Luận văn

So sánh độ phức tạp của thuật toán QuickSort và InsertSort

PHẦN A: NỀN TẢNG LÝ THUYẾT

1. Mô tả chức năng và yêu cầu

1.1. Khái quát về sắp xếp:

Để thuận tiện và giảm thiểu thời gian thao tác mà đặc biệt là để tìm kiếm dữ liệu dễ dàng và nhanh chóng,thong thường trước khi thao tác thì dữ liệu trên mảng,trên tập tin đã có thứ tự.Do vậy thao tác sắp xếp dữ liệu là một trong những thao tác cần thiết và thường gặp trong quá trình lưu trữ,quản lý dữ liệu

Có rất nhiều cách sắp xếp dữ liệu,nhưng ở đây ta chỉ quan tâm đến 2 thuật toán là sắp xếp bằng phương pháp chèn (Insertion Sort) và sắp xếp dựa trên sự phân hoạch (Quick Sort).Ta sẽ đi phân tích hai thuật toán sắp xếp này để so sánh và đánh giá độ phức tạp của chúng.

1.2. Mục tiêu của bài toán:

Phân tích,đánh giá và so sánh độ phức tạp(trên lý thuyết) và so sánh thời gian tính toán(trên thực nghiệm) của 2 giải thuật.

2. Đánh giá độ phức tạp của giải thuật sắp xếp bằng phương pháp chèn(Insertion Sort)

2.1. Ý tưởng thuật toán:

Giả sử ta có dãy a_1, a_2, …, an trong đó i phần tử đầu tiên a_1, a_2, …, a_i đã có thứ tự. Ý tưởng của thuật toán là tìm vị trị thích hợp và chèn phần tử a_{i+1} vào dãy đã có thứ tự trên để có được một dãy mới có thứ tự. Cứ thế, làm đến cuối dãy ta sẽ được một dãy có thứ tự.

Với dãy ban đầu a_1, a_2, …, a_n ta có thể coi đoạn chỉ có một phần tử a_1 là một đoạn đã có thứ tự, sau đó ta chèn phần tử a_2 vào dãy a_1 để có dãy a_1a_2 có thứ tự. Tiếp đó, ta lại chèn phần tử a_3 vào dãy a_1a_2 để có dãy $a_1a_2a_3$ có thứ tự. Cứ thế, đến cuối cùng ta chèn phần tử a_n vào dãy $a_1a_2…a_{n-1}$ ta sẽ được dãy $a_1a_2…a_n$ có thứ tự.

2.2.Cài đặt thuật toán

```
void insertionsort(int a[],int n)
{
    int pos,x;
    for(int i=0;i<n-1;i++)
    {
        x=a[i+1];pos=i;
        while(pos>=0 && a[pos]>x)
        {
            a[pos+1]=a[pos];
            pos--;
        }
        a[pos+1]=x;
    }
}
```

2.3.Đánh giá độ phức tạp:

Ta thấy các phép so sánh xảy ra trong vòng lặp nhằm tìm vị trí thích hợp pos để chèn x. Mỗi lần so sánh mà thấy vị trí đang xét không thích hợp, ta dời phần tử a[pos] sang phải.

Ta cũng thấy số phép gán và số phép so sánh của thuật toán phụ thuộc vào tình trạng của dãy ban đầu. Do đó ta chỉ có thể ước lượng như sau:

2.3.1. Trường hợp tốt nhất: Dãy ban đầu đã có thứ tự. Ta tìm được ngay vị trí thích hợp để chèn ngay lần so sánh đầu tiên mà không cần phải vô vòng lặp. Như vậy, với i chạy từ 2 đến n thì số phép so sánh tổng cộng sẽ là n-1. Còn với số phép gán, do thuật toán không chạy vào vòng lặp nên xét i bất kỳ, ta luôn chỉ phải tốn 2 phép gán(x = a[i] và a[pos] = x). Từ đây, ta tính được số phép gán tổng cộng bằng 2(n - 1).

2.3.2. Trường hợp xấu nhất:Dãy ban đầu có thứ tự ngược. Ta thấy ngay vị trí thích hợp pos luôn là vị trí đầu tiên của dãy đã có thứ tự, và do

đó, để tìm ra vị trí này ta phải duyệt hết dãy đã có thứ tự. Xét i bất kỳ, ta có số phép so sánh là i-1, số phép gán là (i - 1) + 2 = i + 1. Với i chạy từ 2 đến n, ta tính được số phép so sánh tổng cộng bằng 1 + 2 + … + (n - 1) = n(n - 1)/2 và số phép gán bằng 3 + 4 + .. + (n + 1) = (n + 4)(n - 1)/2

Tổng kết lại, ta có độ phức tạp của Insertion Sort như sau:

- Trường hợp tốt nhất: O(n)
- Trường hợp xấu nhất $O(n^2)$

3. Đánh giá độ phức tạp của giải thuật sắp xếp nhanh(Quick Sort)

3.1.Ý tưởng thuật toán:

QuickSort chia mảng thành hai danh sách bằng cách so sánh từng phần tử của danh sách với một phần tử được chọn được gọi là phần tử chốt. Những phần tử nhỏ hơn hoặc bằng phần tử chốt được đưa về phía trước và nằm trong danh sách con thứ nhất, các phần tử lớn hơn chốt được đưa về phía sau và thuộc danh sách con thứ hai. Cứ tiếp tục chia như vậy tới khi các danh sách con đều có độ dài bằng 1.

3.2.Cài đặt thuật toán:

```
void quicksort(int a[],int left,int right)
{
    if(left>=right)return;
    int x=a[(left+right)/2];
    int i=left;
    int j=right;
    do
    {
            while(a[i]<x)i++;
            while(a[j]>x)j--;
            if(i<=j)//chua duyet het
            {
                    swap(a[i],a[j]);
                    i++;
```

```
            j--;
            }
    }while(i<j);
        quicksort(a,left,j);
        quicksort(a,i,right);
}
```

3.3. Độ phức tạp của thuật toán

Ta nhận thấy hiệu quả của thuật toán phụ thuộc vào việc chọn giá trị mốc (hay phần tử chốt).

3.3.1. Trường hợp tốt nhất: mỗi lần phân hoạch ta đều chọn được phần tử median (phần tử lớn hơn hay bằng nửa số phần tử và nhỏ hơn hay bằng nửa số phần tử còn lại) làm mốc. Khi đó dãy được phân hoạch thành hai phần bằng nhau, và ta cần $\log_2(n)$ lần phân hoạch thì sắp xếp xong. Ta cũng dễ nhận thấy trong mỗi lần phân hoạch ta cần duyệt qua n phần tử. Vậy độ phức tạp trong trường hợp tốt nhất thuộc $O(n\log_2(n))$.

3.3.2. Trường hợp xấu nhất: mỗi lần phần hoạch ta chọn phải phần tử có giá trị cực đại hoặc cực tiểu làm mốc. Khi đó dãy bị phân hoạch thành hai phần không đều: một phần chỉ có một phần tử, phần còn lại có n-1 phần tử. Do đó, ta cần tới n lần phân hoạch mới sắp xếp xong. Vậy độ phức tạp trong trường hợp xấu nhất thuộc $O(n^2)$.

Tổng kết lại, ta có độ phức tạp của Quick Sort như sau:

- *Trường hợp tốt nhất: $O(n\log_2(n))$*
- *Trường hợp xấu nhất: $O(n^2)$*
- *Trường hợp trung bình: $O(n\log_2(n))$*

PHẦN B : THỰC NGHIỆM

1. Mô tả giải thuật :

Giải thuật được cài đặt trên ngôn ngữ lập trình c/c++. Ý tưởng của việc cài đặt giải thuật như sau:

Khởi tạo ngẫu nhiên n phần tử, ghi ra 1 file text

Đọc các phần tử từ file text vào file excel

Tính độ phức tạp dựa vào α

2. Cài đặt

2.1.InsertionSort:

```
void insertionsort(int A1[],int num,int &sosanhI,int &hoanviI)
{
    int X=0,k=1,j=0;
    while(k<num)
    {
        j=k+1;
        X=A1[j];
        while(X<A1[j-1])
        {
            sosanhI++;
            A1[j]=A1[j-1];
            hoanviI++;
```

```
            j--;
        }
        A1[j]=X;
        k++;
    }
}
```

2.2.QuickSort

```
void quicksort(int A2[],int first,int last,int &sosanhQ,int &hoanviQ)
{
    if(first>=last)
        return;
    int mid=(first+last)/2;
    int MID=A2[mid];
    int F=first,L=last;
    while(F<=L)
    {
        while(A2[F]<MID)
        {
            F++;
            sosanhQ++;
        }
        while(A2[L]>MID)
            L--;
        if(F<=L)
        {
            doicho(A2[F],A2[L]);
            F++;
            L--;
            hoanviQ++;
        }
    }
    cout.flush();
    quicksort(A2,first,L,sosanhQ,hoanviQ);
    cout.flush();
```

```
        quicksort(A2,F,last,sosanhQ,hoanviQ);
}
```

3. Kết quả thực nghiệm:

	A	B	C	D	E	F
1	STT	So phan tu	InsertionSort		QuickSort	
2			So phep sc sanh	So phep hoan vi	Sc phep so sanh	So phep hoan vi
3	1	4891	6082823	6082823	24057	15424
4	2	7793	15090009	15090009	36428	26103
5	3	6369	10130415	10130415	29546	20983
6	4	9983	24836414	24836414	48870	34118
7	5	9706	23790774	23790774	47128	33709
8	6	7403	13936816	13936816	35286	24745
9	7	5000	6195645	6195645	23396	15984
10	8	2476	1534056	1534056	12063	7266
11	9	7164	12858011	12858011	26671	24073
12	10	8609	18411572	18411572	53793	29158
13	11	5540	7590232	7590232	25456	17776
14	12	9081	20636720	20636720	50646	31120
15	13	1545	583318	583318	5500	4227
16	14	5369	7259348	7259348	25148	17301
17	15	7084	12597716	12597716	38538	23196
18	16	6617	10824211	10824211	33330	21844
19	17	5009	6309076	6309076	18850	15886
20	18	2840	2024251	2024251	11286	8554
21	19	5701	8044058	8044058	22979	18528
22	20	3526	3099037	3099037	15238	10875
23	21	4991	6380179	6380179	20719	15905

Sheet1 Sheet2 Sheet3

B□ng s□ li□u thu đ□□c khi ch□□ng trình ch□y

SUMMARY OUTPUT

Regression Statistics	
Multiple R	0.978889234
R Square	0.958224132
Adjusted R Square	0.957353802
Standard Error	1566645.402
Observations	50

ANOVA

	df	*SS*	*MS*	*F*	*gnificance F*
Regression	1	2.7E+15	2.7E+15	1100.989	9.34E-35
Residual	48	1.18E+14	2.45E+12		
Total	49	2.82E+15			

	Coefficients	*andard Err*	*t Stat*	*P-value*	*Lower 95%*	*Upper 95%*	*ower 95.0%*	*pper 95.0%*
Intercept	-6131831.915	509864	-12.0264	4.31E-16	-7156982	-5106682	-7156982	-5106682
X Variable 1	2805.685036	84.55659	33.18115	9.34E-35	2635.673	2975.697	2635.673	2975.697

Y=2805.685*X-6131832

Phương trình hồi qui tuyến tính của phép so sánh và phép gán trong giải thuật InsertionSort

SUMMARY OUTPUT

Regression Statistics	
Multiple R	0.971953677
R Square	0.944693949
Adjusted R Square	0.94354174
Standard Error	3444.87165
Observations	50

ANOVA

	df	*SS*	*MS*	*F*	*gnificance F*
Regression	1	9.73E+09	9.73E+09	819.8978	7.9E-32
Residual	48	5.7E+08	11867141		
Total	49	1.03E+10			

	Coefficients	*andard Err*	*t Stat*	*P-value*	*Lower 95%*	*Upper 95%*	*ower 95.0%*	*pper 95.0%*
Intercept	-2604.602381	1121.132	-2.32319	0.024453	-4858.79	-350.416	-4858.79	-350.416
X Variable 1	5.32389747	0.18593	28.63386	7.9E-32	4.95006	5.697735	4.95006	5.697735

Y=5.32389747*X-2604.602381

Phương trình hồi qui tuyến tính của phép so sánh trong giải thuật QuickSort

SUMMARY OUTPUT

Regression Statistics	
Multiple R	0.99950975
R Square	0.99901974
Adjusted R Square	0.998999318
Standard Error	300.7166778
Observations	50

ANOVA

	df	*SS*	*MS*	*F*	*gnificance F*
Regression	1	4.42E+09	4.42E+09	48918.59	7.1E-74
Residual	48	4340665	90430.52		
Total	49	4.43E+09			

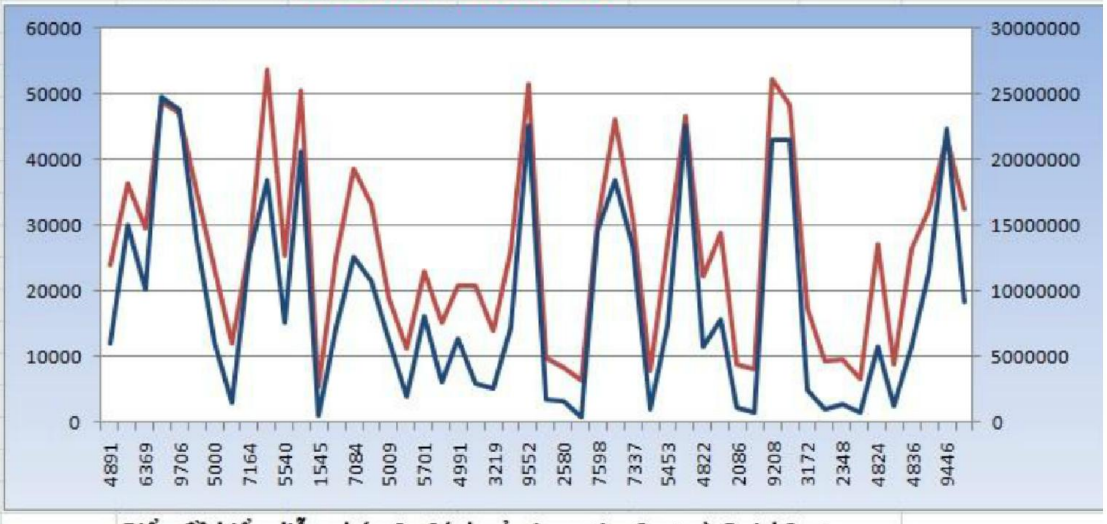
	Coefficients	*andard Err*	*t Stat*	*P-value*	*Lower 95%*	*Upper 95%*	*ower 95.0%*	*pper 95.0%*
Intercept	-1770.050571	97.86811	-18.0861	4.55E-23	-1966.83	-1573.27	-1966.83	-1573.27
X Variable 1	3.589808237	0.016231	221.1755	7.1E-74	3.557174	3.622442	3.557174	3.622442

Y=3.589808237*X-1770.050571

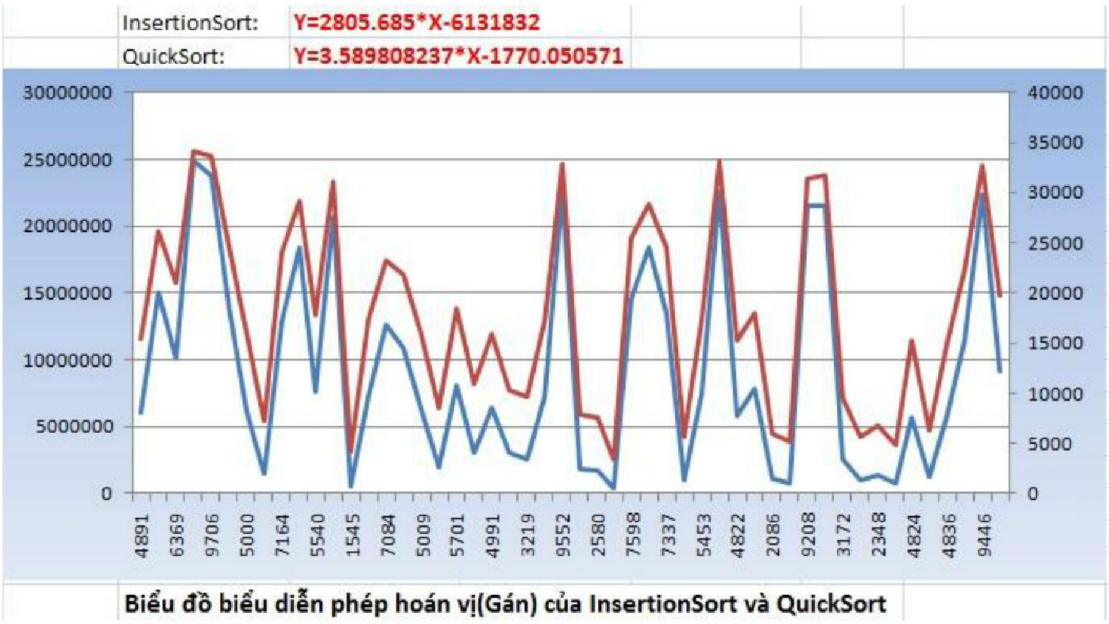
Phương trình hồi qui tuyến tính của phép hoán vị(gán) trong giải thuật QuickSort

InsertionSort: **Y=2805.685*X-6131832**

QuickSort: **Y=5.32389747*X-2604.602381**



Biểu đồ biểu diễn phép So Sánh của InsertionSort và QuickSort



Biểu đồ biểu diễn phép hoán vị(Gán) của InsertionSort và QuickSort

KẾT LUẬN

Dựa vào phương trình hồi qui tuyến tính của Phép Hoán vị(Gán) InsertionSort và phương trình hồi qui tuyến tính Phép Hoán vị(Gán) QuickSort ; phương trình hồi qui tuyến tính của Phép So sánh InsertionSort và phương trình hồi qui tuyến tính Phép So Sánh QuickSort,ta thấy hệ số α của giải thuật QuickSort nhỏ hơn hệ số α của giải thuật InsertionSort,điều này chứng tỏ giải thuật QuickSort chạy nhanh hơn giải thuật InsertSort.Ngoài ra,đồ thị biểu diễn các phương trình hồi qui tuyến tính của 2 giải thuật cũng cho thấy rằng giải thuật QuickSort chạy nhanh hơn giải thuật InsertionSort.

Phần lý thuyết cũng cho thấy độ phức tạp của giải thuật InsertionSort lớn hơn hoặc bằng độ phức tạp của giải thuật QuickSort.

Nhóm chúng em sẽ cố gắng tìm hiểu sâu sắc hơn để hiểu rõ về hai giải thuật này,trong quá trình làm không tránh khỏi thiếu xót,kính mong Thầy bỏ qua. Xin chân thành cảm ơn.