MỤC LỤC

PHỤ LỤC HÌNH ẢNH ***2***

PHỤ LỤC BẢNG ***2***

MỞ ĐẦU ***3***

CHƯƠNG 1. KĨ THUẬT BĂM ***5***

1.1. TÌM HIỂU KĨ THUẬT BĂM *5*

1.1.1. Giới thiệu *5*

1.1.2. Ví dụ *6*

1.2. XÂY DỰNG HÀM BĂM *7*

1.2.1. Phương pháp chia *7*

1.2.2. Phương pháp nhân *8*

1.2.3. Phương pháp phân đoạn *8*

1.2.4. Hàm băm cho các giá trị khoá là xâu ký tự *9*

1.2.5. Bảng băm *10*

1.3. GIẢI QUYẾT ĐỤNG ĐỘ *12*

1.3.1. Phương pháp địa chỉ mở *12*

1.3.2. Phương pháp kết nối (danh sách liên kết) *14*

CHƯƠNG 2. BÀI TOÁN ỨNG DỤNG ***16***

2.1. PHÁT BIỂU BÀI TOÁN VÀ TỔ CHỨC CHƯƠNG TRÌNH. *16*

2.1.1. Phát biểu bài toán *16*

2.1.2. Phân tích bài toán *16*

2.1.3. Tổ chức dữ liệu *16*

2.1.4. Xác định thuật toán (đọc lại) *16*

2.2. GIẢI THUẬT *17*

2.2.1. Giải thuật chính *17*

2.2.2. Giải thuật cụ thể. *17*

2.2.3. Xây dựng chương trình. *18*

CHƯƠNG 3. KẾT LUẬN ***24***

3.1. NHỮNG KẾT QUẢ ĐẠT ĐƯỢC *24*

3.2. NHỮNG HẠN CHẾ *24*

3.3. HƯỚNG PHÁT TRIỂN CỦA ĐỀ TÀI *24*

TÀI LIỆU THAM KHẢO ***25***

PHỤ LỤC HÌNH ẢNH

Hình 1. 1. Minh họa hàm băm 6

Hình 1. 2. Mô tả dữ liệu 10

Hình 1. 3. Phương pháp tạo dây chuyền 15

PHỤ LỤC BẢNG

Bảng 1. 1. Bảng kết quả phân tích 6

Bảng 1. 2. Ví dụ về phương pháp nhân 8

MỞ ĐẦU

- *Lý do, mục đích chọn đề tài*
  - Phép băm là một bài toán cổ điển của khoa học máy tính, đã có nhiều thuật toán khác nhau được nghiên cứu và được dùng rất rộng rãi. Có một số lượng lớn những phân tích và kinh nghiệm để cung cấp các thủ tục băm cho rất nhiều ứng dụng khác nhau. Hàm băm một hàm mạnh trong mã hóa, chữ kí điện từ, xác nhận tính toàn vẹn của thông điệp…
  - Phép băm là một thí dụ tốt về vấn đề dung hoà giữa thời gian chạy và dung lượng bộ nhớ sử dụng. Nếu không có sự giới hạn về bộ nhớ thì chúng ta có thể thực hiện bất kỳ một thao tác tìm kiếm nào chỉ với một lần truy xuất bộ nhớ bằng cách sử dụng khoá như một địa chỉ bộ nhớ. Nếu không có sự giới hạn về thời gian thì chúng ta có thể tối thiểu hoá dung lượng sử dụng bộ nhớ bằng cách dùng một phương pháp tìm kiếm tuần tự. Phép băm cung cấp một phương pháp dùng một lượng vừa phải của bộ nhớ và để làm một sự cân bằng giữa hai thái cực này. Sử dụng hiệu quả bộ nhớ có sẵn và truy xuất nhanh đến bộ nhớ là quan tâm chủ yếu của bất kỳ một phương pháp băm. Phép băm đưa ra một cách tiếp cận đầy đủ tới việc tìm kiếm khác hẳn với việc tìm kiếm đã biết như tìm kiếm trên cấu trúc cây… Do đó phương pháp băm là phương pháp rất hiệu quả để cài đặt từ điển.
  - Chính vì vậy đề tài *"ứng dụng bảng băm xây dựng Từ điển Anh-Việt"* được thực hiện nhằm giới thiệu về kỹ thuật băm và tìm hiểu ứng dụng quan trọng của nó trong từ điển. Có thể lập được một chương trình viết bằng ngôn ngữ C++ giúp người học có thể tra cứu được từ điển Anh-Việt.
- *Mục tiêu cần đạt được của đề tài*
  - Đề tài này được thực hiện nhằm đạt được mục tiêu là hiểu rõ, sâu sắc hơn về phép băm, các hàm băm. Tìm hiểu ứng dụng quan trọng của nó trong ứng dụng tìm kiếm từ trong từ điển.

- *Cơ sở lý thuyết*
  - Phép băm.
  - Danh sách liên kết.
  - Toán học.
- *Đối tượng và phạm vi nghiên cứu*
  - Ngôn ngữ C++.
  - Bảng băm và hàm băm.
  - Danh sách liên kết.
  - Từ điển Anh-Việt và cấu trúc từ điển.
- *Phương pháp nghiên cứu*
  - Tìm hiểu thông tin trên mạng internet, sách, báo, tạp chí…
  - Thông qua sự hướng dẫn của thầy cô giáo và nghiên cứu những tài liệu tham khảo liên quan.
- *Cấu trúc đề tài*
  - Mở đầu.
  - Chương I: Tìm hiểu sơ lược về kĩ thuật băm.
  - Chương II: Ứng dụng bảng băm trong từ điển.
  - Chương III: Kết luận
  - Tài liệu tham khảo

CHƯƠNG 1. KĨ THUẬT BĂM

1.1. TÌM HIỂU KĨ THUẬT BĂM

1.1.1. Giới thiệu

Phép băm được đề xuất và hiện thực trên máy tính từ những năm 50 của thế kỷ XX. Nó dựa trên ý tưởng: biến đổi giá trị khóa thành một số (xử lý băm) và sử dụng số này để đánh chỉ cho bảng dữ liệu.

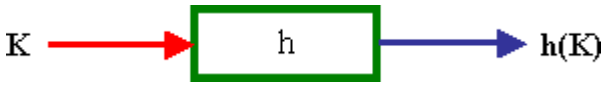
Các phép toán trên các cấu trúc dữ liệu như danh sách, cây nhị phân,… phần lớn được thực hiện bằng cách so sánh các phần tử của cấu trúc, do vậy thời gian truy xuất không nhanh và phụ thuộc vào kích thước của cấu trúc.

Chúng ta sẽ khảo sát một cấu trúc dữ liệu mới được gọi là bảng băm (hash table). Các phép toán trên bảng băm sẽ giúp hạn chế số lần so sánh, và vì vậy sẽ cố gắng giảm thiểu được thời gian truy xuất. Độ phức tạp của các phép toán trên bảng băm thường có bậc là O(1) và không phụ thuộc vào kích thước của bảng băm.[1]

Từ điển Anh – Việt là một ứng dụng rất quen thuộc đối với người dùng máy tính nói chung và người học tiếng anh nói riêng. Gần đây, các phần mềm từ điển Anh – Việt phát triển rất mạnh mẽ cả về số từ vựng cũng như chức năng, như Lạc Việt, Babylon, ngoài ra còn rất nhiều từ điển trực tuyến như translate.google.com, vdict.com, tratu.vn, 1tudien.com …

❖ ***PHÉP BĂM (Hash Function)***

Trong hầu hết các ứng dụng, khoá được dùng như một phương thức để truy xuất dữ liệu. Hàm băm được dùng để ánh xạ giá trị khóa vào một dãy các địa chỉ của bảng băm (*Hình 1.1*).



Hình 1. 1. Minh họa hàm băm

Khóa có thể là dạng số hay dạng chuỗi. Giả sử có 2 khóa phân biệt k_i và k_j nếu $\boldsymbol{h(k_i)=h(k_j)}$ thì hàm băm bị đụng độ.

Bằng một quy tắc biến đổi nào đó, từ giá trị của khoá ta tính ra một địa chỉ (địa chỉ tương đối). Địa chỉ này sẽ dùng để lưu trữ bản ghi tương ứng, đồng thời cũng để tìm kiếm bản ghi ấy. Như vậy nghĩa là ta thiết lập một hàm băm h(k) thực hiện phép ánh xạ tập các giá trị của k lên tập các địa chỉ tương đối, nghĩa là các số nguyên từ 0 đến m-1 mà ta gọi là bảng địa chỉ, m được gọi là độ dài hay kích thước của bảng. Như vậy ta luôn có: $0 \leq h(k) < m$. Giá trị của h(k) sẽ được sử dụng khi lưu trữ cũng như khi tìm kiếm bản ghi ứng với k.[2]

1.1.2. Ví dụ

Xét các bản ghi có khóa tương ứng là các số nguyên gồm không quá 4 chữ số thập phân, chẳng hạn 5402, 0367, 1246, 2983… Giả sử kích thước của bảng là m = 1000 nghĩa là các địa chỉ tính được phải nằm trong khoảng 0 đến 999. Ta chọn quy tắc tính địa chỉ như sau: "Lấy 3 chữ số cuối cùng của khóa làm địa chỉ". Như vậy ứng với các khóa nêu trên ta sẽ có kết quả:

Giá trị khóa	Địa chỉ
5402	402
0376	367
1246	246
2983	983

Bảng 1. 1. Bảng kết quả phân tích

Khi lưu trữ bản ghi ứng với khóa chẳng hạn 5042 sẽ được đưa vào một ô gồm một số byte trong bộ nhớ thực có địa chỉ là $A_0 + 402$. Đến khi tìm kiếm thì địa chỉ $A_0 + 402$ lại được xác định để sử dụng. Để tiện trình bày ta coi địa chỉ $A_0 = 0$ nghĩa là tạm thời coi địa chỉ tương đối như địa chỉ thực.

Rõ ràng với phương pháp này các khóa có giá trị khác nhau cũng có thể cùng ứng với một địa chỉ, ví dụ 5402, 7402, 0402 đều cùng một địa chỉ 402. Lúc đó ta nói hiện tượng đụng độ (collision).

- *Một hàm băm tốt phải thõa mãn các điều kiện sau:*
  - Tính toán nhanh.
  - Các khóa được phân bố đều trong bảng.
  - Ít xảy ra đụng độ.
  - Xử lý được các loại khóa có kiểu dữ liệu khác nhau.

1.2. XÂY DỰNG HÀM BĂM

1.2.1. Phương pháp chia

Nguyên tắc của nó rất đơn giản *"lấy số dư của phép chia giá trị khoá cho kích thước m của bảng băm để làm địa chỉ băm"* nghĩa là:

$$h(k) = k \bmod m$$

Trong đó : k: là khóa.

m: là kích thước của bảng.

Phương pháp này đơn giản nhưng có thể không cho kết quả ngẫu nhiên lắm. Chẳng hạn *m*=1000 thì h(k) chỉ phụ thuộc vào ba số cuối của khóa mà không phụ thuộc vào các số đứng trước. Kết quả có thể ngẫy nhiên hơn nếu B là một số nguyên tố.[3]

1.2.2. Phương pháp nhân

Nguyên tắc của nó là *"lấy khóa nhân với chính nó rồi chọn một số chữ số ở giữa làm kết quả của hàm băm"* nghĩa là:

Ví dụ:

x	x^2	h(x) gồm 3 số ở giữa
5402	29181604	181 hoàûc 816
0367	00134689	134 346
1246	01552516	552 525
2983	08898289	898 982

Bảng 1. 2. Ví dụ về phương pháp nhân

Vì các chữ số ở giữa phụ thuộc vào tất cả các chữ số có mặt trong khóa do vật các khóa có khác nhau đôi chút thì hàm băm cho kết quả khác nhau.[4]

1.2.3. Phương pháp phân đoạn

Nếu khoá có kích thước lớn, kích thước thay đổi thì người ta áp dụng phương pháp phân đoạn. Tức là phân khóa ra nhiều đoạn có kích thước bằng nhau từ một đầu(trừ đoạn tại đầu cuối), nói chung mỗi đoan có độ dài bằng độ dài của kết quả hàm băm. Có 2 cách:

- Tách (spliting): tách các đoạn ra, xếp mỗi đoạn một hàng, dóng thẳng theo đầu trái hoặc đầu phải. Ví dụ : khóa 17046329 tách thành 329, 046, 017 cộng lại ta được 392, 392 mod 1000=392 là kết quả băm khóa đã cho.

- Gấp (folding): gấp các đoạn lại theo đường biên tương tự như gấp giấy. Các chữ số cùng nằm tại một vị trí sau khi gấp được xếp lại thẳng hàng nhau rồi có thể cộng lại chia mod để cho kết quả băm. Theo ví dụ trên gấp vào hai biên ta có : 932, 046, 710, cộng lại có 1679, chia mod 1000 được 679 là kết quả băm khóa cho.

Với phương pháp này ta cũng thấy các chữ số của khoá đều được tham gia vào việc tạo nên địa chỉ băm tương ứng với nó.[5]

1.2.4. Hàm băm cho các giá trị khoá là xâu ký tự

Để băm các xâu ký tự, trước hết chúng ta chuyển đổi các xâu ký tự thành các số nguyên. Các ký tự trong bảng mã ASCII gồm 128 ký tự được đánh số từ 0 đến 127, do đó một xâu ký tự có thể xem như một số trong hệ đếm cơ số 128. Áp dụng phương pháp chuyển đổi một số trong hệ đếm bất kỳ sang một số trong hệ đếm cơ số 10, chúng ta sẽ chuyển đổi được một xâu ký tự thành một số nguyên. Chẳng hạn, xâu "NOTE" được chuyển thành một số nguyên sẽ là : $78.128^3 + 79.128^2 + 84.128 + 69$

Vấn đề nảy sinh với cách chuyển đổi này là, chúng ta cần tính các luỹ thừa của 128, với các xâu ký tự tương đối dài, kết quả nhận được sẽ là một số nguyên cực lớn vượt quá khả năng biểu diễn của máy tính.[6]

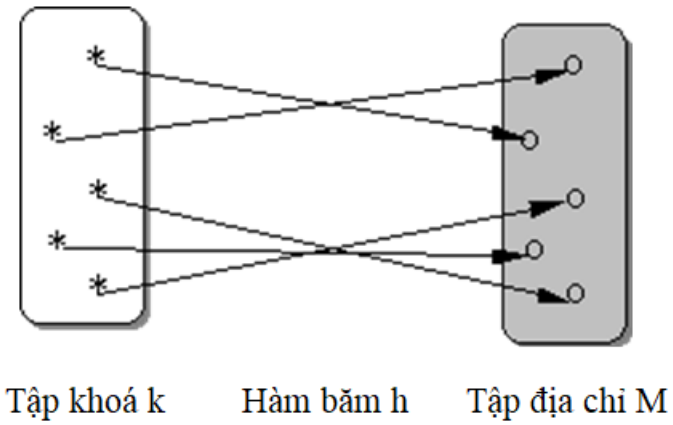
Trong thực tế, thông thường một xâu ký tự được tạo thành từ 26 chữ cái và 10 chữ số, và một vài ký tự khác. Do đó chúng ta thay 128 bởi 37 và tính số nguyên ứng với xâu ký tự theo luật Horner. Chẳng hạn, số nguyên ứng với xâu ký tự "NOTE" được tính như sau: = ((78.37 + 79).37 +84).37 +69

Sau khi chuyển đổi xâu ký tự thành số nguyên bằng phương pháp trên, chúng ta sẽ áp dụng phương pháp chia để tính giá trị băm. Hàm băm các xâu ký tự được cài đặt như sau (Ngôn ngữ C++):

```
unsigned   int   hash(const string &k, int N)
{unsigned  int  value = 0;
        for (int i=0; i< k.length(); i++)
                value = 37 * value + k[i];
        return    value % N;}
```

1.2.5. Bảng băm

1.2.5.1. Mô tả dữ liệu



Hình 1. 2. Mô tả dữ liệu

Giả sử:

k: là tập các khoá (set of keys)

M: tập địa chỉ (set of addresses).

h(k): hàm băm dùng để ánh xạ một khoá k từ tập các khoá k thành một địa chỉ tương ứng trong tập M.

1.2.5.2. Các phép toán trên bảng băm

- Khởi tạo (initialize): khởi tạo bảng băm, cấp phát vùng nhớ hay quy định số phần tử (kích thước) của bảng băm.
- Kiểm tra rỗng (empty): kiểm tra bảng băm có rỗng hay không?
- Tìm kiếm (search): tìm kiếm một phần tử trong bảng băm theo khoá k chỉ định trước.
- Thêm một phần tử mới (insert): thêm một phần tử vào bảng băm. Sau khi thêm số phần tử của bảng băm tăng thêm một đơn vị.

- Loại bỏ(remove): loại bỏ một phần tử khỏi bảng băm, số phần tử sẻ giảm đi một.[7]

1.2.5.3. Các bảng băm thông dụng

Với mỗi loại bảng băm cần thiết phải xác định tập khoá k, xác định tập địa chỉ M và xây dựng hàm băm h cho phù hợp.

a. Bảng băm với phương pháp kết nối trực tiếp (bảng băm dây chuyền): mỗi địa chỉ của bảng băm tương ứng một danh sách liên kết. Các phần tử được kết nối với nhau trên một danh sách liên kết.

b. Bảng băm với phương pháp kết nối hợp nhất: bảng băm này được cài đặt bằng danh sách kề, mỗi phần tử có hai trường: trường key chứa khoá của phần tử và trường next chứa phần tử kế bị xung đột, các phần tử được kết nối với nhau qua trường kết nối next.

c. Bảng băm với phương pháp dò tuần tự: khi thêm phần tử vào bảng băm nếu bị đụng độ thì sẽ dò địa chỉ kế tiếp… cho đến khi gặp địa chỉ trống đầu tiên thì thêm phần tử vào địa chỉ này.

d. Bảng băm với phương pháp dò bậc hai: ví dụ khi thêm phần tử vào bảng băm này, nếu băm lần đầu bị xung đột thì sẽ dò đến địa chỉ mới, ở lần dò thứ i sẽ xét phần tử cách i^2 cho đến khi gặp địa chỉ trống đầu tiên thì thêm phần tử vào địa chỉ này.

e. Bảng băm với phương pháp băm kép: bảng băm này dùng hai hàm băm khác nhau, băm lần đầu với hàm băm thứ nhất nếu bị xung đột thì xét địa chỉ khác bằng hàm băm thứ hai.

f. Bảng băm với phương pháp dò tuyến tính: Nếu băm lần đầu bị xung đột thì băm lại lần một, nếu bị xung đột nữa thì băm lại lần hai,... quá trình băm lại diễn ra cho đến khi không còn xung đột nữa. Các phép băm lại thường sẽ chọn các địa chỉ khác cho các phần tử.[8]

1.3. GIẢI QUYẾT ĐỤNG ĐỘ

Bước đầu tiên của việc tìm kiếm bằng phép băm là tính một hàm băm (hash function) để chuyển đổi từ khoá tìm kiếm vào địa chỉ bảng. Trong trường hợp lý tưởng các khoá khác nhau nên ánh xạ vào các địa chỉ khác nhau nhưng thực tế thì không có hàm băm hoàn chỉnh và sẽ có 2 hay nhiều khoá khác nhau băm đến cùng một địa chỉ. Phần thứ hai của tìm kiếm băm là giải quyết xung đột (collision-resolution) để cư xử với các khoá như đã nói. Trong mục này chúng ta sẽ trình bày hai phương pháp giải quyết va chạm. Trong phương pháp thứ nhất, mỗi khi xảy ra va chạm, chúng ta tiến hành thăm dò để tìm một vị trí còn trống trong bảng và đặt dữ liệu mới vào đó. Một phương pháp khác là, chúng ta tạo ra một cấu trúc dữ liệu lưu giữ tất cả các dữ liệu được băm vào cùng một vị trí trong bảng và "gắn" cấu trúc dữ liệu này vào vị trí đó trong bảng.[9]

1.3.1. Phương pháp địa chỉ mở

Trong phương pháp này, các dữ liệu được lưu trong các thành phần của mảng, mỗi thành phần chỉ chứa được một dữ liệu. Mỗi khi cần xen một dữ liệu mới với khoá k vào mảng, nhưng tại vị trí h(k) đã chứa dữ liệu, chúng ta sẽ tiến hành thăm dò một số vị trí khác trong mảng để tìm ra một vị trí còn trống và đặt dữ liệu mới vào vị trí đó. [10]

Giả sử vị trí mà hàm băm xác định ứng với khoá k là i, i = h(k). Từ vị trí này chúng ta lần lượt xem xét các vị trí:

$$i_0 , i_1 , i_2 ,\ldots, i_m ,\ldots$$

Trong đó $i_0 = i$, i_m (m=0,1,2,…) là vị trí thăm dò ở lần thứ m.

1.3.1.1. Thăm dò tuyến tính

Đây là phương pháp thăm dò đơn giản và dễ cài đặt nhất. Với khoá k, giả sử vị trí được xác định bởi hàm băm là i = h(k), khi đó dãy thăm dò là i , i+1, i+2 , …

Như vậy thăm dò tuyến tính có nghĩa là chúng ta xem xét các vị trí tiếp liền nhau kể từ vị trí ban đầu được xác định bởi hàm băm. Khi cần xen vào một dữ liệu mới với khoá

k, nếu vị trí i = h(k) đã bị chiếm thì ta tìm đến các vị trí đi liền sau đó, gặp vị trí còn trống thì đặt dữ liệu mới vào đó.[11]

Phương pháp thăm dò tuyến tính có ưu điểm là cho phép ta xem xét tất cả các vị trí trong mảng, và do đó phép toán xen vào luôn luôn thực hiện được, trừ khi mảng đầy. Song nhược điểm của phương pháp này là các dữ liệu tập trung thành từng đoạn, trong quá trình xen các dữ liệu mới vào, các đoạn có thể gộp thành đoạn dài hơn. Điều đó làm cho các phép toán kém hiệu quả, chẳng hạn nếu i = h(k) ở đầu một đoạn, để tìm dữ liệu với khoá k chúng ta cần xem xét cả một đoạn dài.

1.3.1.2. Thăm dò bình phương

Để khắc phục tình trạng dữ liệu tích tụ thành từng cụm trong phương pháp thăm dò tuyến tính, chúng ta không thăm dò các vị trí kế tiếp liền nhau, mà thăm dò bỏ chỗ theo một quy luật nào đó. Trong thăm dò bình phương, nếu vị trí ứng với khoá k là i = h(k), thì dãy thăm dò là: i , $i + 1^2$, $i + 2^2$,… , $i + m^2$,…[12]

Ví dụ: Nếu cỡ của mảng N = 11, và i = h(k) = 3, thì thăm dò bình phương cho phép ta tìm đến các địa chỉ 3, 4, 7, 1, 8 và 6.

Phương pháp thăm dò bình phương tránh được sự tích tụ dữ liệu thành từng đoạn và tránh được sự tìm kiếm tuần tự trong các đoạn. Tuy nhiên nhược điểm của nó là không cho phép ta tìm đến tất cả các vị trí trong mảng, chẳng hạn trong ví dụ trên, trong số 11 vị trí từ 0, 1, 2, …, 10, ta chỉ tìm đến các vị trí 3, 4, 7, 1, 8 và 6. Hậu quả là phép toán xen vào có thể không thực hiện được, mặc dù trong mảng vẫn còn các vị trí không chứa dữ liệu. Nếu cỡ của mảng là số nguyên tố, thì thăm dò bình phương cho phép ta tìm đến một nửa số vị trí trong mảng. Cụ thể hơn là, các vị trí thăm dò $h(k) = m^2$ (mode N) với m=0, 1,…, ⌊N/2⌋ là khác nhau. Vậy khẳng định trên chúng ta suy ra rằng, nếu cỡ của mảng là số nguyên tố và mảng không đầy quá 50% thì phép toán xen vào luôn luôn thực hiện.

1.3.1.3. Băm kép

Phương pháp băm kép (double hashing) có ưu điểm như thăm dò bình phương là hạn chế được sự tích tụ dữ liệu thành cụm; băm kép còn cho phép ta thăm dò tới tất cả các vị trí trong mảng. Trong băm kép, chúng ta sử dụng hai hàm băm h_1 và h_2:

- Hàm băm h_1 đóng vai trò như hàm băm h trong các phương pháp trước, nó xác định vị trí thăm dò đầu tiên.
- Hàm băm h_2 xác định bước thăm dò.[13]

Điều đó có nghĩa là, ứng với mỗi khoá k, dãy thăm dò là:

$$h_1(k) + m\, h_2(k), \text{ với } m = 0, 1, 2, \ldots$$

Bởi vì $h_2(k)$ là bước thăm dò, nên hàm băm h_2 phải thoả mãn điều kiện $h_2(k) \neq 0$ với mọi k. Khẳng định trên sẽ đúng nếu chúng ta lựa chọn cỡ của mảng là số nguyên tố.Trong các ứng dụng, chúng ta có thể chọn cỡ mảng N là số nguyên tố và chọn M là số nguyên tố, M < N, rồi sử dụng các hàm băm. [14]

$$h_1(k) = k \,\%\, N$$

$$h_2(k) = 1 + (k \,\%\, M)$$

Ví dụ: Giả sử N = 11, và các hàm băm được xác định như sau:

$$h_1(k) = k \,\%\, 11$$

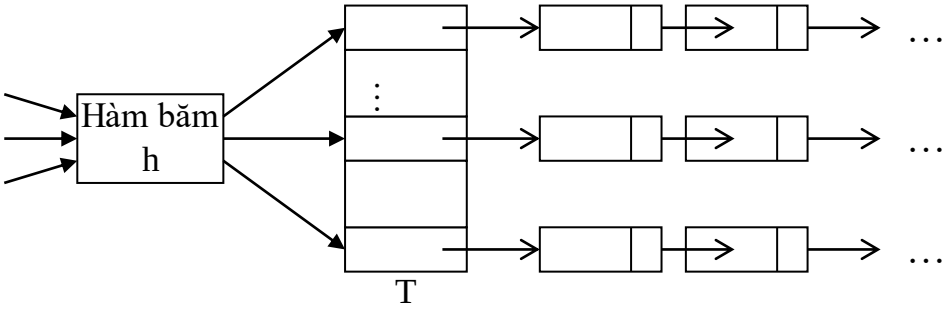
$$h_2(k) = 1 + (k \,\%\, 7)$$

với k = 58, thì bước thăm dò là $h_2(58) = 1 + 2 = 3$, do đó dãy thăm dò là: $h_1(58)$ = 3, 6, 9, 1, 4, 7, 10, 2, 5, 8, 0. còn với k = 36, thì bước thăm dò là $h_2(36) = 1 + 1 = 2$, và dãy thăm dò là 3, 5, 7, 9, 0, 2, 4, 6, 8, 10.

1.3.2. Phương pháp kết nối (danh sách liên kết)

Một cách tiếp cận khác để giải quyết sự va chạm là chúng ta tạo một cấu trúc dữ liệu để lưu tất cả các dữ liệu được băm vào cùng một vị trí trong mảng. Cấu trúc dữ liệu thích hợp nhất là danh sách liên kết (dây chuyền). Khi đó mỗi thành phần trong bảng băm T[i],

với i = 0, 1, …, N – 1, sẽ chứa con trỏ trỏ tới đầu một danh sách liên kết. Cách giải quyết va chạm như trên được gọi là phương pháp tạo dây chuyền (separated chaining). [15]. Lược đồ lưu tập dữ liệu trong bảng băm sử dụng phương pháp tạo dây chuyền được mô tả trong hình 4.



Hình 1. 3. Phương pháp tạo dây chuyền

Ưu điểm của phương pháp giải quyết va chạm này là số dữ liệu được lưu không phụ thuộc vào cỡ của mảng, nó chỉ hạn chế bởi bộ nhớ cấp phát động cho các dây chuyền. Các phép toán được thực hiện rất dễ dàng, để xen vào bảng băm dữ liệu khoá k, chúng ta chỉ cần xen dữ liệu này vào đầu danh sách liên kết được trỏ tới bởi con trỏ T[h(k)]. Phép toán xen vào chỉ đòi hỏi thời gian O(1), nếu thời gian tính giá trị băm h(k) là O(1). Việc tìm kiếm hoặc loại bỏ một dữ liệu với khoá k được quy về tìm kiếm hoặc loại bỏ trên danh sách liên kết T[h(k)]. Thời gian tìm kiếm hoặc loại bỏ đương nhiên là phụ thuộc vào độ dài của danh sách liên kết.

Kết luận rằng giải quyết va chạm bằng cách thăm dò, hay giải quyết va chạm bằng cách tạo dây chuyền, thì bảng băm đều không thuận tiện cho sự thực hiện các phép toán tập động khác, chẳng hạn phép toán Min (tìm dữ liệu có khoá nhỏ nhất), phép toán DeleteMin (loại dữ liệu có khoá nhỏ nhất), hoặc phép duyệt dữ liệu.

CHƯƠNG 2. BÀI TOÁN ỨNG DỤNG

2.1. PHÁT BIỂU BÀI TOÁN VÀ TỔ CHỨC CHƯƠNG TRÌNH.

2.1.1. Phát biểu bài toán

Dùng một thuật toán tìm một từ tiếng anh bất kì trong từ điển, kết quả nhận được là từ loại và nghĩa tiếng việt của từ đã nhập sao cho nhanh nhất.

2.1.2. Phân tích bài toán

Với đồ án môn học lần này, em muốn tạo một phần mềm Từ điển Anh – Việt bằng ngôn ngữ lập trình C++ đạt các yêu cầu sau:

- Chức năng tra từ tiếng anh sang tiếng việt.
- Import dữ liệu từ file với định dạng cho trước.

2.1.3. Tổ chức dữ liệu

❖ ***Input***

Dữ liệu của chương trình này được lưu trữ trên file văn bản chứa các thông tin về các từ tiếng anh và nghĩa tiếng việt.

Thông tin vào của chương trình là từ khoá (từ tiếng anh) muốn tìm kiếm, tra cứu.

❖ ***Output***

Chương trình sẽ cung cấp đầy đủ thông tin về từ loại, nghĩa của từ tương ứng với từ khoá mà người dùng nhập vào (từ tiếng anh) nếu từ khoá đó có trong file lưu trữ.

Ngược lại chương trình sẽ thông báo cho người dùng rằng từ khoá trên không có trong từ điển.

2.1.4. Xác định thuật toán (đọc lại)

Một vấn đề mà người lập trình quan tâm đó là tìm ra thuật toán sao cho tối ưu nhất, có nghĩa là máy chỉ phải thực hiện số bước để tìm ra kết quả là ít nhất và các phép tính sẽ đơn giản nhất. Thời gian tìm kiếm trên bảng băm trong trường hợp tốt nhất là một hằng số

O(1). Chúng ta áp dụng ý tưởng tìm kiếm trên bảng băm kết hợp vào việc truy xuất trên danh sách liên kết bằng cách sử dụng khoá hay một phần của khoá. Kỹ thuật băm là một khái niệm trừu tượng, ít được chú ý, tương đối khó hiểu đối với nhiều người. Chính vì vậy, chúng em lựa chọn bảng băm kết hợp với danh sách liên kết để lưu trữ và sử dụng phương pháp này để tìm kiếm bởi vì sử dụng danh sách liên kết sẽ thuận lợi cho các chức năng khác với mục đích như là một ví dụ giúp mọi người phần nào đó hiểu thêm về kiểu các trúc dữ liệu này.

2.2. GIẢI THUẬT

2.2.1. Giải thuật chính

- Bước 1: Lựa chọn công việc.
- Bước 2:
  - Nếu chọn Tìm kiếm → Tìm kiếm và in kết quả → Bước 1.
  - Nếu chọn Thoát → Kết thúc chương trình.
- Hàm băm được sử dụng: Hàm băm dùng phương pháp chia h(k)=k%M.

2.2.2. Giải thuật cụ thể.

- Bước 1: Nhập từ khoá cần tìm kiếm
- Bước 2: Thực hiện tìm kiếm, nếu tìm thấy in ra kết quả ngược lại thông báo từ đó không có trong từ điển và có thể thêm vào.

Khi tìm một phần tử có khóa k trong cây, hàm băm h(k) sẽ xác định địa chỉ i trong khoảng từ 0 đến M-1 ứng với nhánh i có thể chứa phần tử này. Thời gian tìm kiếm tỉ lệ với số ký tự tạo nên khoá. Nếu đặt n là độ dài của ký tự cần tìm thì với mỗi lần tìm kiếm ta phải mất tối đa là n lần truy xuất để đi xuống n mức. Như vậy thuật toán tìm kiếm độ phức tạp là O(n).

2.2.3. Xây dựng chương trình.

- Các thư viện sử dụng trong chương trình:

  Sử dụng một số thư viện chuẩn: ***windows.h; conio.h; fstream; math.h; iostream***

- Giao diện chính:
  - Tab: Thêm từ mới.
  - Esc: Thoát chương trình.
  - Enter: Xem nghĩa từ.
  - Highlight từ hiện tại.
  - Thay đổi màu text với background.
  - Di chuyển con trỏ.
  - Backspace: Xóa từ.
- Hàm băm:

```
int taptudien(string tu) { //Tim xem tu nay thuoc tap tu dien bam nao
        if(tu != "") {
                char x = tu[0];
                if((x < 123) && (x > 96)) {
                        return x - 97;
                }
        }
        return -1;
}
```

- Các chức năng của từ điển:
  - Tìm kiếm từ

```
NODEWORD* timKiem(hashtable *tudien, string input) {
        int index = tapTuDien(input);
        NODEWORD *p = tudien[index].head;
        while (p != NULL) {
                if (soSanh(input, p)) {
```

```
                return p;
            }
            p = p->right;
        }
        return NULL;
}
```

- Thêm từ

```
void themTuMoi(hashtable *&tudien) {
        system("cls");
        NODEWORD *p = new NODEWORD();
        p->info.first = NULL;
        string temp = "";
        // nhap tu
        cout << "Nhap tu: ";
        getline(cin, temp);
        if (temp == "") return;
        else {
            p->info.tu = temp;
        }
        // nhap loai tu
        temp = "";
        cout << "Nhap loai tu: ";
        getline(cin, temp);
        p->info.loai = temp;
        // nhap toi da 5 nghia
        for (int i = 0; i < NGHIA_TOI_DA; i++) {
            temp = "";
            cout << "Nhap nghia " << i + 1 << ": ";
```

```
            getline(cin, temp);
            if (temp == "") break;
            p->info.nghia[i] = temp;
        }
        int count = 0;
        while (true) {
            temp = "";
            cout << "Nhap vi du " << ++count << ": ";
            getline(cin, temp);
            if (temp == "") break;
            chenVdVaoDauDs(p->info.first, temp);
        };
        chenTuVaoCuoiTd(tudien, p);
        dulieuThaydoi = true;
}
```

- Sửa từ

```
void suaTu(NODEWORD *&p) {
        system("cls");

        cout << "Sua tu: '" << p->info.tu << "':" << endl;

        string temp = "";
        cout << "Sua loai tu (" << p->info.loai << "): ";
        getline(cin, temp);
        if (temp == "") return;
        p->info.loai = temp;
```

```
for (int i = 0; i < NGHIA_TOI_DA; i++) {
	temp = "";
	cout << "Sua nghia " << i + 1 << " (" << p->info.nghia[i] << "): ";
	getline(cin, temp);
	if (temp == "") break;
	p->info.nghia[i] = temp;
}

NODEVIDU *vd = p->info.first;
int count = 0;
if (vd == NULL) {
	// tu nay chua co vi du nao
	while (true) {
		cout << "Them vi du " << ++count << ": ";
		getline(cin, temp);
		if (temp == "") break;
		chenVdVaoDauDs(p->info.first, temp);
	};
}
else {
	// tu nay da co it nhat 1 vi du
	while (vd != NULL) {
		cout << "Sua vi du " << ++count << " (" << vd->info << "): ";
		getline(cin, temp);
		if (temp == "") break;
		vd->info = temp;
		vd = vd->next;
	}
}
```

```
        dulieuThaydoi = true;
}
```

- Xóa từ

```
void xoaTu(hashtable *&tudien, NODEWORD *&p) {
        system("cls");

        cout << "Ban co chac muon xoa tu nay: '" << p->info.tu << "' (y/n)?";
        char c = _getch();
        if (c != 'y') {
                return;
        }

        int index = tapTuDien(p->info.tu);
        NODEWORD* head = tudien[index].head;
        NODEWORD* tail = tudien[index].tail;

        if (head == tail && head->info.tu == p->info.tu) {
                head = tail = NULL;
        }
        else if (head->info.tu == p->info.tu) {
                if (head->right == NULL) {
                        head = tail = NULL;
                }
                head = head->right;
                head->left = NULL;
        }
        else if (tail->info.tu == p->info.tu) {
```

```
            tail = tail->left;
            tail->right = NULL;
        }
        else {
            NODEWORD *q = head;
            do {
                q = q->right;
                if (q->info.tu == p->info.tu) {
                    NODEWORD *r = q->left;// nut ben trai nut q
                    NODEWORD *s = q->right; // nut ben phai nut q
                    q->left = NULL;
                    q->right = NULL;
                    q = NULL;
                    delete q;
                    r->right = s;
                    s->left = r;
                    break;
                }
            } while (true);
        }
        p = NULL;
        delete p;
        tudien[index].head = head;
        tudien[index].tail = tail;
        dulieuThaydoi = true;}
```

CHƯƠNG 3. KẾT LUẬN

3.1. Những kết quả đạt được

Qua quá trình xây dựng chương trình, em đã thực hiện được những yêu cầu đề ra của đề tài là:

- Hình thành một cẩm nang tra cứu từ trong tiếng Anh. Đưa ra một cách tìm kiếm với độ phức tạp khá thấp.
- Sử dụng tốt các chức năng hướng đối tượng của C++ để xây dựng chương trình tương đối tốt, rõ ràng.

Cũng trong quá trình thực hiện đề tài này em đã thu được nhiều kiến thức bổ ích, mở rộng kiến thức của mình về lập trình hướng đối tượng và kinh nghiệm xử lý một số tình huống không như ý muốn trong quá trình lập trình bằng ngôn ngữ C++. Qua đây góp phần tạo tiền đề để sau này có thể đi sâu tìm hiểu ngôn ngữ lập trình C++ và các ngôn ngữ lập trình khác nữa.

3.2. Những hạn chế

Bên cạnh những vấn đề đã làm được, em nhận thấy vẫn còn nhiều vấn đề còn hạn chế:

- Giao diện chưa đẹp.
- Chưa kết hợp nhuần nhuyễn giữa sử dụng chuột và sử dụng bàn phím.

3.3. Hướng phát triển của đề tài.

Hy vọng trong tương lai, có điều kiện tìm hiểu em sẽ hoàn thiện đề tài của mình hơn bằng cách khắc phục những hạn chế mà chương trình hiện nay chưa làm được để có một chương trình hoàn thiện hơn.

TÀI LIỆU THAM KHẢO

[1],[2]: Đỗ Xuân Lôi; *Cấu trúc dữ liệu và giải thuật*; Nhà xuất bản Đại học quốc gia Hà Nội; tái bản năm 2007.

[3],[4],[5]: Nguyễn Văn Linh; *Cấu trúc dữ liệu;* Thư viện học mở Online Việt Nam; năm 2010.

[6]: Lê Khắc Minh Tuệ; *Hash: A string matching Algotrithm;* https://vnoi.info/wiki/algo/string/hash.

[7],[8]: Ngô Quốc Hùng; *Cấu trúc dữ liệu: Bảng băm;* https://sites.google.com/site/ngo2uochung/courses/ctdl-hashtable.

[9],[10],[11],[12],[13],[14],[15]: Giải Thuật Lập Trình; *Cơ chế giải quyết xung đột cơ bản;* http://www.giaithuatlaptrinh.com/?p=967

[16]: Lê Văn Vinh; *Cấu trúc dữ liệu và Giải thuật*; Nhà xuất bản Đại Hoc quốc gia Thành phố Hồ Chí Minh, năm 2013.