**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

**Ảnh có chứa văn bản, áp phích, Phông chữ, Đồ họa

Mô tả được tạo tự động**

**BÁO CÁO TIẾN ĐỘ TUẦN**

**Môn học: Project 1**

***Chủ đề: Ứng dụng thuật toán tìm kiếm***

Tuần: 18/11 – 23/11

**Giáo viên hướng dẫn: Lê Bá Vui**

**Sinh viên thực hiện: *Lê Minh Triết***

**Hà Nội - 2024**

Contents

[1. Công việc đã hoàn thành trong tuần 3](#_Toc183261751)

[1.1. Lựa chọn các thuật toán tìm kiếm 3](#_Toc183261752)

[1.2. Tạo khung chính cho dự án 3](#_Toc183261753)

[1.3. Thiết kế ban đầu cho class SearchAlgorithm 3](#_Toc183261754)

[1.4. Thiết kế ban đầu cho class NeedSortedAlgorithm 4](#_Toc183261755)

[1.5. Triển khai các thuật toán LinearSearch, BinarySearch 4](#_Toc183261756)

[1.5.1. LinearSearch 4](#_Toc183261757)

[1.5.2. BinarySearch 4](#_Toc183261758)

[1.5.3. Đánh giá kết quả 4](#_Toc183261759)

[2. Dự kiến công việc tuần sau 6](#_Toc183261760)

1. Công việc đã hoàn thành trong tuần

1.1. Lựa chọn các thuật toán tìm kiếm

7 thuật toán tìm kiếm cơ bản được lựa chọn

1. Tìm kiếm tuần tự (Linear Search).
2. Tìm kiếm bằng bảng băm (Hashing).
3. Tìm kiếm nhị phân (Binary Search).
4. Tìm kiếm nhảy (Jump Search).
5. Tìm kiếm Exponential (Exponential Search).
6. Tìm kiếm Ternary (Ternary Search).
7. Tìm kiếm nhị phân theo Interpolation (Interpolation Search).

1.2. Tạo khung chính cho dự án

Dự án đi theo mô hình hướng đối tượng, sẽ gồm các class chính được miêu tả trong biểu đồ sau

Ảnh có chứa văn bản, ảnh chụp màn hình, biểu đồ, hàng

Mô tả được tạo tự động

Hình 1. Biểu đồ Class cho Project

Trong đó, SearchAlgorithm là lớp cha cho mọi thuật toán tìm kiếm, và SearchContext là class để quyết định lớp thuật toán tìm kiếm nào trong quá trình chạy sẽ được gọi tới (sử dụng Strategy Pattern).

Ở các con của SearchAlgorithm có một lớp con NeedSortedSearchAlgorithm đại diện cho các thuật toán tìm kiếm phải sắp xếp lại dãy.

1.3. Thiết kế ban đầu cho class SearchAlgorithm

Các thuộc tính cơ bản:

* arr: một vector<int> lưu lại danh sách cần tìm kiêm.
* Các thuộc tính để lưu lại thời gian chạy (bao gồm tổng thời gian chạy, thời gian xây dựng các cấu trúc dữ liệu (bảng băm với HashTableSearch, hoặc là thời gian xây dựng mảng đã sắp xếp với các thuật toán cần sắp xếp) và thời gian tìm kiếm.

Các hàm quan trọng

* int search(int value): tìm kiếm một giá trị và trả về vị trí.
* void printInfo(): In ra thời gian chạy để tìm kiếm (bao gồm các loại thời gian trên).
* void setArr(vector<int> arr): Đưa dãy cần tìm kiếm vào.

1.4. Thiết kế ban đầu cho class NeedSortedAlgorithm

Các thuộc tính cơ bản

* isSorted: đánh dấu xem đã sắp xếp lại dãy cần tìm kiếm chưa.
* sortedIndexArr: một vector chứa các pair lưu giữ mảng sau khi sắp xếp và chỉ số cũ trong mảng ban đầu.

Các phương thức quan trọng

* bool isAscending(vector<T> arr): Kiểm tra dãy có tăng không.
* void buildSortedArr(): Xây dựng sortedIndexArr.

1.5. Triển khai các thuật toán LinearSearch, BinarySearch

1.5.1. LinearSearch

**Yêu cầu:** Không

**Thuật toán:** Duyệt qua từng phần tử của dãy, từ đầu tới cuối tới khi tìm thấy hoặc kết thúc dãy.

**Độ phức tạp:** Tồi nhất là O(n).

1.5.2. BinarySearch

**Yêu cầu:** Dãy đã được sắp xếp

**Thuật toán:** So sánh với giá trị nằm giữa dãy. Nếu đây là giá trị cần tìm thì trả về. Nếu giá trị cần tìm nhỏ hơn thì thực hiện tương tự với dãy ở nửa đầu. Ngược lại tìm ở nửa sau của dãy.

**Độ phức tạp:** Trung bình là O(log n).

1.5.3. Đánh giá kết quả

Đã triển khai thành công hai thuật toán tìm kiếm cơ bản nhất này. Source code đã được đính kèm ở phần dưới.

Đã thử test với 2 dãy sau:

* Test 1: Dãy gồm 1000000 số từ 0 tới 106-1 theo thứ tự tăng dần.
* Test 2: Dãy gồm 1000000 số từ 0 tới 106-1 theo thứ tự giảm dần.

Với mỗi Test, thực hiện tìm 12 giá trị cách đều nhau là i\*106/13 (lấy phần nguyển, thu được kết quả sau

Ảnh có chứa văn bản, ảnh chụp màn hình, mẫu, tác phẩm nghệ thuật

Mô tả được tạo tự động

Hình 2. Test 1

Ảnh có chứa văn bản, ảnh chụp màn hình, mẫu, tác phẩm nghệ thuật

Mô tả được tạo tự động

Hình 3. Test 2

Nhận xét ban đầu:

Table 1

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test | Search  Algorithm | Thời gian tốt nhất (ms) | | Thời gian trung bình (ms) | | Thời gian tồi nhất (ms) | |
| Total | Search | Total | Search | Total | Search |
| 1 | Linear | 0.432 | 0.430 | 2.056 | 2.055 | 3.919 | 3.917 |
| 1 | Binary | 0.001 | 0.001 | 3.946 | 0.002 | 47.319 | 0.002 |
| 2 | Linear | 0.494 | 0.492 | 2.389 | 2.387 | 4.344 | 4.341 |
| 2 | Binary | 0.002 | 0.001 | 14.028 | 0.001 | 168.257 | 0.003 |

Do vậy, có thể nhận xét rằng:

* Binary Search mất thời gian chủ yếu khi build (khi sắp xếp lại dãy). Còn lại thời gian tìm kiếm luôn nhỏ hơn 2000 lần so với Linear Search, điều này là phù hợp do độ phức tạp của Binary Search là O(logn) còn của LinearSearch là O(n).
* Nếu chỉ có 1 truy vấn tìm kiếm cho dãy, Linear Search hoạt động hiệu quả hơn, nhưng nếu có nhiều truy vấn thì BinarySearch là hiệu quả hơn rất nhiều.

2. Dự kiến công việc tuần sau

* Thiết lập giao diện để tương tác với người dùng trên Terminal.
* Triển khai các thuật toán Ternary Search và Jump Search.