TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI

KHOA CÔNG NGHỆ THÔNG TIN

---------------o0o---------------



**Bài tập lớn môn học**

**CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT**

Giảng viên hưỡng dẫn: TS. Hoàng Văn Thông

Sinh viên thực hiện: Lý Trần Vinh

Lớp: CNTT VA 1

**Hà Nội tháng 10 năm 2023**

**Mục lục**

**Bài 1 (Bài số 6 trong danh sách bài tập)**

* 1. **Đề Bài**:

Xây dựng một lớp quản lý một từ điển Anh-Việt, từ điển được lưu vào cây nhị phân mà tại mỗi nút của nó lưu một từ tiếng Anh và nghĩa tiêngs Việt tương ứng có các phương thức:

1. Đọc từ điển từ file để xây dựng cây
2. Ghi từ điển trong cây vào file (duyệt theo thứ tự giữa, duyệt đến nút nào thì ghi vào file giá trị của nút đó)
3. Thêm một phần tử mớ vào từ điểm đang lưu trong cây
4. Xoá bỏ một từ điểm đang lưu trong cây
5. Cập nhập lại nghĩa một từ đang lưu trong cây
6. Tìm kiếm một từ trong cây
   1. **Phân tích bài toán:**

**- Phân tích đề thực hiện yêu cầu, từ đó ta xác định và xây dụng các hàm , chức năng, lớp, thuộc tính:**

**1.2.1. Lớp *node:***

**- Biến:**

**- `E`: Lưu trữ từ tiếng Anh.**

**- `V`: Lưu trữ nghĩa tiếng Việt.**

**- `left`, `right`: Con trỏ trỏ đến nút con bên trái và bên phải.**

**- Hàm:**

**- node(string E, string V): Hàm khởi tạo một nút với giá trị cho từ tiếng Anh và nghĩa tiếng Việt.**

**1.2.2. Lớp Dictionary:**

**- Phương thức:**

**- `add`: Thêm một từ mới vào cây nhị phân.**

**- `minVal`: Tìm giá trị nhỏ nhất trong cây (thường được sử dụng trong trường hợp xoá nút có hai con).**

**- `del`: Xoá một từ từ cây nhị phân.**

**- `find`: Tìm một từ trong cây nhị phân.**

**- `updataVoc`: Cập nhật nghĩa của một từ trong cây hoặc thêm mới nếu từ không tồn tại.**

**- `readFile`: Đọc dữ liệu từ file để xây dựng cây nhị phân.**

**- `solveWriteFile`: Hàm hỗ trợ duyệt cây và ghi dữ liệu vào file.**

**- `writeFile`: Ghi dữ liệu của cây nhị phân vào file.**

**1.2.3. Hàm run:**

**- Là hàm chính của chương trình, chứa vòng lặp để thực hiện các lựa chọn từ người dùng.**

**1.2.4. Hàm showData:**

**- Hiển thị dữ liệu từ cây ra màn hình để kiểm tra.**

**1.2.5. Hàm del\_ram:**

**- Dùng để giải phóng bộ nhớ khi kết thúc chương trình, xoá toàn bộ cây.**

**1.2.6. Hàm main:**

**- Là hàm bắt đầu thực thi chương trình, chỉ gọi hàm `run`.**

* 1. **Cài đặt chương trình:**

[***https://github.com/Youknow2509/btl\_ctdl/blob/main/ex6.cpp***](https://github.com/Youknow2509/btl_ctdl/blob/main/ex6.cpp)

* 1. **Phân tích thời gian chạy của từng phương thức có trong các lớp:**
     1. Hàm add: O(n)
* Trường hợp tốt nhất (best case): Trường hợp này xảy ra khi cây đã cân bằng và chiều cao của cây là log(n), với n là số lượng nút trong cây. Trong trường hợp này, mỗi lần gọi đệ qui giảm một nửa số lượng nút cần xem xét, do đó thời gian chạy là O(log n).
* Trường hợp xấu nhất (worst case): Trường hợp này xảy ra khi cây không cân bằng và trở thành một dãy thẳng. Trong trường hợp này, mỗi lần gọi đệ qui chỉ giảm một nút, và thời gian chạy là O(n), với n là số lượng nút trong cây.
  + 1. Hàm minVal: O(n)
* Thời gian chạy trung bình:

Trong trường hợp cây cân bằng, thời gian chạy của hàm này là O(log n), với n là số lượng nút trong cây. Mỗi lần lặp giảm chiều cao của cây đi một đơn vị.

* Trường hợp xấu nhất:

Trong trường hợp cây không cân bằng và trở thành một dãy thẳng, thời gian chạy là O(n), với n là số lượng nút trong cây. Mỗi lần lặp chỉ giảm một nút, và số lượng nút cần xem xét là n.

* + 1. Hàm del: O(n)
* Trường hợp tốt nhất (best case):

Trường hợp này xảy ra khi cây cân bằng và chiều cao của cây là log(n), với n là số lượng nút trong cây. Trong trường hợp này, mỗi lần gọi đệ qui giảm một nửa số lượng nút cần xem xét, do đó thời gian chạy là O(log n).

Trường hợp xấu nhất (worst case):

* Trường hợp này xảy ra khi cây không cân bằng và trở thành một dãy thẳng. Trong trường hợp này, mỗi lần gọi đệ qui chỉ giảm một nút, và số lượng nút cần xem xét là n. Do đó, thời gian chạy là O(n).
  + 1. Hàm find: O(n)
* Trường hợp tốt nhất (best case):

Trường hợp này xảy ra khi từ cần tìm nằm ở ngay nút gốc của cây. Trong trường hợp này, thời gian chạy là O(1), vì chúng ta đã tìm thấy từ ở ngay nút gốc.

* Trường hợp xấu nhất (worst case):

Trường hợp này xảy ra khi cây không cân bằng và trở thành một dãy thẳng. Trong trường hợp này, thời gian chạy là O(n), với n là số lượng nút trong cây.

Mỗi lần gọi đệ qui chỉ giảm một nút, và số lượng nút cần xem xét là n.

* + 1. Hàm updataVoc: O(n)
* Thời gian chạy của hàm này phụ thuộc vào thời gian chạy của hàm find và add.

Trong trường hợp tốt nhất, khi từ cần cập nhật nằm ở ngay nút gốc của cây, thời gian chạy sẽ là O(1).

Trong trường hợp xấu nhất, khi cây không cân bằng và trở thành một dãy thẳng, thời gian chạy sẽ là O(n), với n là số lượng nút trong cây.

* + 1. Hàm readFile: O(n log n)
* Trong trường hợp tốt nhất, khi tệp rỗng, thời gian chạy là O(1).
* Trong trường hợp xấu nhất, khi có n từ vựng trong tệp, và cây trở thành một dãy thẳng, thời gian chạy là O(n log n) (do cần cân bằng cây).
  + 1. Hàm solveWriteFile O(n):
* Trong trường hợp tốt nhất và xấu nhất, thời gian chạy đều là O(n), với n là số lượng nút trong cây.
  + 1. Hàm writeFile: O(n)
* Trong trường hợp tốt nhất và xấu nhất, thời gian chạy của hàm writeFile đều là O(n), với n là số lượng nút trong cây.
  + 1. Hàm showData: O(n)
* Mỗi nút trong cây chỉ được ghé thăm một lần, và với mỗi nút, hàm thực hiện một số thao tác cố định (in thông tin từ nút lên màn hình). Trong trường hợp tốt nhất và xấu nhất, thời gian chạy của hàm showData đều là O(n), với n là số lượng nút trong cây.
  + 1. Hàm del\_ram: O(n)
* Mỗi nút trong cây chỉ được xoá một lần, và với mỗi nút, hàm thực hiện một số thao tác cố định (gọi đệ qui để xoá các nút con và sau đó xoá nút hiện tại). Trong trường hợp tốt nhất và xấu nhất, thời gian chạy của hàm del\_ram đều là O(n), với n là số lượng nút trong cây.

**Bài 2 (Bài số 14 trong danh sách bài tập)**

* 1. **Đề Bài**:

Sử dụng lớp bảng bắm để xây dựng lớp quản lý một thư viện giúp người quản lý thư viện có thể tra cúu nhanh một loại danh sách nào đó nằm ở đâu trong thư viện. Với các phương thức:

1. Nạp thông tin sách của thư viện từ file vào bảng băm
2. Tìm kiếm bằng phương pháp tìm kiếm trên bảng băm
3. Thêm sách mới vào thư viện
4. Xoá sách khi biết tên
5. Lưu thông tin thư viện trên file

*Lưu ý: Mỗi loại sách có mã sách, tên sách, mô tả vị trí trong thư viện, số lượng*

* 1. **Phân tích bài toán:**

**2.2.1. Lớp `Book`:**

**- Lớp này đại diện cho thông tin của một cuốn sách với các thuộc tính như `bookCode` (mã sách), `name` (tên sách), `loc` (vị trí trong thư viện), và `amount` (số lượng sách).**

**2.2.2. Lớp `Library`:**

**-Thuộc tính:**

**- `sizeHashTable`: Kích thước của bảng băm (đặt là `10000` trong trường hợp này).**

**- `books`: Mảng chứa thông tin sách.**

**- Phương thức:**

**- `createHashTable()`: Khởi tạo bảng băm, đặt tất cả các phần tử của mảng `books` về 0.**

**- `hash(nameBook)`: Phương thức này thực hiện hàm băm để xác định vị trí của sách trong bảng băm dựa trên tên sách.**

**- `cmpBook(b1, b2)`: So sánh hai đối tượng sách để kiểm tra xem chúng có giống nhau hay không.**

**- `readFile(filename)`: Đọc thông tin sách từ file và nạp vào bảng băm.**

**- `writeFile(filename)`: Ghi thông tin sách từ bảng băm vào file.**

**- `find(nameBook)`: Tìm kiếm sách trong bảng băm theo tên sách.**

**- `findBook(nameBook)`: Hiển thị thông tin của sách khi tìm kiếm theo tên sách.**

**- `insertBook(b)`: Thêm sách mới vào thư viện sử dụng phương thức linear probing để giải quyết xung đột.**

**- `delBook(nameBook)`: Xoá sách khỏi thư viện dựa trên tên sách.**

**- `getBooks()`: Trả về mảng chứa thông tin sách.**

**- `getSizeHashTable()`: Trả về kích thước của bảng băm.**

**2.2.3. Hàm `showLibrary` và `run`:**

**- `showLibrary(b[], size)`: Hiển thị thông tin sách trong thư viện.**

**- `run()`: Hàm chính để chạy chương trình. Cho phép người dùng chọn các tùy chọn như nhập dữ liệu từ file, tìm kiếm sách, thêm sách mới, xoá sách, lưu thông tin thư viện, hiển thị thông tin sách, và thoát chương trình.**

**2.2.4. Hàm `main()`:**

**- Gọi hàm `run()` để bắt đầu chương trình.**

**- Bảng băm được triển khai thông qua phương thức `hast` và sử dụng linear probing để giải quyết xung đột. Một số điều cần lưu ý để giữ tính hiệu quả của bảng băm là tránh việc cập nhật các sách giống nhau và xử lý đọc file bằng cách bỏ qua dòng trống.**

**- Việc nhập và xuất thông tin sách từ/ra file được thực hiện thông qua các phương thức `readFile` và `writeFile`.**

**- Chương trình cho phép người dùng thực hiện các thao tác quản lý thư viện thông qua giao diện dòng lệnh, cung cấp tính năng cơ bản cho việc tra cứu, thêm, xoá, và lưu thông tin sách.**

* 1. **Cài đặt chương trình:**

[***https://github.com/Youknow2509/btl\_ctdl/blob/main/ex14.cpp***](https://github.com/Youknow2509/btl_ctdl/blob/main/ex14.cpp)

* 1. **Phân tích thời gian chạy của từng phương thức có trong các lớp:**
     1. Hàm createHashTable(): O(sizeHashTable)
     2. Hàm hash(): O(L), với L là độ dài của nameBook (tên sách).
     3. Hàm cmpBook(): O(1)
     4. Hàm readFile(): Thời gian chạy: Phụ thuộc vào số lượng sách trong file và kích thước của bảng băm.
     5. Hàm writeFile (): Thời gian chạy: Phụ thuộc vào số lượng sách trong file và kích thước của bảng băm.
     6. Hàm find(): O(n)
     7. Hàm findBook(): Phụ thuộc vào hàm find()
     8. Hàm insertBook(): O(N)
     9. Hàm delBook(): Phụ thuộc hàm find()
     10. Hàm showLibary(): O(n)