

Chương 1

THUẬT TOÁN QUICK SORT

TS. Nguyễn Tấn Trần Minh Khang

ThS. Cáp Phạm đình Thăng

Chương 01 - 1

1. BÀI TOÁN DẪN NHẬP

- **Bài toán:** Cho mảng một chiều các số nguyên và giá trị x . Hãy tách mảng a ban đầu thành 2 mảng b và c sao cho mảng b chỉ chứa các giá trị nhỏ hơn x , mảng c chứa các giá trị lớn hơn x .

1. BÀI TOÁN DẪN NHẬP

```
11. void Split(int a[], int n,  
               int x,  
               int b[], int &k,  
               int c[], int &l)  
12. {  
13.     k = l = 0;  
14.     for(int i=0; i<n; i++)  
15.         if(a[i]<x)  
16.             b[k++] = a[i];  
17.         else  
18.             if(a[i]>x)  
19.                 c[l++] = a[i];  
20. }
```

2. TƯ TƯỞNG THUẬT TOÁN QUICK SORT

- Thuật toán quick sort chia không gian cần sắp xếp thành 2 không gian con là không gian con 1 và không gian con 2. Không gian con 1 là không gian mà tất cả các phần tử thuộc không gian này đều nhỏ hơn tất cả các phần tử thuộc không gian con 2.
 - + Nếu không gian con thứ nhất có nhiều hơn một phần tử thì sắp xếp không gian con này bằng thuật toán Quick Sort.
 - + Nếu không gian con thứ hai có nhiều hơn một phần tử thì sắp xếp không gian con này bằng thuật toán Quick Sort.

TS. Nguyễn Tấn Trần Minh Khang

ThS. Cáp Phạm đình Thắng

Chương 01 - 4

3. HÀM CÀI ĐẶT

```
10. void QuickSort(int a[], int n)
11. {
12.     if (n <= 1)
13.         return;
14.     int b[100]; int k;
15.     int c[100]; int l;
16.     int TrongTai = a[0];
17.     Split(a, n, TrongTai, b, k, c, l);
18.     QuickSort(b, k);
19.     QuickSort(c, l);
20.     for(int i=0; i<k; i++)
21.         a[i] = b[i];
22.     for(int i=0; i<n-k-l; i++)
23.         a[k+i] = TrongTai;
24.     for(int i=0; i<l; i++)
25.         a[k+(n-k-l)+i] = c[i];
```

26. }
TS. Nguyễn Tấn Trần Minh Khang

3. HÀM CÀI ĐẶT

```
11. void Split(int a[], int n,  
              int x,  
              int b[], int &k,  
              int c[], int &l)  
12. {  
13.     k = l = 0;  
14.     for(int i=0; i<n; i++)  
15.         if(a[i]<x)  
16.             b[k++] = a[i];  
17.         else  
18.             if(a[i]>x)  
19.                 c[l++] = a[i];  
20. }
```

Chương 1

THUẬT TOÁN QUICK SORT

TS. Nguyễn Tấn Trần Minh Khang

ThS. Cáp Phạm đình Thăng

Chương 01 - 7

2. TƯ TƯỞNG THUẬT TOÁN QUICK SORT

- Thuật toán quick sort chia không gian cần sắp xếp thành 2 không gian con là không gian con 1 và không gian con 2. Không gian con 1 là không gian mà tất cả các phần tử thuộc không gian này đều nhỏ hơn tất cả các phần tử thuộc không gian con 2.
 - + Nếu không gian con thứ nhất có nhiều hơn một phần tử thì sắp xếp không gian con này bằng thuật toán Quick Sort.
 - + Nếu không gian con thứ hai có nhiều hơn một phần tử thì sắp xếp không gian con này bằng thuật toán Quick Sort.

TS. Nguyễn Tấn Trần Minh Khang

ThS. Cáp Phạm đình Thắng

Chương 01 - 8

4. MỘT CÁCH CÀI ĐẶT KHÁC

```
10. void QuickSort(int a[],  
    int Left, int Right)  
  
11. {  
12.     if (Left < Right)  
13.     {  
14.         int m1, m2;  
15.         Partition(a, Left,  
                    Right, m1, m2);  
16.         QuickSort(a, Left, m1);  
17.         QuickSort(a, m2, Right);  
18.     }  
19. }
```

```

10. void Partition(int a[],
    int Left, int Right,
    int &m1,int &m2)
11. {
12.     int pivot=a[(Left+Right)/2];
13.     int low = Left;
14.     int high = Right;
15.     while(low<high)
16.     {
17.         while (a[low]<pivot)
18.             low++;
19.         while (a[high]>pivot)
20.             high--;
21.         if (low<=high)
22.         {
23.             HoanVi (a[low],a[high]);
24.             low++;
25.             high--;
26.         }
27.     }
28.     m1 = high;

```

4. MỘT CÁCH CÀI ĐẶT KHÁC

```
10. void SapTang(int a[], int n)
11. {
12.     QuickSort(a, 0, n-1);
13. }
14. void HoanVi(int &a, int &b)
15. {
16.     int temp=a;
17.     a=b;
18.     b=temp;
19. }
```

```
10. void Partition(int a[],
    int Left, int Right,
    int &m1,int &m2)
11. {
12.     int pivot=a[(Left+Right)/2];
13.     int low = Left;
14.     int high = Right;
15.     while(low<high)
16.     {
17.         while (a[low]<pivot)
18.             low++;
19.         while (a[high]>pivot)
20.             high--;
21.         if (low<=high)
22.         {
23.             HoanVi (a[low],a[high]);
24.             low++;
25.             high--;
26.         }
27.     }
28.     m1 = high;
```

0 1 2 3 4 5 6 7 8 9 10 11

5 10 8 41 87 8 81 15 59 6 8 15

pivot=8

low



high



0	1	2	3	4	5	6	7	8	9	10	11
5	10	8	41	87	8	81	15	59	6	8	15

pivot=8

low

high



0 1 2 3 4 5 6 7 8 9 10 11

5 10 8 41 87 8 81 15 59 6 8 15

`pivot=8`

low



high



0 1 2 3 4 5 6 7 8 9 10 11

5 10 8 41 87 8 81 15 59 6 8 15

pivot=8

low



high



0 1 2 3 4 5 6 7 8 9 10 11

5	10	8	41	87	8	81	15	59	6	8	15
---	----	---	----	----	---	----	----	----	---	---	----

pivot=8

low



high



0 1 2 3 4 5 6 7 8 9 10 11

5	8	8	41	87	8	81	15	59	6	10	15
---	---	---	----	----	---	----	----	----	---	----	----

`pivot=8`

low



high



0 1 2 3 4 5 6 7 8 9 10 11

5	8	8	41	87	8	81	15	59	6	10	15
---	---	---	----	----	---	----	----	----	---	----	----

pivot=8

low



high



0 1 2 3 4 5 6 7 8 9 10 11

5	8	8	41	87	8	81	15	59	6	10	15
---	---	---	----	----	---	----	----	----	---	----	----

pivot=8

low



high



0 1 2 3 4 5 6 7 8 9 10 11

5	8	8	41	87	8	81	15	59	6	10	15
---	---	---	----	----	---	----	----	----	---	----	----

pivot=8

low



high



0 1 2 3 4 5 6 7 8 9 10 11

5	8	8	41	87	8	81	15	59	6	10	15
---	---	---	----	----	---	----	----	----	---	----	----

pivot=8

low



high



0 1 2 3 4 5 6 7 8 9 10 11

5	8	6	41	87	8	81	15	59	8	10	15
---	---	---	----	----	---	----	----	----	---	----	----

`pivot=8`

`low`



`high`



0 1 2 3 4 5 6 7 8 9 10 11

5	8	6	41	87	8	81	15	59	8	10	15
---	---	---	----	----	---	----	----	----	---	----	----

`pivot=8`

`low`



`high`



0 1 2 3 4 5 6 7 8 9 10 11

5	8	6	41	87	8	81	15	59	8	10	15
---	---	---	----	----	---	----	----	----	---	----	----

pivot=8

low



high



0 1 2 3 4 5 6 7 8 9 10 11

5	8	6	41	87	8	81	15	59	8	10	15
---	---	---	----	----	---	----	----	----	---	----	----

pivot=8

low



high



0	1	2	3	4	5	6	7	8	9	10	11
5	8	6	41	87	8	81	15	59	8	10	15

`pivot=8`

`low`



`high`



0	1	2	3	4	5	6	7	8	9	10	11
5	8	6	41	87	8	81	15	59	8	10	15

pivot=8

low

high



0	1	2	3	4	5	6	7	8	9	10	11
5	8	6	41	87	8	81	15	59	8	10	15

pivot=8

low

high



0	1	2	3	4	5	6	7	8	9	10	11
5	8	6	41	87	8	81	15	59	8	10	15

pivot=8

low



high



0	1	2	3	4	5	6	7	8	9	10	11
5	8	6	8	87	41	81	15	59	8	10	15

pivot=8

low

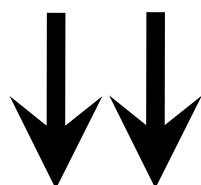
high



0	1	2	3	4	5	6	7	8	9	10	11
5	8	6	8	87	41	81	15	59	8	10	15

pivot=8

low high

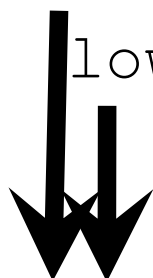


0	1	2	3	4	5	6	7	8	9	10	11
5	8	6	8	87	41	81	15	59	8	10	15

pivot=8

high

low



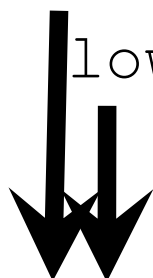
0 1 2 3 4 5 6 7 8 9 10 11

5	8	6	8	87	41	81	15	59	8	10	15
---	---	---	---	----	----	----	----	----	---	----	----

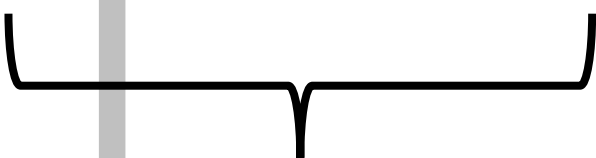
pivot=8

high

low



0	1	2	3	4	5	6	7	8	9	10	11
5	8	6	8	87	41	81	15	59	8	10	15

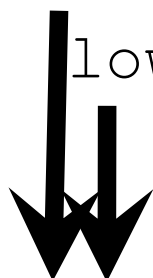


(Left, high)

pivot=8

high

low



0 1 2 3 4 5 6 7 8 9 10 11

5	8	6	8	87	41	81	15	59	8	10	15
---	---	---	---	----	----	----	----	----	---	----	----

(low, Right)

Chương 1

THUẬT TOÁN QUICK SORT

TS. Nguyễn Tấn Trần Minh Khang

ThS. Cáp Phạm đình Thăng

Chương 01 - 37

```
11. void QuickSort (int a[],
                    int d, int c)
12. {
13.     if (d >= c)
14.         return;
15.     int low = d;
16.     int high = c;
17.     int tt = a[(d + c) / 2];
18.     do {
19.         while (a[low] < tt)
20.             low++;
21.         while (a[high] > tt)
22.             high--;
23.         if (low <= high)
24.         {
25.             HoanVi (a[low], a[high]);
26.             low++;
27.             high--;
28.         }
29.     } while (low < high);
30.     if (d < high)
31.         QuickSort (a, d, high);
32.     if (c > low)
33.         QuickSort (a, low, c);
34. }
```

Chương 1

THUẬT TOÁN QUICK SORT

TS. Nguyễn Tấn Trần Minh Khang

ThS. Cáp Phạm đình Thăng

Chương 01 - 39

BÀI TẬP LUYỆN TẬP

– Bài toán: Định nghĩa hàm sắp mảng một chiều các số thực tăng dần bằng thuật toán quick sort.

– Hàm cài đặt

```
10. void SapTang(float a[],  
               int n)  
11. {  
12.     QuickSort(a, 0, n-1);  
13. }  
14. void HoanVi(float&a, float&b)  
15. {  
16.     float temp=a;  
17.     a=b;  
18.     b=temp;  
19. }
```

TS. Nguyễn Tấn Trần Minh Khang

ThS. Cáp Phạm đình Thắng

Chương 01 - 40


```

10. void QuickSort(float a[],
                    int d,int c)
11. {
12.     if (d>=c) return;
13.     int i=d;
14.     int j=c;
15.     float tt=a[(d+c)/2];
16.     do{
17.         while(a[i]<tt) i++;
18.         while(a[j]>tt) j--;
19.         if (i<=j)
20.             {
21.                 HoanVi(a[i],a[j]);
22.                 i++;
23.                 j--;
24.             }
25.     }while(i<j);
26.     if (d<j)
27.         QuickSort(a,d,j);
28.     if (c>i)
29.         QuickSort(a,i,c);

```

BÀI TẬP LUYỆN TẬP

- Bài toán: Định nghĩa hàm sắp mảng một chiều các phân số tăng dần bằng thuật toán quick sort.

- Cấu trúc dữ liệu

```
10. struct phanso
```

```
11. {
```

```
12.     int tu;
```

```
13.     int mau;
```

```
14. };
```

```
15. typedef struct phanso PHANSO;
```

```
16. void HoanVi (PHANSO&a, PHANSO&b)
```

```
17. {
```

```
18.     PHANSO temp=a;
```

```
19.     a=b;
```

```
20.     b=temp;
```

TS. Nguyễn Tấn Trần Minh Khang

ThS. Cáp Phạm đình Thắng

Chương 01 - 42

BÀI TẬP LUYỆN TẬP

– Hàm cài đặt

```
1. int SoSanh(PHANSO x, PHANSO y)
2. {
3.     float a=(float)x.tu/x.mau;
4.     float b=(float)y.tu/y.mau;
5.     if(a>b)
6.         return 1;
7.     if(a<b)
8.         return -1;
9.     return 0;
10. }
11. void SapTang(PHANSO a[], int n)
12. {
13.     QuickSort(a, 0, n-1);
14. }
```

TS. Nguyễn Tấn Trần Minh Khang

ThS. Cáp Phạm đình Thắng

Chương 01 - 43

```
10. void QuickSort (PHANSO a[],  
                    int d, int c)  
11. {  
12.     if (d >= c)  
13.         return;  
14.     int i = d;  
15.     int j = c;  
16.     PHANSO tt = a[(d + c) / 2];  
17.     do {  
18.         while (sosanh(a[i], tt) == -1)  
19.             i++;  
20.         while (sosanh(a[j], tt) == 1)  
21.             j--;  
22.         if (i <= j)  
23.         {  
24.             HoanVi(a[i], a[j]);  
25.             i++;  
26.             j--;  
27.         }  
28.     } while (i < j);  
29.     if (d < j)  
30.         QuickSort(a, d, j);  
31.     if (c > i)  
32.         QuickSort(a, i, c);  
33. }
```

BÀI TẬP

- Hãy nêu một ưu điểm và một khuyết điểm mà theo bạn là tiêu biểu nhất của phương pháp sắp xếp Quick Sort khi **cài đặt bằng đệ quy**.
- Ưu điểm của cài đặt thuật toán QuickSort bằng đệ quy: Rõ ràng, dễ hiểu và dễ cài đặt.
- Khuyết điểm của cài đặt thuật toán QuickSort bằng đệ quy: Tràn stack trong tình huống có nhiều lời gọi hàm đệ quy khi kích thước của mảng cần sắp xếp lớn.