Chương 1
THUẬT TOÁN SELECTION SORT

TS. Nguyễn Tấn Trần Minh Khang

ThS. Cáp Phạm đình Thăng

1. BÀI TOÁN DẪN NHẬP

- Bài toán: Viết hàm tìm vị trí giá trị nhỏ nhất trong mảng một chiều các số thực
- Ví dụ:

0	1	2	3	4
12	43	1	34	22

- Kết quả: 2

1. BÀI TOÁN DẪN NHẬP

- Hàm cài đặt

```
1. int ViTriNhoNhat(float a[],
                    int n)
2. {
3.     int lc = 0;
4.     for(int i=0;i<n;i++)
5.         if (a[i]<a[lc])
6.             lc = i;
7.     return lc;
8. }
```

2. TƯ TƯỞNG THUẬT TOÁN

- Bài toán: Cho mảng một chiều a có n phần tử: a_0, a_1, a_2,…, a_{n-2}, a_{n-1}. Hãy sắp xếp các phần tử trong mảng tăng dần.

2. TƯ TƯỞNG THUẬT TOÁN

- Bước 0: Tìm vị trí giá trị nhỏ nhất trong phạm vi [0,n-1] và hoán vị giá trị tại vị trí này và phần tử a[0].
- Bước 1: Tìm vị trí giá trị nhỏ nhất trong phạm vi [1,n-1] và hoán vị giá trị tại vị trí này và phần tử a[1].
- …
- Bước i: Tìm vị trí giá trị nhỏ nhất trong phạm vi [i,n-1] và hoán vị giá trị tại vị trí này và phần tử a[i].
- …
- Bước n-2: Tìm vị trí giá trị nhỏ nhất trong phạm vi [n-2,n-1] và hoán vị giá trị tại vị trí này và phần tử a[n-2].

3. ÁP DỤNG THUẬT TOÁN

- Hãy sắp xếp mảng sau tăng dần:

24	45	23	13	43	-1

- Thứ tự các bước khi sắp tăng dần mảng trên bằng thuật toán Selection sort.

3. ÁP DỤNG THUẬT TOÁN

- Bước 0:
  - Tìm vị trí nhỏ nhất trong phạm vi [0,5] của mảng

0	1	2	3	4	5
24	45	23	13	43	-1

  - Kết quả: Vị trí 5.
  - Hoán vị giá trị tại vị trí nhỏ nhất và giá trị tại vị trí 0:

0	1	2	3	4	5
24	45	23	13	43	-1

  - Kết quả sau khi sắp xếp ở bước 0

0	1	2	3	4	5
-1	45	23	13	43	24

3. ÁP DỤNG THUẬT TOÁN

- Bước 1:
  - Tìm vị trí nhỏ nhất trong phạm vi [1,5] của mảng

0	1	2	3	4	5
-1	45	23	13	43	24

  - Kết quả: Vị trí 3.
  - Hoán vị giá trị tại vị trí nhỏ nhất và giá trị tại vị trí 1:

0	1	2	3	4	5
-1	45	23	13	43	24

  - Kết quả sau khi sắp xếp ở bước 1

0	1	2	3	4	5
-1	13	23	45	43	24

3. ÁP DỤNG THUẬT TOÁN

- Bước 2:
  - Tìm vị trí nhỏ nhất trong phạm vi [2,5] của mảng

0	1	2	3	4	5
-1	13	23	45	43	24

  - Kết quả: Vị trí 2.
  - Hoán vị giá trị tại vị trí nhỏ nhất và giá trị tại vị trí 2:

0	1	2	3	4	5
-1	13	23	45	43	24

  - Kết quả sau khi sắp xếp ở bước 2

0	1	2	3	4	5
-1	13	23	45	43	24

3. ÁP DỤNG THUẬT TOÁN

- Bước 3:
  - Tìm vị trí nhỏ nhất trong phạm vi [3,5] của mảng

0	1	2	3	4	5
-1	13	23	45	43	24

  - Kết quả: Vị trí 5.
  - Hoán vị giá trị tại vị trí nhỏ nhất và giá trị tại vị trí 3:

0	1	2	3	4	5
-1	13	23	45	43	24

  - Kết quả sau khi sắp xếp ở bước 3

0	1	2	3	4	5
-1	13	23	24	43	45

3. ÁP DỤNG THUẬT TOÁN

- Bước 4:
  - Tìm vị trí nhỏ nhất trong phạm vi [4,5] của mảng

0	1	2	3	4	5
-1	13	23	24	43	45

  - Kết quả: Vị trí 4.
  - Hoán vị giá trị tại vị trí nhỏ nhất và giá trị tại vị trí 4:

0	1	2	3	4	5
-1	13	23	24	43	45

  - Kết quả sau khi sắp xếp ở bước 4

0	1	2	3	4	5
-1	13	23	24	43	45

4. HÀM CÀI ĐẶT

- Bài toán: Định nghĩa hàm sắp mảng một chiều các số nguyên tăng dần bằng thuật toán Selection sort.
- Hàm cài đặt

```
10. void SelectionSort(int a[],
                       int n)
11. {
12.   for(int i=0;i<=n-2;i++)
13.   {
14.      int lc = i;
15.      for(int j=i;j<=n-1;j++)
16.         if(a[j]<a[lc])
17.            lc = j;
18.      int temp = a[i];
19.      a[i] = a[lc];
20.      a[lc] = temp;
21.   }
22. }
```

5. DANH SÁCH LIÊN KẾT ĐƠN

- Bài toán: Định nghĩa hàm sắp dslk đơn các số nguyên tăng dần bằng thuật toán Selection sort.
- Cấu trúc dữ liệu

```
10. struct node
11. {
12.     int info;
13.     struct node*pNext;
14. };
15. typedef struct node NODE;
16. struct list
17. {
18.     NODE*pHead;
19.     NODE*pTail;
20. };
21. typedef struct list LIST;
```

5. DANH SÁCH LIÊN KẾT ĐƠN

- Hàm cài đặt

```
10. void SelectionSort(LIST &l)
11. {
12.   for(NODE*p=l.pHead;p->pNext;
                          p=p->pNext)
13.   {
14.     NODE*lc = p;
15.     for(NODE*q=p;q;
                    q=q->pNext)
16.       if(q->info < lc->info)
17.         lc = q;
18.     int temp = p->info;
19.     p->info = lc->info;
20.     lc->info = temp;
21.   }
22. }
```

6. MỘT CÁCH CÀI ĐẶT KHÁC

- Bài toán: Định nghĩa hàm sắp mảng một chiều các số nguyên tăng dần bằng thuật toán Selection sort.
- Hàm cài đặt 1

```
10. void SelectionSort(int a[],
                       int n)
11. {
12.   for(int i=0;i<=n-2;i++)
13.   {
14.     int lc = i;
15.     for(int j=n-1;j>=i+1;j--)
16.       if(a[j]<a[lc])
17.         lc = j;
18.     int temp = a[i];
19.     a[i] = a[lc];
20.     a[lc] = temp;
21.   }
22. }
```

6. MỘT CÁCH CÀI ĐẶT KHÁC

- Bài toán: Định nghĩa hàm sắp mảng một chiều các số nguyên tăng dần bằng thuật toán Selection sort.
- Hàm cài đặt 2

```
10. void SelectionSort(int a[],
                       int n)
11. {
12.   for(int i=n-1;i>=1;i--)
13.   {
14.     int lc = i;//hay lc = 0;
15.     for(int j=0;j<=i;j++)
16.       if(a[j]>a[lc])
17.         lc = j;
18.     int temp = a[i];
19.     a[i] = a[lc];
20.     a[lc] = temp;
21.   }
22. }
```

6. MỘT CÁCH CÀI ĐẶT KHÁC

- Bài toán: Định nghĩa hàm sắp mảng một chiều các số nguyên tăng dần bằng thuật toán Selection sort.
- Hàm cài đặt 3

```
10. void SelectionSort(int a[],
                       int n)
11. {
12.   for(int i=n-1;i>=1;i--)
13.   {
14.     int lc = i;//hay lc = 0;
15.     for(int j=i;j>=0;j--)
16.       if(a[j]>a[lc])
17.         lc = j;
18.     int temp = a[i];
19.     a[i] = a[lc];
20.     a[lc] = temp;
21.   }
22. }
```

7. DANH SÁCH LIÊN KẾT KÉP

- Bài toán: Định nghĩa hàm sắp dslk kép các số nguyên tăng dần bằng thuật toán Selection sort.
- Cấu trúc dữ liệu

```
11. struct node
12. {
13. |    int info;
14. |    struct node*pNext;
15. |    struct node*pPrev;
16. };
17. typedef struct node NODE;
18. struct list
19. {
20. |    NODE*pHead;
21. |    NODE*pTail;
22. };
23. typedef struct list LIST;
```

7. DANH SÁCH LIÊN KẾT KÉP

```
10. void SelectionSort(LIST &ℓ)
11. {
12.   for(NODE*p=ℓ.pHead;p->pNext;
                          p=p->pNext)
13.   {
14.     NODE*lc = p;
15.     for(NODE*q=p->pNext;q;
                        q=q->pNext)
16.       if(q->info < lc->info)
17.         lc = q;
18.     int temp = p->info;
19.     p->info = lc->info;
20.     lc->info = temp;
21.   }
22. }
```

7. DANH SÁCH LIÊN KẾT KÉP

```
10. void SelectionSort(LIST &ℓ)
11. {
12.   for(NODE*p=ℓ.pHead;p->pNext;
                          p=p->pNext)
13.   {
14.     NODE*lc = p;
15.     for(NODE*q=ℓ.pTail;q!=p;
                        q=q->pPrev)
16.       if(q->info < lc->info)
17.         lc = q;
18.     int temp = p->info;
19.     p->info = lc->info;
20.     lc->info = temp;
21.   }
22. }
```

7. DANH SÁCH LIÊN KẾT KÉP

```
10. void SelectionSort(LIST &ℓ)
11. {
12.   for(NODE*p=ℓ.pTail;p->pPrev;
                          p=p->pPrev)
13.   {
14.     NODE*lc = p;
15.     for(NODE*q=ℓ.Head;q!=p;
                        q=q->pNext)
16.       if(q->info > lc->info)
17.         lc = q;
18.     int temp = p->info;
19.     p->info = lc->info;
20.     lc->info = temp;
21.   }
22. }
```

7. DANH SÁCH LIÊN KẾT KÉP

```
10. void SelectionSort(LIST &ℓ)
11. {
12.   for(NODE*p=ℓ.pTail;p->pPrev;
                          p=p->pPrev)
13.   {
14.     NODE*lc = p;
15.     for(NODE*q=p->pPrev;q;
                        q=q->pPrev)
16.       if(q->info > lc->info)
17.         lc = q;
18.     int temp = p->info;
19.     p->info = lc->info;
20.     lc->info = temp;
21.   }
22. }
```