngọc Khánh

1. **GIỚI THIỆU CHUNG VỀ CẤU TRÚC DỮ LIỆU VÀ THUẬT TOÁN**
2. **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.

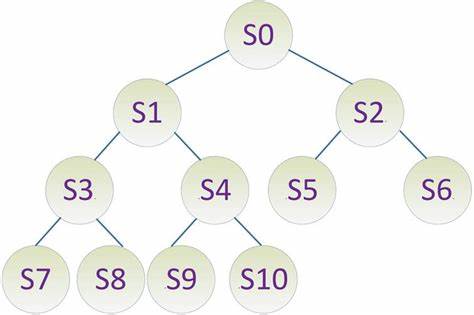
1. **Tổ chức bài viết**

* 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. **NỘI DUNG CHÍNH**
2. **Khái niệm**

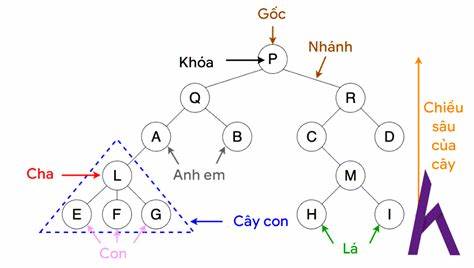
* KD-tree là một cấu trúc dữ liệu phân vùng không gian để tổ chức các điểm trong không gian k chiều.
* KD-tree là một trường hợp đặc biệt của cây phân vùng không gian nhị phân.
* Là một cây nhị phân trong đó mỗi nút là một điểm k chiều. Mỗi nút không phải lá có thể được coi là ngầm tạo ra một siêu mặt 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.

VD:



1. **Cách hoạt động của thuật toán KD-tree**
   1. **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.
  1. **Tìm kiếm**
* Bắt đầu từ nút gốc của cây.
* So sánh điểm tìm kiếm với điểm phân chia của nút hiện tại dựa trên trục được chọn.
* Dựa vào kết quả so sanh, di chuyển đến nút con tương ứng và tiếp tục quá trình này cho đến khi đạt được một điều kiện dừng (vd: tìm thấy điểm mong muốn hoặc không còn nút nào để duyệt).
  1. **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.



1. **Phân tích KD-tree**
   1. **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.
* **Hỗ trợ tìm kiếm k-nearest neighbors**: Kd-tree có thể được sử dụng để tìm kiếm k điểm gần nhất trong không gian nhiều chiều một cách hiệu quả.

**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. **Mô tả cách KD-tree tích hợp vào các ứng dụng cụ thể và lợi ích của nó mang lại**

**-Tìm kiếm điểm gần nhất**: KD-tree thường được sử dụng để tìm điểm gần nhất hoặc các điểm trong một phạm vi nhất định trong không gian k chiều. Ứng dụng này rất hữu ích trong các hệ thống thông tin địa lý, tìm kiếm hình ảnh, và trò chơi máy tính.

**- Phân loại và gom nhóm**: Trong các vấn đề phân loại và gom nhóm dữ liệu, KD-tree có thể được sử dụng để tìm kiếm các điểm tương đồng hoặc tạo ra các cụm dữ liệu dựa trên sự tương đồng trong không gian k chiều.

**- Tìm kiếm nhanh trong cơ sở dữ liệu**: KD-tree cũng được sử dụng để tối ưu việc tìm kiếm trong cơ sở dữ liệu, đặc biệt là khi cần tìm kiếm nhanh trong dữ liệu có cấu trúc và chiều dữ liệu lớn.

**- Xử lý dữ liệu hình ảnh**: Trong xử lý hình ảnh và video, KD-tree có thể được sử dụng để tìm kiếm các điểm ảnh tương tự hoặc để phát hiện các vùng quan trọng trong không gian hình ảnh.

**- Tìm kiếm đường đi tối ưu**: KD-tree có thể được sử dụng để tìm kiếm đường đi tối ưu trong không gian k chiều, chẳng hạn như tìm đường đi ngắn nhất giữa hai điểm trong một mạng lưới đường.

1. **Thường hợp phổ biến nhất tìm kiếm gần nhất**

* **B1: Xác định nút gốc**: Bắt đầu từ nút gốc của cây KD.
* **B2: Tìm kiếm đường đi xuống cây**: Dựa vào giá trị của thuộc tính của nút gốc, so sánh với giá trị của thuộc tính của điểm tham chiếu. Dựa vào kết quả so sánh, chọn một trong hai nhánh của cây để tiếp tục tìm kiếm.
* **B3: Tìm kiếm qua các nút con**: Lặp lại quá trình tìm kiếm từ bước 2 trên các nút con cho đến khi đạt được nút lá.
* **B4: Cập nhật điểm gần nhất**: Trong quá trình đi xuống cây, đánh giá khoảng cách của điểm gần nhất hiện tại với điểm tham chiếu. Nếu tìm thấy một nút lá mà khoảng cách từ nút lá đến điểm tham chiếu nhỏ hơn khoảng cách của điểm gần nhất hiện tại, cập nhật điểm gần nhất.
* **B5: Quay lại các nút cha**: Sau khi đạt được nút lá, quay lại các nút cha và kiểm tra xem có nút cha nào khác mà có thể chứa điểm gần nhất hơn không.
* Quá trình này tiếp tục cho đến khi đã kiểm tra tất cả các nút hoặc đạt được một điểm gần nhất đủ gần với điểm tham chiếu.
* 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.

1. **Các biến thể và tối ưu hóa**
   1. **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.
  1. **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.
  1. **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.
  1. **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.
  1. **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. **Triển khai và thử nghiệm**
2. **Kết luận**
3. CÁC 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)