ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

BÁO CÁO ĐỒ ÁN I

Sinh viên thực hiện : **Trương Việt Long**

MSSV : 20194105

Mã lớp : 709156

Giảng viên hướng dẫn : **TS. Phạm Đăng Hải**

Hà Nội, tháng 11 năm 2021

PHẦN I

Chương 1: Cài đặt một số thuật toán Sắp xếp

1. Mô tả chương trình

Từ dữ liệu đầu vào là một mảng số nguyên bất kỳ với n phần tử (n xác định), người dùng lựa chọn thuật toán và chương trình xử lý sắp xếp các phần tử trong mảng theo thứ tự tăng dần. Cuối cùng, chương trình đưa ra đầu ra.

Yêu cầu chương trình:

- Đầu vào dữ liệu:
  - Nhập vào từ bàn phím
  - Đọc từ tệp văn bản (.txt)
  - Sinh ngẫu nhiên
- Các thuật toán sử dụng:
  - Sắp xếp Nổi bọt (Bubble Sort)
  - Sắp xếp Lựa chọn (Selection Sort)
  - Sắp xếp Vun đống (Heap Sort)
  - Sắp xếp Nhanh (Quick Sort)
  - Sắp xếp Trộn (Merge Sort)
- Đầu ra dữ liệu:
  - In ra màn hình
  - Ghi ra tệp văn bản (.txt)

2. Các thuật toán Sắp xếp

2.1. Sắp xếp Nổi bọt (Bubble Sort)

Mô tả thuật toán:

- Trong mỗi vòng lặp, duyệt lần lượt các cặp phần tử trong nhóm các phần tử chưa đúng thứ tự. Nếu cặp này ở thứ tự sai (phần tử đứng trước lớn hơn) thì đổi chỗ cho nhau.
- Cuối mỗi vòng lặp, phần tử lớn nhất trong nhóm các phần tử vừa xét sẽ đưa về cuối nhóm, phần tử này đã ở vị trí chính xác. Sau đó tiếp tục vòng lặp tiếp theo cho đến khi tất cả phần tử đã đúng vị trí.

Độ phức tạp thời gian: $O(n^2)$.

Đánh giá:

- Thuật toán ngắn gọn, đơn giản, dễ cài đặt.
- Kém hiệu quả về mặt thời gian.

2.2. Sắp xếp Lựa chọn (Selection Sort)

Mô tả thuật toán:

- Trong mỗi vòng lặp, duyệt lần lượt các phần tử trong nhóm các phần tử chưa đúng thứ tự, tìm ra phần tử có giá trị nhỏ nhất trong nhóm này.
- Cuối vòng lặp, ta đổi chỗ phần tử này cho phần tử đầu tiên trong nhóm, khi đó phần tử này đã ở vị trí chính xác. Sau đó tiếp tục vòng lặp tiếp theo cho đến khi tất cả phần tử đã đúng vị trí.

Độ phức tạp thời gian: $O(n^2)$.

Đánh giá:

- Thuật toán ngắn gọn, đơn giản, dễ cài đặt.
- Giảm số cặp phải đổi vị trí so với sắp xếp Nổi bọt.
- Kém hiệu quả về mặt thời gian.

2.3. Sắp xếp Vun đống (Heap Sort)

Mô tả thuật toán:

- Thuật toán này dựa trên đặc điểm của đống cực đại (max-heap) trong cây nhị phân hoàn thiện (Complete Binary Tree).
    - Max-heap là cây nhị phân mà mỗi nút cha sẽ luôn không nhỏ hơn 2 con của nó. Khi đó giá trị nút lớn nhất sẽ là nút gốc của cây.
    - Với cây nhị phân hoàn thiện, ta đánh số thứ tự từ 0 với các nút tính từ gốc tăng dần theo độ cao, từ trái sang phải. Ở một nút thứ tự `i` bất kỳ, thứ tự nút con trái của nó là (`2i+1`), nút con phải của nó là (`2i+2`). Khi lần lượt lưu trữ các nút vào mảng, chỉ số sẽ tương ứng giá trị trên.
    - Khi một nút có 2 nhánh con là các max-heap, để cây từ nút này trở đi trở thành max-heap, ta lần lượt so sánh nút cha này với 2 con của nó, nếu nút cha có giá trị nhỏ hơn nút con lớn nhất thì đổi chỗ 2 nút cho nhau, sau đó lặp lại việc sắp xếp theo nhánh đó, đến khi nút cha ban đầu thỏa mãn max-heap. Quá trình này sẽ là hàm `heapify()`.
- Đầu tiên, ta biến đổi cây nhị phân thành một max-heap bằng việc lần lượt `heapify()` các nhánh cây nhị phân có độ cao lớn, giảm dần cho tới nút gốc. Giá trị nút lớn nhất sẽ ở gốc.
- Sau đó, lặp lại việc chuyển nút gốc xuống cuối, loại ra khỏi cây nhị phân trong lần lặp tiến theo và `heapify()` từ nút gốc mới.

Độ phức tạp thời gian: $O(n.log n)$.

Đánh giá:

- Hiệu quả cao về mặt thời gian.
- Để cài đặt đòi hỏi kiến thức về thuật toán với Cây nhị phân.

2.4. Sắp xếp Nhanh (Quick Sort)

Mô tả thuật toán

- Đây là một thuật toán chia để trị, chia đôi nhóm cần sắp xếp qua một điểm (`pivot`), bên trái là các giá trị nhỏ hơn pivot, bên phải là các giá trị lớn hơn pivot, sau đó tiếp tục lặp lại với 2 nhóm nhỏ hơn đó.
- Khi kích thước mỗi phía của nhóm cần sắp xếp không lớn hơn 1, nhóm này đã được sắp xếp đúng thứ tự, ta không cần chia đôi nữa.
- Trong cách giải này, quá trình chia đôi các nhóm cần sắp xếp thể hiện trong hàm `partition()`. Chọn `pivot` là phần tử đầu tiên trong nhóm, lần lượt thực hiện các vòng lặp sau:
    - Với các phần tử bên trái (tính từ sau `pivot`), lần lượt dịch một con trỏ (`left`) sang phải cho đến khi gặp phần tử lớn hơn `pivot`.
    - Với các phần tử bên phải, lần lượt dịch một con trỏ khác (`right`) sang trái cho đến khi gặp phần tử nhỏ hơn `pivot`.
    - Phần tử nằm ở hai con trỏ hiện tại nằm nhầm phía, nên ta đổi chỗ hai phần tử này cho nhau.
    - Tiếp tục vòng lặp đến khi 2 con trỏ vượt qua nhau.
    - Sau khi dừng, ta chuyển `pivot` vào vị trí giữa hai phía (đổi chỗ cho phần tử nằm ở con trỏ `right`).

Độ phức tạp thời gian: O(n.*log*n).

Đánh giá:

- Hiệu quả cao về mặt thời gian.
- Để cài đặt đòi hỏi kiến thức về thuật toán.

2.5. Sắp xếp Trộn (Merge Sort)

Mô tả thuật toán

- Đây là thuật toán chia để trị, chia đôi nhóm cần sắp xếp thành 2 nhóm con, sắp xếp riêng 2 nhóm con này theo thuật toán, sau đó trộn 2 nhóm con theo thứ tự.
- Khi nhóm con cần sắp xếp có kích thước là 1, nhóm đã được sắp xếp đúng thứ tự, ta không cần chia đôi nữa.
- Trong cách giải này, quá trình trộn 2 nhóm con đã được sắp xếp thể hiện trong hàm `merge()`:

- Lần lượt ghi giá trị nhỏ hơn trong 2 phần tử đầu của 2 nhóm con vào cha, loại phần tử này khỏi nhóm con và lặp lại bước trên.
- Khi một trong 2 nhóm con đã được ghi hết, ta ghi toàn bộ phần còn lại của nhóm con kia vào cha.

Độ phức tạp thời gian: O(n.*log*n).

Đánh giá:

- Hiệu quả cao về mặt thời gian.
- Để cài đặt đòi hỏi kiến thức về thuật toán.
- Khi thực hiện trộn phải phát sinh thêm bộ nhớ.

3. Kết quả chạy chương trình

Đây là minh họa cho chương trình khi chạy thử:

```
Chon phuong thuc input:
1 - Nhap vao tu Terminal
2 - Doc tu file "input.txt"
3 - Random day so
Khac - Thoat chuong trinh
3

Chon phuong thuc output:
1 - In ra Terminal
2 - Xuat ra file "output.txt"
Khac - Thoat chuong trinh
1

Chon thuat toan sap xep:
1 - Sap xep noi bot (Bubble Sort)
2 - Sap xep lua chon (Selection Sort)
3 - Sap xep vun dong (Heap Sort)
4 - Sap xep nhanh (Quick Sort)
5 - Sap xep tron (Merge Sort)
Khac - Thoat chuong trinh
4

Nhap so phan tu: 6
Day so:
23698  8731 26727 24189 14399 29209
Random day so thanh cong!

Dang tinh toan...

Day so da sap xep:
 8731 14399 23698 24189 26727 29209

Thuat toan voi 6 phan tu chay trong: 0.000000 giay
C:\Users\Long\Programming\IT3150\Phan1\Bai1>
```

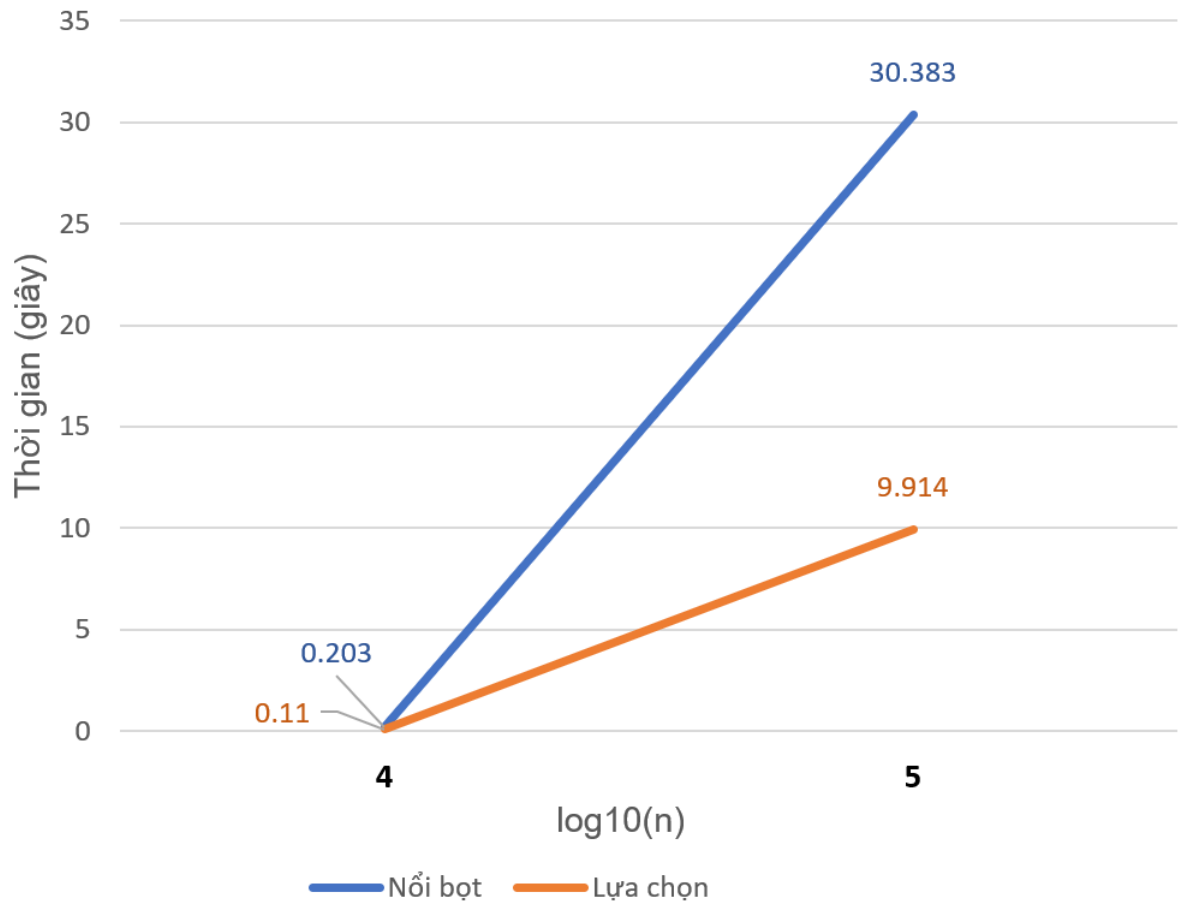
4. Đánh giá thuật toán

Ta sẽ so sánh các thuật toán sắp xếp qua các đồ thị dưới đây:

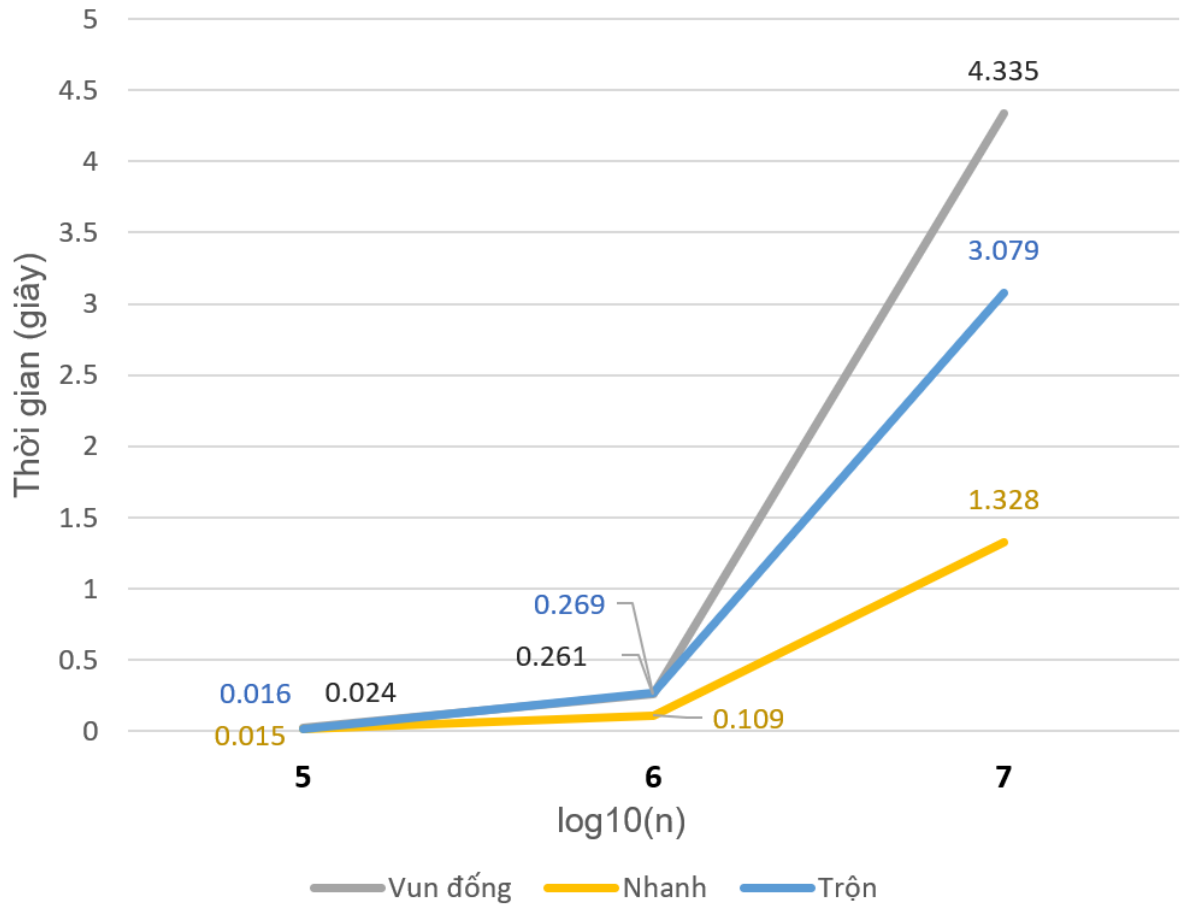
- Trục hoành thể hiện kích thước đầu vào, được đo bằng logarit cơ số 10 của kích thước dữ liệu vào.
- Trục tung thể hiện thời gian thực hiện, đơn vị giây.

Các loại thuật toán có độ phức tạp khác nhau nên thời gian xử lý sẽ có sự chênh lệch rất lớn, vì vậy chia ra làm hai đồ thị mà mỗi đồ thì thuật toán có cùng độ phức tạp. Các thuật toán trong cùng một đồ thị sẽ sử dụng cùng một dữ liệu đầu vào.

Ở đây, ta sẽ chỉ đo duy nhất thời gian thuật toán thực hiện sắp xếp, không đo thời gian nhập xuất dữ liệu. Các kết quả ở đây có tính chính xác phụ thuộc vào máy tính đo.



So sánh giữa sắp xếp Nổi bọt và sắp xếp Lựa chọn với cùng độ phức tạp thời gian $O(n^2)$. Có thể thấy sắp xếp Lựa chọn giảm bớt quá trình đổi chỗ giữa các thành phần so với sắp xếp Nổi bọt nên hiệu quả thời gian tăng dần khi dữ liệu càng lớn.



So sánh giữa 3 thuật toán có cùng độ phức tạp thời gian O(n.*log*n): sắp xếp Vun đống, sắp xếp Nhanh, sắp xếp Trộn. Có thể thấy dựa trên bảng là sắp xếp Nhanh có hiệu quả thời gian cao nhất. Điều này có thể đến do trong các mỗi lần đệ quy, việc xử lý (sử dụng qua hàm phụ trợ) của sắp xếp Nhanh là đơn giản nhất.

Chương 2: Cài đặt thuật toán giải một số bài toán

1. Giải bài toán Mã đi tuần bằng thuật toán Quay lui

1.1. Phát biểu bài toán

Mã đi tuần là bài toán về việc di chuyển một quân mã trên bàn cờ vua (8×8). Quân mã được đặt ở một ô trên một bàn cờ trống nó phải di chuyển theo quy tắc của cờ vua để đi qua mỗi ô trên bàn cờ đúng một lần.

1.2. Mô tả chương trình

Khi bắt đầu, chương trình sẽ yêu cầu người dùng nhập vào một vị trí xuất phát của quân mã. Từ vị trí xuất phát này, thuật toán Quay lui sẽ tìm đến khi ra lời giải đường đi thỏa mãn yêu cầu. Khi đó, chương trình sẽ in ra bàn cờ 8×8, lần lượt hiển thị từng bước đi trong lời giải.

1.3. Thuật toán sử dụng

Bài toán Mã đi tuần sẽ được giải bằng thuật toán Quay lui.

Quay lui là một kĩ thuật thiết kế giải thuật dựa trên đệ quy. Ý tưởng của quay lui là giải quyết vấn đề bắt đầu từ lời giải rỗng và xây dựng dần lời giải bộ phận (partial solution) để ngày càng tiến gần tới lời giải bài toán. Nếu một lời giải bộ phận không thể tiếp tục phát triển, ta sẽ bỏ nó và quay sang xét tiếp các ứng cử viên khác. Bản chất của quay lui là một quá trình tìm kiếm theo chiều sâu (Depth-First Search).

Thuật toán có mã giả tổng quát như sau:

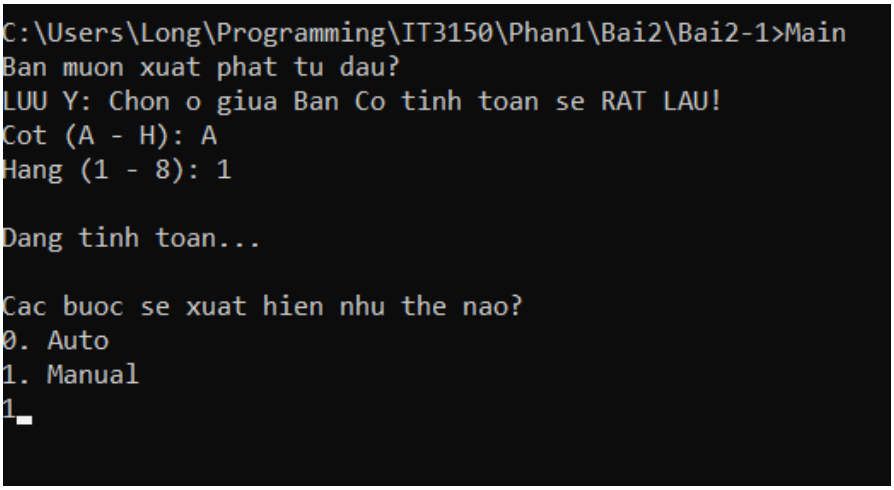
```
Backtracking(k) {
     for([Mỗi phương án chọn i(thuộc tập D)]) {
          if ([Chấp nhận i]) {
               [Chọn i cho X[k]];
               if ([Thành công]) {
                    [Đưa ra kết quả];
               } else {
                    Backtracking(k+1);
                    [Bỏ chọn i cho X[k]];
               }
          }
     }
}
```

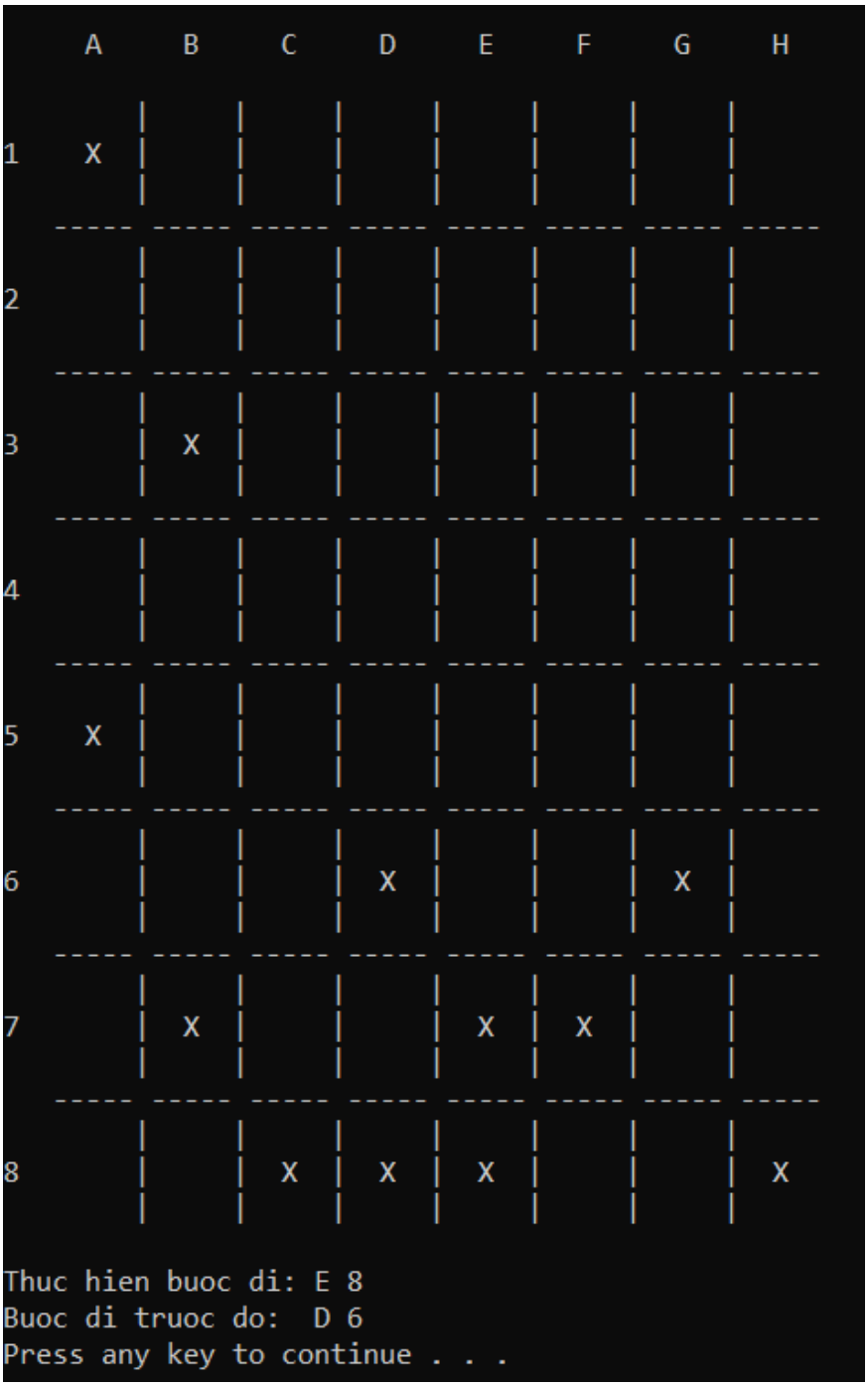
Cụ thể, áp dụng với cách giải bài toán Mã đi tuần:

- Đầu tiên, ta kiểm tra điều kiện thành công (điều kiện dừng) của bài toán là `k = 64`, vì để đi hết mọi ô trong bàn cờ ta chỉ cần dùng 63 nước đi. Khi xét tới bước thứ 64 tức là đã hoàn thành 63 nước trước đó, thuật toán dừng lại và in ra kết quả. Ở đây điều kiện dừng được đưa ra ngoài vòng lặp vì chỉ phụ thuộc vào k chứ không phụ thuộc vào phương án đi.
- Với một bước đi bất kỳ (có số thứ tự `k < 64`), ta sẽ tạo vòng lặp cho 8 phương án di chuyển có thể của quân mã theo luật cờ vua.
- Điều kiện chấp nhận phương án là nước đi tiếp theo nằm trong phạm vi bàn cờ và vị trí này chưa từng đi qua.
- Khi điều kiện thỏa mãn, ta sẽ ghi lại vị trí nước đi này và đánh dấu đã đi qua trên bàn cờ, và tiến tới đệ quy cho bước đi tiếp theo.
- Sau khi thực hiện đệ quy xong mà không đạt được điều kiện dừng, ta xóa nước đi này, trả lại vị trí đã đánh dấu và chuyển sang phương án khác.

1.4. Kết quả chạy chương trình

Đây là minh họa cho chương trình khi chạy thử:





2. Giải bài toán Người du lịch bằng thuật toán Vét cạn và Quay lui nhánh cận.

2.1. Phát biểu bài toán

Một người du lịch muốn đi tham quan n thành phố T_1,T_2,…, T_n. Xuất phát từ một thành phố nào đó, người du lịch muốn đi qua tất cả các thành phố còn lại, mỗi thành phố đi qua duy nhất 1 lần rồi quay trở lại thành phố xuất phát.

Gọi C_{ij} là chi phí đi từ thành phố T_i đến T_j . Hãy tìm một hành trình thỏa yêu cầu bài toán sao cho chi phí là nhỏ nhất.

2.2. Mô tả chương trình

Từ dữ liệu đầu vào là ma trận chi phí với n thành phố, thuật toán sẽ tìm lời giải là một cách di chuyển với chí phí nhỏ nhất. Cuối cùng, chương trình in ra màn hình chi phí nhỏ nhất và lời giải trên.

Yêu cầu chương trình:

- Đầu vào dữ liệu:
    - Nhập vào từ bàn phím
    - Đọc từ tệp văn bản (.txt)
    - Sinh ngẫu nhiên
- Thuật toán sử dụng:
    - Thuật toán Vét cạn (Brute Force)
    - Thuật toán Quay lui nhánh cận (Branch and Bound)

2.3. Thuật toán sử dụng

2.3.1. Thuật toán Vét cạn

Vét cạn là thuật toán dựa trên ý tưởng liệt kê ra mọi phương án có thể của lời giải và xét xem phương án nào thỏa mãn yêu cầu của đề bài.

Với bài toán Người đi du lịch, ta sẽ liệt kê ra mọi phương án hoán vị trong cách di chuyển qua tất cả các thành phố, tính chi phí di chuyển của phương án và so sánh với chi phí tối ưu tại thời điểm đó. Phương án nào cho ra chi phí nhỏ nhất sẽ là lời giải của bài toán.

Ở đây, ta cài đặt thuật toán tìm ra hoán vị tiếp theo của một phương án di chuyển. Xét với một mảng n phần tử chứa giá trị đại diện các thành phố, chỉ số của phần tử là số thứ tự của bước đi đến phần tử đó. Khởi điểm từ phương án giá trị các phần tử tăng dần, tại một phương án bất kỳ, ta có mã giả sau:

```
nextPermutation(){
     [Tìm max(i) mà arr[i] < arr[i+1] ];
     if ([Không tìm thấy i]) [Hoán vị cuối cùng];
     [Tìm max(j) mà arr[j] > arr[i] ];
     swap(arr[i], arr[j]);
     [Đảo ngược dãy sau i];
     return [Tìm thấy hoán vị tiếp theo];
}
```

Các quá trình được mô tả như sau:

- Đầu tiên, ta tìm ra chỉ số lớn nhất trong mảng (đặt là `i`) mà giá trị phần tử tại nó nhỏ hơn giá trị phần tử sau đó.
- Nếu không tìm được chỉ số `i`, tức là giá trị các phần tử được sắp xếp theo thứ tự giảm dần, không thể tìm ra hoán vị tiếp theo. Ở đây, ta không tìm hoán vị với điểm xuất phát, nên khi `i = 0`, ta sẽ dừng tìm hoán vị.
- Ta tìm chỉ số lớn nhất (đặt là `j`) mà giá trị phần tử tại nó lớn hơn giá trị phần tử tại `i`. Sau đó ta đổi chỗ phần tử tại `i` với phần tử tại `j`.
- Từ sau vị trí `i` ở trên, ta đảo ngược lại giá trị phần tử của mảng. Ta đã sinh ra được hoán vị tiếp theo.

Độ phức tạp thời gian: O(n!).

2.3.2. Thuật toán Quay lui nhánh cận

Trong phương pháp Quay lui nhánh cận, về các cài đặt tương đối giống cài đặt thuật toán Quay lui thông thường. Tuy nhiên, trước đi thực hiện đệ quy, thuật toán Quay lui nhánh cận sẽ kiểm tra xem phương án này có khả năng cho ra kết quả tốt hơn kết quả hiện tại hay không. Nếu điều này không thể xảy ra, ta sẽ chuyển sang phương án khác.

Cụ thể, với bài toán Người đi du lịch:

- Ta lưu thêm một giá trị bằng với chi phí nhỏ nhất trong đồ thị di chuyển.
- Đầu tiên, ta kiểm tra điều kiện dừng là `k = n`, vì khi thực hiện bước đi thứ `n`, ta đã đi qua hết `n` thành phố và hiện tại quay trở về điểm xuất phát. Ta sẽ kiểm tra chi phí đã bỏ ra có thật sự nhỏ hơn chi phí tối ưu không. Nếu có, ta cập nhật lại chi phí tối ưu và ghi lại lịch trình.
- Khi `k < n`, ta sẽ lần lượt thử các phương án là các thành phố trong danh sách (trừ thành phố xuất phát), nếu thành phố đã từng đi qua thì chuyển sang phương án tiếp theo.
- Nếu thành phố chưa từng đi qua thì ta sẽ ghi lại, đánh dấu thành phố đã đi và cập nhật chi phí hiện tại.
- Sau đó, ta sẽ kiểm tra xem giả sử toàn bộ bước đi còn lại có chi phí nhỏ nhất, thì tổng chi phí dự kiến có nhỏ hơn chi phí tối ưu hay không, nếu có ta mới tiếp tục đệ quy sang bước di chuyển tiếp theo.
- Cuối cùng, sau quá trình trên, ta xóa đánh dấu, trả lại chi phí và chuyển sang phương án tiếp theo.

2.4. Kết quả chạy chương trình

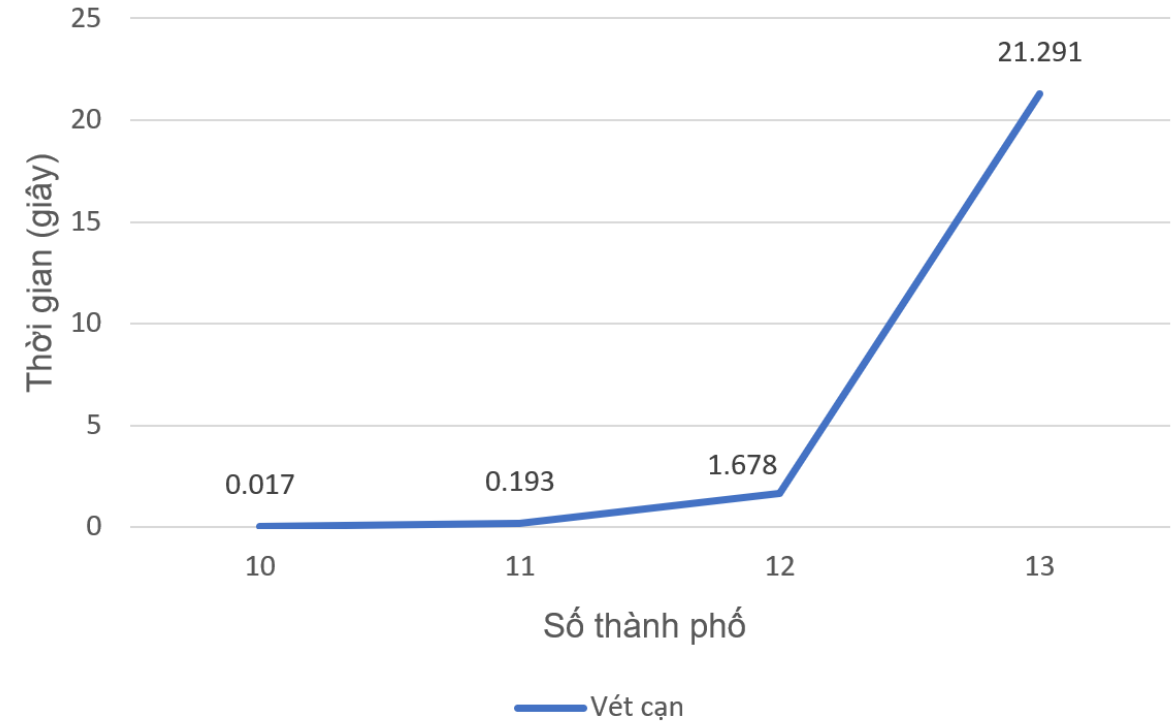
Đây là minh họa cho chương trình khi chạy thử:

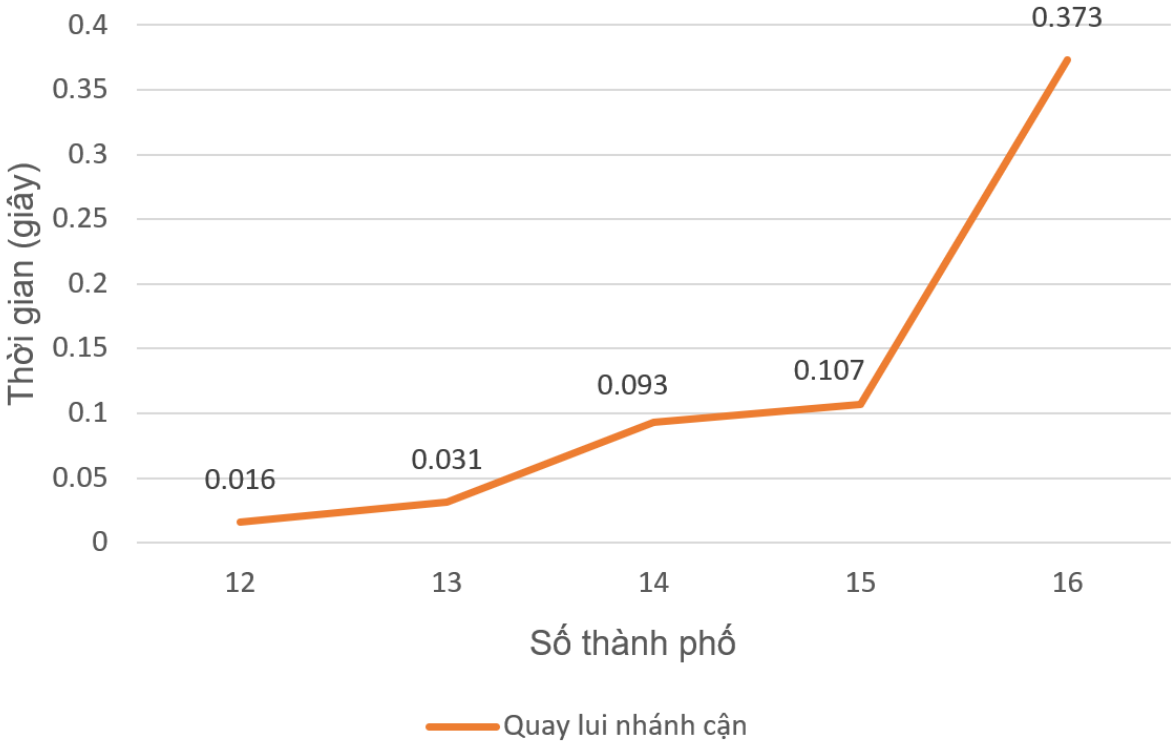
```
Chon phuong thuc input:
1 - Nhap vao tu Terminal
2 - Doc tu file "input.txt"
3 - Random
Khac - Thoat chuong trinh
3

Chon thuat toan:
1 - Vet can (Brute Force)
2 - Nhanh can (Branch and Bound)
Khac - Thoat chuong trinh
2

Nhap so thanh pho: 10
Ma tran chi phi:
     0  28802  31274  30091  17484  14223   6304  30865  29761  32731
 28802      0  11652   3540  24817  15262  24314   3187  26701  26397
 31274  11652      0  13827   8190  29687  16511  18175  24294   1516
 30091   3540  13827      0  12764  17265   9521  30822   8600  22787
 17484  24817   8190  12764      0   1906  29953   7407    458  29177
 14223  15262  29687  17265   1906      0  26396   4304  10698  18503
  6304  24314  16511   9521  29953  26396      0  28034   1263  19901
 30865   3187  18175  30822   7407   4304  28034      0   6341   3914
 29761  26701  24294   8600    458  10698   1263   6341      0   3350
 32731  26397   1516  22787  29177  18503  19901   3914   3350      0

Dang tinh toan...

Chi phi nho nhat: 52643
Cach di chuyen chi phi nho nhat:
0 -> 5 -> 4 -> 2 -> 9 -> 7 -> 1 -> 3 -> 8 -> 6 -> 0

Thuat toan voi 10 thanh pho chay trong: 0.000000000 giay
C:\Users\Long\Programming\IT3150\Phan1\Bai2\Bai2-2>
```

2.5. Đánh giá thuật toán

Ta sẽ biểu diễn thời gian đo theo hai đồ thị dưới đây:

- Trục hoành thể hiện số thành phố (kích thước dữ liệu vào).
- Trục tung thể hiện thời gian thực hiện, đơn vị giây.





Từ hai đồ thị trên, có thể dễ dàng thấy rằng thuật toán Quay lui nhánh cận cho hiệu quả thời gian cao hơn rất nhiều so với thuật toán Vét cạn. Trong bài toán này, thuật toán Vét cạn luôn thực hiện với thời gian tồi nhất là O(n!). Trong khi đó thuật toán Quay lui nhánh cận dù lý thuyết tồi hơn nhưng khi đã cắt bỏ nhiều nhánh lời giải không cần thiết, việc tính toán giảm tải, hiệu quả hơn rất nhiều.

Chương 3: Lập bảng chỉ mục cho file văn bản

1. Đề bài

Đọc một tệp văn bản, hãy lập một bảng chỉ mục (index table) cho tệp văn bản đó.

Bản chỉ dẫn liệt kê tất cả các từ xuất hiện trong văn bản tho quy cách:

- Mỗi từ được liệt kê một lần cùng với số lần xuất hiện trong văn bản và dòng xuất hiện từ đó.
- Các từ phải được sắp xếp theo thứ tự từ điển.

2. Mô tả chương trình

Từ dữ liệu đầu vào là tệp văn bản, chương trình sẽ lần lượt đọc vào từ kí tự, ghép các kí tự chữ cái thành các từ, nếu từ này không có ý nghĩa tra cứu hay từ là danh từ riêng sẽ bị loại bỏ. Các từ còn lại sẽ được lưu trữ trong cấu trúc dữ liệu lựa chọn từ trước và cuối cùng sẽ được in ra màn hình như một bảng chỉ mục.

Các yêu cầu chi tiết:

- Văn bản đầu vào là tiếng Anh, định dạng ASCII, mặc định lưu trong file "vb.txt".
- Từ là dãy chữ cái (A...Z, a...z) liên tiếp nhau, các từ phân biệt nhau bởi khoảng trống, dấu phân cách hoặc các kí tự khác không phải chữ cái. Từ khi đưa vào bảng chỉ mục sẽ chuyển hết tất cả các ký tự thành chữ thường.
- Những từ không đưa vào bảng chỉ mục:
    - Những từ không có ý nghĩa tra cứu (ví dụ: for, the, an,...). Các từ này lưu trong tệp "stopw.txt", mỗi từ một dòng.
    - Những danh từ riêng, là những từ có chữ cái đầu in hoa nhưng không đứng sau dấu chấm câu.
- Trình bày bảng chỉ mục gồm các kết quả theo dòng:
    - Đầu tiên là từ, sau đó là phần dãy số.
    - Số đầu tiên là số lần xuất hiện từ, các số tiếp theo là các dòng mà từ đó xuất hiện.

3. Các cấu trúc dữ liệu sử dụng

3.1. Danh sách liên kết

Danh sách liên kết là tập hợp tuyến tính các phần tử dữ liệu, thứ tự không phụ thuộc địa chỉ vật lý. Mỗi phần tử trong danh sách sẽ chứa một con trỏ chỉ tới địa chỉ phần tử tiếp theo.

Một phần tử trong danh sách liên kết lưu trữ các từ `Node` trong chương trình sẽ bao gồm các thuộc tính:

```
typedef struct Node {
    char keyWord[WORD_SIZE];
    int count;
    NumNode *firstLine, *lastLine;
    struct Node *prev, *next;
} Node;
```

- Xâu `keyWord` lưu trữ nội dung từ.
- `count` là giá trị đếm số lần từ xuất hiện.
- Các con trỏ tới cấu trúc `NumNode` là `firstLine`, `lastLine`. `firstLine` chỉ tới phần tử lưu giá trị dòng đầu tiên mà từ xuất hiện, `lastLine` chỉ tới phần tử lưu giá trị dòng cuối cùng gần nhất mà từ xuất hiện.
- Các con trỏ chỉ tới phần tử trước đó, phần tử tiếp theo trong danh sách liên kết là `prev` và `next`.

Chương trình còn có con trỏ `linkedListHead` chỉ tới phần tử đầu tiên trong danh sách liên kết.

Để lưu các dòng mà từ xuất hiện, ta tạo thêm danh sách liên kết một cấu trúc dữ liệu khác là `NumNode`, phần tử đầu danh sách được trỏ bởi `firstLine`:

```
typedef struct NumberNode {
    int num;
    struct NumberNode *next;
} NumNode;
```

Các phần tử được lưu trong danh sách theo thứ tự từ điển. Khi thêm một từ vào danh sách:

- Ta duyệt lần lượt từ phần tử đầu tiên.
- Nếu gặp phần tử có `keyWord` đứng trước trong từ điển so với từ được thêm, ta chuyển sang phần tử tiếp theo. Ta lặp lại việc trên cho tới khi gặp phần tử có `keyWord` trùng hoặc đứng sau trong từ điển so với từ được thêm.
- Nếu trường hợp trùng, ta sẽ tăng `count` của phần tử này và thêm giá trị dòng mới (nếu có) vào danh sách dòng xuất hiện.

- Nếu trường hợp đứng sau, ta sẽ tạo ra một phần tử mới và chèn vào trước phần tử có `keyWord` đứng sau đang nhắc tới.

3.2. Mảng băm các danh sách liên kết

Băm là cấu trúc dữ liệu sử dụng hàm đặc biệt để ánh xạ một giá trị (value) của một phần tử với một khóa cụ thể (key) để truy cập các phần tử nhanh hơn. Hiệu quả của ánh xạ phụ thuộc vào hiệu quả của hàm băm được sử dụng.

Trong chương trình, ta sử dụng một hàm băm `hashValue()` để tính giá trị key cho một từ. Giá trị này sử dụng làm chỉ số khi thêm từ vào mảng các danh sách liên kết `hashTable`:

- Hàm `hashValue()` là hàm tính tổng giá trị các chữ cái trong từ theo giá trị trong bảng mã ASCII. Sau đó chia lấy dư tổng trên cho một số nguyên xác định trước (mặc định là 100) và trả về số dư tìm được.
- Mảng `hashTable` là mảng lưu các con trỏ trỏ tới đầu các danh sách liên kết. Mỗi phần tử trong cùng một danh sách có cùng giá trị khóa giống nhau và bằng chỉ số con trỏ đầu của danh sách đó.

Khi thêm một từ, ta xác định giá trị băm thông qua hàm hashValue(). Lấy đó làm chỉ số con trỏ đầu danh sách liên kết, ta thêm từ vào danh sách liên kết đó. Cách bước thêm vào danh sách liên kết là tương tự như cấu trúc dữ liệu Danh sách liên kết ở trên.

3.3. Cây nhị phân tìm kiếm

Cây nhị phân tìm kiếm là cấu trúc dữ liệu cây nhị phân (mỗi node có tối đa hai con là con trái và con phải) mà với mỗi node bất kỳ, giá trị tất cả các node thuộc cây con trái (nếu có) đều nhỏ hơn node đang xét; giá trị tất cả các node thuộc cây con phải (nếu có) đều lớn hơn node đang xét.

Một phần tử trong cây nhị phân tìm kiếm `TreeNode` sử dụng trong chương trình sẽ bao gồm các thuộc tính:

```
typedef struct TreeNode {
    char keyWord[WORD_SIZE];
    int count;
    NumNode *firstLine, *lastLine;
    struct TreeNode *parent, *leftChild, *rightChild;
} TreeNode;
```

- `keyWord`, `count`, `firstLine`, `lastLine` tương tự như với `Node` trong danh sách liên kết.
- Các con trỏ chỉ tới các phần tử khác: con trỏ chỉ tới phần tử cha `parent`, chỉ tới phần tử con trái `leftChild`, chỉ tới phần tử con phải `rightChild`.

Chương trình còn có con trỏ `treeRoot` chỉ tới phần tử gốc của cây.

Cây được lưu sao cho khi xét mỗi phần tử, phần tử thuộc cây con trái sẽ có từ đứng trước theo thứ tự từ điển so với từ trong phần tử đang xét; phần tử thuộc cây con phải sẽ có từ đứng sau theo thứ tự từ điển so với từ trong phần tử đang xét.

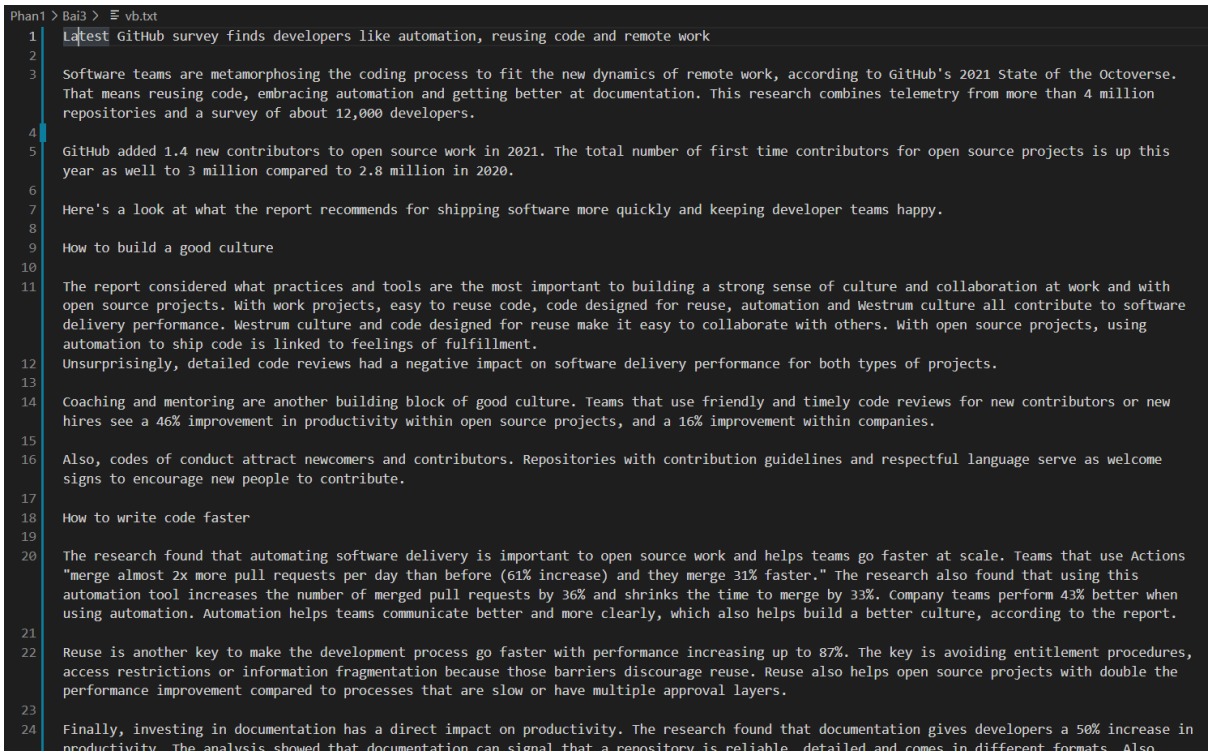
Khi thêm một từ vào cây:

- Ta xuất phát từ phần tử gốc của cây.
- Tại mỗi phần tử, ta so sánh thứ tự của từ thêm vào cây với keyWord của phần tử đang xét:
    - Nếu từ đứng trước so với keyWord của phần tử, ta chuyển sang phần tử con trái của phần tử đang xét.
    - Nếu từ đứng sau so với keyWord của phần tử, ta chuyển sang phần tử con phải của phần tử đang xét.
    - Nếu trùng nhau, ta sẽ tăng `count` của phần tử này và thêm giá trị dòng mới (nếu có) vào danh sách dòng xuất hiện. Ta dừng việc thêm từ này tại đây
- Lặp lại việc trên cho tới khi ta gặp phần tử lá của cây. Khi đó ta sẽ tạo phần tử mới cho từ và thêm vào con trái hoặc con phải của lá này theo thứ tự từ điển. Phần tử mới sẽ là một lá mới.

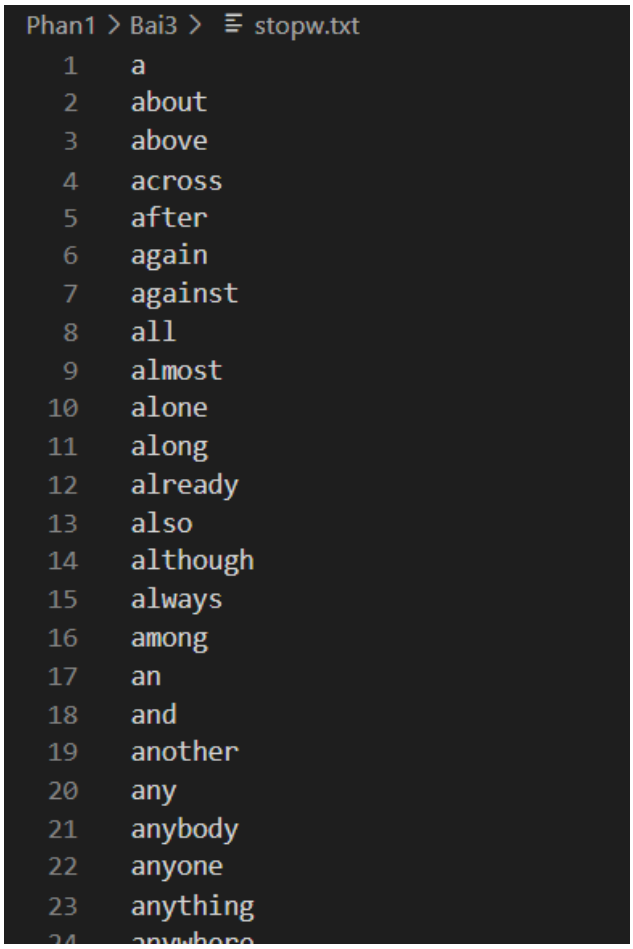
4. Kết quả chạy chương trình

(Các hình ảnh không thể chứa toàn bộ nội dung)

Ví dụ “vb.txt” chứa bài viết sau:



Và “stopw.txt” chứa các từ:



Chương trình sau khi thực hiện với Danh sách liên kết sẽ cho ra:

```
Chon file van ban:
1 - Doc tu file "vb.txt"
2 - Doc tu file khac
1

Chon cau truc du lieu luu tru:
1 - Danh sach lien ket
2 - Mang bam
3 - Cay nhi phan tim kiem
Khac - Thoat chuong trinh
1

Danh sach tu:
access                          1, 22
according                       2, 3, 20
added                           1, 5
analysis                        1, 24
approval                        1, 22
attract                         1, 16
authors                         1, 30
automating                      1, 20
automation                      7, 1, 3, 11, 20
avoiding                        1, 22
barriers                        1, 22
becoming                        1, 26
block                           1, 14
build                           2, 9, 20
building                        2, 11, 14
coaching                        1, 14
code                            10, 1, 3, 11, 12, 14, 18, 30
codes                           1, 16
coding                          1, 3
collaborate                     1, 11
collaboration                   1, 11
collocated                      1, 28
combines                        1, 3
comes                           1, 24
commitments                     1, 28
```