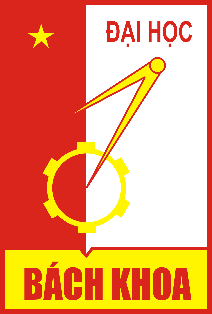
**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

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

──────── \* ───────



BÁO CÁO MÔN HỌC

**PROJECT I**

**Tên đề tài : Các bài toán về sắp xếp, đệ qui, danh sách liên kết**

Sinh viên thực hiện: **HOÀNG MẠNH HIỆP**

**Lớp : CNTT2.02**

Giảng viên hướng dẫn: **PHẠM ĐĂNG HẢI**

**HÀ NỘI 10-2018**

**MỤC LỤC**

[Mục lục](#_Toc512211602) 2

[Tài liệu tham khảo 4](#_Toc512211603)

Phần I: Mở đầu 5

[Lời mở đầu 5](#_Toc512211603)

Phần II: Nội dung 6

[II.1. Bài toán 1 6](#_Toc512211605)

[1.Đề bài 6](#_Toc512211606)

[Bubble sort 7](#_Toc512211611)

[Quicksort 8](#_Toc512211611)

[Heapsort 9](#_Toc512211611)

[Mergesort 10](#_Toc512211611)

[So sánh các thuật toán sắp xếp 11](#_Toc512211611)

[Đa luồng trong sắp xếp 12](#_Toc512211611)

[II.2. Bài toán 2 12](#_Toc512211610)

[1. Đề bài 12](#_Toc512211611)

[Bài toán quân hậu 12](#_Toc512211611)

[1. Ý tưởng bài toán 12](#_Toc512211612)

[2. Giải thuật và mã giả 13](#_Toc512211613)

[3.Độ phức tạp của thuật toán 13](#_Toc512211614)

[Bài toán người du lịch vét cạn 13](#_Toc512211611)

[1. Ý tưởng bài toán 13](#_Toc512211612)

[2. Giải thuật và mã giả 13](#_Toc512211613)

[3.Độ phức tạp của thuật toán 13](#_Toc512211614)

[Bài toán người du lịch nhánh cận 14](#_Toc512211611)

[1. Ý tưởng bài toán 15](#_Toc512211612)

[2. Giải thuật và mã giả 16](#_Toc512211613)

[3.Độ phức tạp của thuật toán 16](#_Toc512211614)

[II.3. Bài toán 3 15](#_Toc512211615)

[Đề bài 16](#_Toc512211616)

[1.Ý tưởng bài toán 16](#_Toc512211617)

[2.Giải thuật và mã giả 17](#_Toc512211618)

[3.Độ phức tạp của thuật toán 17](#_Toc512211619)

[Phần III : Kết luận 17](#_Toc512211620)

[1.Kết quả 18](#_Toc512211621)

[2.Kiến thức thu được sau khi làm bài tập 18](#_Toc512211624)

**Danh mục tài liệu tham khảo**

1. Giáo trình cấu trúc dữ liệu và giải thuật, Nguyễn Đức Nghĩa

**PHẦN I: MỞ ĐẦU**

**Lời mở đầu**

Trên thực tế không có sự thành công nào mà không gắn liền với những sự hỗ trợ, giúp đỡ dù ít hay nhiều, dù trực tiếp hay gián tiếp của người khác. Trong suốt thời gian từ khi bắt đầu học tập ở giảng đường đại học đến nay, em đã nhận được rất nhiều sự quan tâm, giúp đỡ của quý thầy cô, gia đình và bạn bè. Với lòng biết ơn sâu sắc nhất, em xin gửi lời cảm ơn đến thầy Phạm Đăng Hải, Giảng viên Khoa Công nghệ Thông tin Trường Đại học Bách Khoa Hà Nội đã giúp đỡ nhóm em hoàn thành tốt bài tập này.

Trong quá trình hoàn thành bài tập, cũng như là trong quá trình làm bài báo cáo khó tránh khỏi sai sót, rất mong các thầy, cô bỏ qua. Đồng thời do trình độ lý luận cũng như kinh nghiệm thực tiễn còn hạn chế nên bài báo cáo không thể tránh khỏi những thiếu sót, em rất mong nhận được ý kiến đóng góp thầy để em học thêm được nhiều kinh nghiệm và sẽ hoàn thành tốt hơn bài báo cáo những lần sau.

***Hà Nội, tháng 10 - 2018***

***Sinh viên thực hiện***

***Hoàng Mạnh Hiệp***

**PHẦN II: NỘI DUNG**

II.1. Bài toán 1: Các bài toán sắp xếp

* Đề bài:

Input: Nhập vào dãy số ngẫu nhiên ( từ bàn phím, file ).

Output: Xuất ra dãy số theo thứ tự không giảm.( In ra màn hình hoặc in ra file) bằng cái thuật toán Bubble sort, Quicksort, Heapsort, Merge sort, ...

Cài đặt thuật toán sắp xếp theo mô hình đa luồng? Đánh giá thời gian thực hiện.

* Bubble sort

1. Ý tưởng thuật toán:

* Ý tưởng chính của thuật toán này là thực hiện phép so sánh hai phần tử cạnh nhau.
* Xuất phát từ phần tử đầu danh sách duyệt đến cuối mảng, ta tiến hành so sánh với phần tử bên phải.
* Nếu phần tử đang xét lớn hơn phần tử bên phải ta tiến hành đổi chỗ hai phần tử. Tiếp tục thực hiện như thế đối với dãy có n phần tử sau n-1 bước ta sẽ có một dãy không giảm.

1. Giải thuật :

Xét mảng a[n]:

Bước 1: Bắt đầu từ i = 0, j=0 ( Phần tử đầu tiên)

Bước 2: Lần lượt so sánh và đổi chỗ từ trái sang phải đối với các phần tử j=0 đến j= n-i-1 nếu a[j] > a[j+1] thì đổi chỗ hai phần tử này với nhau.

Bước 3: Sau đó tiếp tục tăng vòng lặp lên với i = i + 1.

Bước 4: Nếu i < n thì quay lại bước 2. Ngược lại kết thúc vòng lặp, ta được dãy đã sắp xếp theo thứ tự không giảm.

1. Đánh giá thuật toán:

Sắp xếp nổi bọt là một thuật toán đơn giản, nhưng tính hiệu quả của nó không cao. Được đánh giá là kém hiệu quả nhất. Chính vì thế mà sắp xếp nổi bọt rất ít khi được sử dụng.

Thuật toán có độ phức tạp là Θ(), do phải thực hiện hai vòng lặp lồng nhau.

Trong trường hợp tồi nhất thuật toán mất phép so sánh và đổi chỗ.

Trong trường hợp tốt nhất thuật toán cũng phải mất phép so sánh và đổi chỗ.

Trung bình thuật toán mất phép đổi chỗ và phép so sánh.

* Quick sort

1. Ý tưởng thuật toán:

* Ý tưởng chính của quicksort là chia mảng làm hai phần bằng cách so sánh từng phần tử trong mảng với một phân tử gọi là phần tử chốt.
* Việc chọn phần tử chốt có ảnh hưởng rất lớn đến thời gian chạy thuật toán.
* Các phần tử mảng nhỏ hơn hoặc bằng phần tử chốt sẽ được chia về bên trái. Tương tự với các phần tử lớn hơn phần tử chốt sẽ được đưa về bên phải phân tử chốt. Quá trình phân chia này diễn ra cho đến khi độ dài các mảng con bằng 1.

1. Giải thuật:

* Cách chọn phần tử chốt:

Chọn phần tử đứng đầu hoặc đứng cuối làm phẩn tử chốt

Chọn phần tử giữa dãy làm phần tử chốt

Chọn ngẫu nhiên một phần tử trong dãy làm phần tử chốt.

* Các bước thực hiện:

Bước 1: Chọn phần tử chốt

Bước 2: Khai báo 2 biến để duyệt về 2 phía của phần tử chốt

Bước 3: Một biến duyệt các phần tử bên trái và biến còn lại duyệt các phần tử bên phải phần tử chốt.

Bước 4: Khi biến bên trái lớn hơn phần tử chốt thì di chuyển sang phải. Biến bên phải nhỏ hơn phần tử chốt thì di chuyển sang trái.

Bước 5: Tiếp tục làm như vậy cho đến khi được dãy có độ dài là 1. Đến đây ta được dãy đã sắp xếp theo thứ tự không giảm

1. Đánh giá thuật toán :

* Quick sort là thuật toán nhanh nhất hiện nay, có thời gian tính trung bình là O(nlogn).
* Quick sort là thuật toán sắp xếp tại chỗ và không có tính ổn định cao do phụ thuộc nhiều vào cách chọn phần tử chốt.
* Nhược điểm của thuật toán này là đối với dãy đã sắp xếp sẵn khi đó ta phải mất n lần gọi đệ qui và mỗi lần gọi đệ qui chỉ loại được một phần tử. Thời gian thực hiện thuật toán trong trường hợp này là O().
* Quick sort là thuật toán được phát triển dựa trên kĩ thuật chia để trị. Thời gian tính của thuật toán phụ thuộc vào cách chọn phần tử nào là phần tử trụ. Ta có 3 cách phân đoạn:
* Phân đoạn không cân bằng: bài toán con có kích thước n-1 và bài toán kia có kích thước là 0.
* Phân đoạn hoàn hảo: việc phân đoạn luôn được thực hiện dưới dạng phân đôi,mỗi bài toán con có kích thước là n/2.
* Phân đoạn cân bằng: việc phân đoạn có sự chênh lệch không nhiều lắm giữa hai bài toán con.
* Ta tính thời gian trung bình của thuật toán:

Xét phần tử chốt ngẫu nhiên, thời gian tính trung bình (thời gian phân đoạn kích thước i) \* ( xác suất phân đoạn có kích thước i)

Trong trường hợp ngẫu nhiên thì tất cả kích thước đồng khả năng nên xác suất bằng 1/n. Do đó ta có:

T(n) = T(i) + T(n – i – 1) + cn

Giải công thức đệ qui ta có:

E(T(n)) = O(nlogn).

* Heapsort

1. Ý tưởng thuật toán:

* Heapsort dựa trên một cấu trúc dữ liệu gọi là đống nhị phân, gọi đơn giản là đống nó tương tự như Selection Sort.
* Các phần tử trong dãy sẽ được tổ chức dưới dạng cây.
* Trước tiên cần xây dựng đống sao cho với mọi nút cha đều lớn hơn nút con. Khi đó, nút gốc đã là nút lớn nhất. Tiến hành đổi chỗ phần tử ở nút gốc và nút thứ n-1.
* Phần tử lớn nhất sẽ được xếp vào cuối dãy số và loại bỏ khỏi đống. Lặp lại các bước cho ( n-2 ) phần tử còn lại của dãy số.

1. Giải thuật:

Ta xét mảng a[0…n-1]

* Cách biểu diễn mảng a[0..n-1] dưới dạng đống:

a[0] ở vị trí gốc, lần lượt sau đó các phần tử a[2i + 1] sẽ được xếp vào bên trái nút i và a[2\*i + 2] sẽ ở bên phải.

* Cách vun đống:

Việc sắp xếp các phần tử ban đầu trở thành đống gọi là vun đống. Có hai kiểu vun đống: vun đống max – heap(phần tử lớn nhất ở gốc) với mọi nút i thì nút cha luôn lớn hơn các nút con, vun đống min – heap(phần tử nhỏ nhất ở gốc) với mọi nút i thì nút cha nhỏ hơn nút con.

Trong bài này ta xét vun đống max – heap:

Nếu hai cây con gốc thứ 2\*i +1 và (2\*i + 2) đã là đống thì cây con gốc i trở thành đống chỉ cần so sánh a[i] với giá trị lớn hơn trong a[2\*i +1] và a[2\*i + 2]. Nếu a[i] nhỏ hơn thì đổi chỗ chúng cho nhau. Tiếp tục so sánh cho đến khi gặp nút lá.

* Sắp xếp bằng vun đống:

Ta sẽ vun mảng a[0..n-1] từ dưới lên bắt đầu từ phần tử a[j] = n/2 -1 ngược lên đến phần tử a[0].

Bước 1: Sau khi mảng a[1..n-1] đã là đống, lấy phần tử a[0] trên đỉnh của đống ra khỏi đống đặt vào vị trí cuối cùng n-1, và chuyển phần tử thứ cuối cùng a[n-1] lên đỉnh đống thì phần tử a[n-1] đã được đứng đúng vị trí.

Bước 2: Vun lại phần còn lại của mảng với n-2 phần tử.

Bước 3: Tiếp tục lặp cho đến khi mảng chỉ còn lại một phần tử.

1. Đánh giá thuật toán:

* Thời gian tính của thuật toán Max – Heap:

Từ nút i bất kì phải di chuyển theo đường đi xuống phía dưới của cây. Độ dài của đường đi này sẽ không thể vượt quá chiều cao của cây là h.

Mỗi mức ta phải thực hiện hai phép so sánh nên số phép so sánh sẽ không quá 2h.

* Thời gian tính là O(h) hay O(logn)
* Thời gian tính của thuật toán Heap – sort trung bình là O(nlogn).
* Merge Sort

1. Ý tưởng thuật toán :

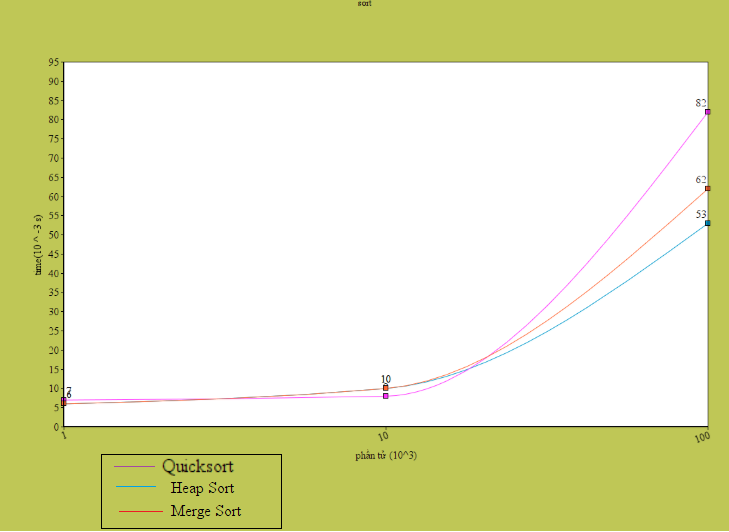
* Ý tưởng của thuật toán này bắt nguồn từ việc trộn hai danh sách đã sắp xếp thành một danh sách mới cũng được sắp xếp.
* Merge Sort là một thuật toán sắp xếp các danh sách theo kiểu sắp xếp so sánh, được phát triển dựa trên chia để trị.
* Ta sẽ chia mảng lớn thành những mảng con nhỏ hơn bằng cách chia đôi mảng lớn và chúng ta tiếp tục chia đôi các mảng con cho tới khi mảng con nhỏ nhất chỉ còn 1 phần tử.

1. Giải thuật :

* Ta xét mảng a[n], bắt đầu bởi left và kết thúc bởi right left, mid, right với mid = (left + right)/2.
* Lần lượt chia nhỏ mảng này cho đến khi được mảng con nhỏ nhất chỉ còn một phần tử a[left , right] 🡪 a[left , mid] và a[mid +1, right], …
* Tiến hành trộn hai mảng con có một phần tử ta được mảng con đã được sắp xếp có 2 phần tử.
* Tiếp tục trộn các cặp mảng con đã sắp xếp gồm hai phần tử để được mảng con có 4 phần tử đã được sắp xếp.
* Tiến hành làm như vậy cho đến khi toàn bộ mảng được sắp xếp.

1. Đánh giá thuật toán :

* Merge Sort là thuật toán sắp xếp có thời gian thực hiện là O(nlogn) trong mọi trường hợp, do đó với dữ liệu lớn và cần ít thao tác sắp xếp thì thuật toán này sẽ tối ưu hơn Quicksort.
* Tuy nhiên nó cũng có những nhược điểm đó là khó cài đặt và không nhận dạng được mảng đã sắp xếp.
* So sánh các thuật toán sắp xếp:
* Đối với các dữ liệu nhỏ, số lượng phần tử 1.000 hay 10.000 thì thời gian tính của các thuật gần như nhau.
* Đối với các dữ liệu lớn tầm 100.000 hay 1.000.000 phần tử thì ta thấy sự khác biệt giữa thời gian thực hiện giữa các thuật toán.
* Cụ thể, thời gian sắp xếp giữa các thuật toán được thể hiện trong biểu đồ sau:



* Từ biểu đồ trên, ta thấy thời gian tính toán của thuật toán quicksort là chậm hơn so với mergesort và heapsort. Do việc chọn phần tử chốt trong quicksort ảnh hưởng nhiều đến thuật toán này.
* Thuật toán heapsort có thời gian chạy nhanh nhất, tiếp theo đó là mergesort.
* Đa luồng trong sắp xếp:
* Ta tạo ra các luồng trong C++ bằng thư viện #include<thread>
* Lần lượt chia mảng thành các phần, mỗi phần là 1 luồng: t1,t2,… cho các luồng này chạy bằng câu lệnh t1.join(), t2.join(),… tiến hành sắp xếp mảng với các luồng chạy song song nhau.
* Sau khi sắp xếp các phần của mảng trong các luồng ta tiến hành trộn lại mảng. Lần lượt gộp các mảng nhỏ lại với nhau bằng thuật toán theo thuật toán sắp xếp trộn, lần lượt so sánh các phần tử giữa hai mảng con đưa phần tử nhỏ hơn vào mảng kết quả, tiếp tục quá trình này cho đến khi một trong hai mảng con rỗng, khi một mảng rỗng ta lấy phần còn lại của mảng kia cho vào cuối mảng kết quả.
* Đánh giá thời gian thực hiện :

Khi tiến hành sắp xếp bằng đa luồng thì thời gian thực hiện thuật toán giảm đi đáng kể. Đặc biệt, với dữ liệu đầu vào lớn tầm 10^6, 10^7,.. phần tử. Việc này giúp tiết kiệm thời gian và tài nguyên hệ thống hơn nhờ các công việc chay song song nhau.

II.2**.** Bài toán 2 : Backtracking

* Đề bài **:**
* Bài toán quân hậu:
* Tìm vị trí các quân hậu trên bàn cờ nx n( với n > 3) sao cho chúng không ăn được lẫn nhau.
* Yêu cầu : Chương trình chạy được, dữ liệu in ra trực quan.
* Sử dụng thuật toán đệ qui quay lui.
* Bài toán người du lịch :
* Input : Ma trận chi phí nhập vào từ file, bàn phím.
* Output: Xuất ra màn hình hoặc ra file đường đi ngắn nhất.
* Sử dụng thuật toán vét cạn và nhánh cận
* Bài toán quân hậu **:**

1. Ý tưởng thuật toán :

* Bài toán sử dụng thuật toán quay lui thử đến khi tìm được lời giải.
* Với n < 4 và n != 1 thì bài toán không có lời giải
* Với n = 1 thì bài toán có một lời giải duy nhất.
* Mấu chốt là việc tìm kiếm vị trí đặt quân hậu tiếp theo ở vị trí cột nào.
* Do các quân hậu không được ăn nhau nên các quân hậu này không thể nằm trên cùng một đường( đường này có thể là hàng, cột, đường chéo). Do đó mỗi hàng chỉ có thể chứa 1 quân hậu.

1. Giải thuật:

* Ta xét mảng 1 chiều a[n] để lưu lại vị trí các quân hậu hợp lệ. Với a[n] chính là tọa độ vị trí của cột trên hàng n.
* Trước tiên ta cần một hàm check() để kiểm tra các quân hậu với điều kiện không ăn được nhau. Tức là không nằm trên cùng một hàng, cùng một cột và không nằm trên cùng một đường chéo.
* Do các quân hậu không ăn được nhau nên mỗi hàng chỉ có thể chứa 1 quân hậu.
* Đầu tiên thử vị trí cho quân hậu ở hàng 1 vào các cột( n cách đặt). Với mỗi vị trí ở hàng 1 này, ta tiến hành:
* Thử đặt quân hậu ở hàng 2 vào các cột sao cho không bị các quân hậu ở hàng 1 khống chế.
* Tương tự với mỗi vị trí của quân hậu ở hàng 2, ta lại tiếp tục thử với quân hậu ở hàng 3.
* Tiếp tục làm như vậy cho đến quân hậu ở hàng n. Tới đây ta đã thu được kết quả của bài toán.
* Khi đã làm đến quân hậu thứ n ta sẽ in ra kết quả của bài toán.

1. Đánh giá thuật toán:

* Thuật toán quay lui là một kĩ thuật thiết kế giải thuật dựa trên đệ qui. Ý tưởng của quay lui là tìm lời giải ở từng bước, mỗi bước chọn một trong số các lựa chọn phù hợp tiến hành đệ qui.
* Thuật toán này có nhược điểm là thực hiện các công việc dư thừa, mỗi lần chúng ta quay lui cần phải đánh giá lại lời giải trong khi nhiều lúc là không cần thiết.
* Độ phức tạp là O(n!). Tuy nhiên nếu chỉ cần một lời giải bất kì thì độ phức tạp chỉ là O(n).
* Bài toán người du lịch vét cạn :

1. Ý tưởng thuật toán :

* Bài toán sử dụng thuật toán quay lui thử đến khi tìm được lời giải.
* Cố định một thành phố xuất phát bất kì, cần tìm đường đi sao cho mỗi thành phố đi qua một lần( ngoại trừ thành phố xuất phát) và kết thúc ở thành phố xuất phát sao cho chi phí nhỏ nhất.

1. Giải thuật:

* Ta xét ma trận chi phí c[n][n] với đầu vào từ file dòng đầu là n thành phố,các dòng tiếp theo là một ma trận n x n ( n là số thành phố ), khởi tạo giá trị chi phí tốt nhất của hành trình f\_best = 9999999999.
* Trước tiên ta cần một hàm check() để kiểm tra các thành phố đã được đi qua chưa.
* Cố định thành phố xuất phát là thành phố đầu tiên. Hành trình cần tìm sẽ có dạng a[0…n] với a[0] = a[n] và giữa 2 thành phố bất kì phải có đường đi tức là c[i][j] ≠ +∞.
* Tiến hành duyệt quay lui:
* Với mỗi thành phố a[1] có thể chọn mà a[0] có đường đi tới trực tiếp.
* Tương tự với mỗi cách thử chọn của a[2] thì a[1] có đường đi tới, với mỗi thành phố a[i] thì có thể chọn các thành phố mà nó chưa đi qua và có đường đi trực tiếp đến.
* Tiếp tục làm như vậy cho đến khi tìm được hành trình hoàn chỉnh, tiến hành so sánh với giá trị tốt nhất(f\_best) của các hành trình trước đó nếu nhỏ hơn thì cập nhật lại f\_best.

1. Đánh giá thuật toán :

* Thuật toán quay lui là một kĩ thuật thiết kế giải thuật dựa trên đệ qui. Ý tưởng của quay lui là tìm lời giải ở từng bước, mỗi bước chọn một trong số các lựa chọn phù hợp tiến hành đệ qui.
* Thuật toán này có nhược điểm là thực hiện các công việc dư thừa, với các hành trình có chi phí lớn hơn giá trị tối ưu mà chưa hoàn chỉnh thì nó vẫn xét tiếp các nhánh này.
* Bài toán người du lịch nhánh và cận :

1. Ý tưởng thuật toán :

* Bài toán sử dụng thuật toán quay lui thử đến khi tìm được lời giải.
* Cố định một thành phố xuất phát bất kì, cần tìm đường đi sao cho mỗi thành phố đi qua một lần( ngoại trừ thành phố xuất phát) và kết thúc ở thành phố xuất phát sao cho chi phí nhỏ nhất.
* Sử dụng thuật toán nhánh cận giúp ta đánh giá cận chi phí của các hành trình từ đó rút bỏ những bước đi không cần thiết.

1. Giải thuật:

* Ta xét ma trận chi phí c[n][n] với đầu vào từ file dòng đầu là n thành phố, các dòng tiếp theo là một ma trận n x n ( n là số thành phố ), khởi tạo giá trị chi phí tốt nhất của hành trình f\_best = 9999999999, cận trên của bài toán g = f + (n – i +1) \* Cmin, với f là chi phí đi được i thành phố.
* Trước tiên ta cần một hàm check() để kiểm tra các thành phố đã được đi qua chưa.
* Cố định thành phố xuất phát là thành phố đầu tiên. Hành trình cần tìm sẽ có dạng a[0…n] với a[0] = a[n] và giữa 2 thành phố bất kì phải có đường đi tức là c[i][j] ≠ +∞.
* Tiến hành duyệt quay lui:
* Với mỗi thành phố a[1] có thể chọn mà a[0] có đường đi tới trực tiếp.
* Tương tự với mỗi cách thử chọn của a[2] thì a[1] có đường đi tới, với mỗi thành phố a[i] thì có thể chọn các thành phố mà nó chưa đi qua và có đường đi trực tiếp đến.
* Với mỗi bước đi ta tiến hành kiểm tra nếu chi phí hiện tại lớn hơn chi phí tối ưu thì không xét trường hợp này.
* Tiếp tục làm như vậy cho đến khi tìm được hành trình hoàn chỉnh, tiến hành so sánh với giá trị tốt nhất(f\_best) của các hành trình trước đó nếu nhỏ hơn thì cập nhật lại f\_best.

1. Đánh giá thuật toán :

* Thuật toán quay lui là một kĩ thuật thiết kế giải thuật dựa trên đệ qui. Ý tưởng của quay lui là tìm lời giải ở từng bước, mỗi bước chọn một trong số các lựa chọn phù hợp tiến hành đệ qui.
* Thuật toán này đã hạn chế được một số hành trình đi không cần thiết cải tiến được nhược điểm của thuật toán vét cạn đáng kể ví dụ với n = 13 thì thuật toán vét cạn mất khoảng 52,35(s), còn đối với nhánh cận thì chỉ mất một khoảng thời gian rất nhỏ để ra kết quả bài toán như trên, cho ta thấy hiệu quả của giải thuật này.

II.3. Bài toán lập chỉ mục của sách :

* Đề bài :
* Lập index cho một văn bản
* Input: Đầu vào là văn bản vào các từ khóa(mỗi từ trên một dòng) từ file.
* Output: In ra file số lần xuất hiện của từ, xuất hiện ở những dòng nào.
* Yêu cầu: Sử dụng danh sách liên kết đơn hoặc cây nhị phân tìm kiếm.

1. Ý tưởng thuật toán :

* Bài toán sử dụng danh sách liên kết đơn để lưu trữ các từ khóa, dòng, ..bằng cách lần lượt thêm các nút vào trong danh sách với điều kiện đi kèm.
* Số lần xuất hiện của từ khóa sẽ được đếm theo từng dòng mà có từ xuất hiện một.
* Ta duyệt qua danh sách các từ khóa và dòng nếu gặp từ khóa xuất hiện ở dòng nào đó thì thêm các nút vào danh sách các nút dòng với dữ liệu là các dòng có chứa từ khóa, tiếp tục làm như vậy cho đến khi duyệt qua hết các giá trị.
* Tiếp đó với mỗi từ khóa sau khi duyệt ta tiến hành thêm vào nút với dữ liệu là các từ khóa, số lần xuất hiện và danh sách các dòng tương úng xuất hiện.

1. Giải thuật:

* Trước tiên, ta khai báo cấu trúc một cái node cho các dòng và nút các từ khóa.
* Hàm thêm node vào sau, với danh sách ban đầu rỗng thì cho node đầu danh sách bằng node cần thêm, ngược lại thì ta duyệt danh sách đến phần tử cuối rồi thêm vào sau phần tử đó.
* Đọc dữ liệu từ file vào các dòng của văn bản và các từ khóa vào một mảng string.
* Sử dụng hàm tìm chuỗi trong chuỗi trả về vị trí đầu tiên trong chuỗi cần tìm.
* Cần một hàm đếm số lần xuất hiện của các từ khóa. Ở đây ta sẽ đếm số lần xuất hiện của từ khóa trên từng dòng một và tiến hành tính tổng số lần xuất hiện trên các dòng.
* Khi gặp một dòng có xuất hiện từ khóa, thêm dòng đó vào danh sách các node dòng và sau một vòng lặp của từ khóa tiến hành gán xóa đi các nút dòng để thêm nút dòng vào các nút tiếp theo.

1. Đánh giá thuật toán :

* Bài toán sử dụng danh sách liên kết nên số lượng phần tử có thể tùy ý và dễ dàng thay đổi kích thước, dễ dàng bỏ đi các phần tử trong nó, rất phù hợp với bài toán này ( có số lượng phần tử thay đổi thường xuyên).
* Để truy xuất hay chèn một nút vào danh sách liên kết thì độ phức tạp của nó là O(n) do cần phải duyệt qua các phần tử, không thể truy xuất đến phần tử cần tìm ngay được. Tuy nhiên, nếu ta biết được vị trí con trỏ trỏ đến nút cần xóa hay chèn thì chỉ là O(1).

**PHẦN III: KẾT LUẬN**

1. Kết quả:

* Sau khi làm xong bài tập 1, chương trình sắp xếp với các dãy số tùy ý in kết quả ra màn hình hoặc ra file.
* Sau khi làm xong bài tập 2, ta có thể sử dụng thuật toán quay lui giải quyết một số bài toán thú vị như soduku, quân hậu, mã đi tuần, du lịch, cái túi,… Tìm được cách xếp hậu với số quân là n, tìm được đường đi tối ưu từ thành phố bất kì và quay trở lại đó với đầu vào là k thành phố
* Sau khi làm xong bài toán 3, ta có thể sử dụng danh sách liên kết giải quyết một số bài toán. Lập được index của một file văn bản đầu vào, hoặc lập index cho 1 sách,…

1. Kiến thức thu được sau khi hoàn thành bài tập lớn:

* Sau khi làm xong bài tập 1, ta có thể sử dụng các thuật toán sắp xếp với các bài toán thích hợp để mang lại hiệu quả tốt nhất. Bên cạnh đó có thể sử dụng đa luồng trong sắp xếp với các dữ liệu lớn để tối ưu hóa tài nguyên máy tính cũng như thời gian.
* Sau khi làm xong bài tập 2, ta có thể sử dụng thuật toán quay lui giải quyết một số bài toán thú vị như soduku, quân hậu, mã đi tuần, du lịch, cái túi,… Bên cạnh đó, ta có thể sử dụng thuật toán nhánh cận để giảm bớt khả năng tìm kiếm lời giải không cần thiết, cải thiện thêm thời gian thực hiện thuật toán
* Sau khi làm xong bài toán 3, ta có thể sử dụng thành thạo danh sách liên kết giải quyết một số bài toán.