LY TU TRONG COLLEGE

CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

GIẢNG VIÊN: NGUYỄN VÂN GIANG

LY TU TRONG COLLEGE

CHƯƠNG 1: MỞ ĐẦU

Mục tiêu:

Sau khi học xong người học hiểu được:

- Mục đích, vị trí, vai trò của môn học;
- Khái niệm cấu trúc dữ liệu, giải thuật và các yêu cầu chung khi xây dựng cấu trúc dữ liệu và giải thuật.

LY TU TRONG COLLEGE

1.1. CẤU TRÚC DỮ LIỆU (Data Structure)

- CTDL là cách thức tổ chức dữ liệu sao cho phản ánh đúng thực tế bài toán.
- CTDL là tập hợp các biến và các kiểu dữ liệu được liên kết với nhau.

Ví dụ: CTDL của chương trình quản lý SV

```
typedef struct SV
{     char MaSV[10];
      char TenSV[50];
      int NS;
      float Diem;
};
SV a[100];
```

- Một cấu trúc dữ liệu tốt phải thỏa:

+ Phản ánh đúng thực tế

+ Phù hợp với các thao tác xử lý trên dữ liệu đó

+ Tiết kiệm tài nguyên của hệ thống (thời gian,bộ nhớ):

LY TU TRONG COLLEGE

1.2. GIẢI THUẬT (Algorithms)

- Giải thuật là trình tự các thao tác xử lý dữ liệu để giải quyết bài toán.
- Gọi n là kích thước của dữ liệu đầu vào, ta ký hiệu T(n) là hàm biểu diễn thời gian thực hiện của giải thuật.
- Mức độ tăng lên của T(n) khi n tăng lên gọi là độ phức tạp tính tóan của giải thuật và được biểu điễn bằng hàm O (gọi là Big O).
- Các lớp độ phức tạp tính toán phổ biến:

$$O(1) < O(\log n) < O(n) < O(n\log n) < O(n^2) < O(n^3) < O(2^n)$$

1.3. MỐI QUAN HỆ GIỮA CTDL VÀ GIẢI THUẬT

Data structures + Algorithms = Programs

- Để xác định được giải thuật phù hợp cần phải biết nó tác động đến những loại dữ liệu nào

- Khi chọn lựa một cấu trúc dữ liệu cũng cần phải hiểu rõ những thao tác nào sẽ được tác động lên nó

BÀI TẬP

Viết chương trình DanhSachSoNguyen.cpp thực hiện các công việc sau:

a. Nhập mảng số nguyên

b. Xuất mảng số nguyên

c. Cho phép người dùng nhập một số x và kiểm tra xem trong mảng có số x hay không ?

Chỉ số i	0	1	2	3	4	5	6	7 (n-1)
Giá trị a[i]	7	2	0	9	8	1	4	3

Chỉ số i	0	1	2	3	4	5	6	7 (n-1)
Giá trị a[i]	7	2	0	9	8	1	4	3

```
void NhapMang (int a[],int &n)
{
      printf("Nhap so luong phan tu: ");
      scanf("%d",&n);
      for (int i=0;i<n;i++)
            {
                  printf("a[%d] =",i);
                  scanf("%d",&a[i]);
                  printf("\n");
            }
}
```

LY TU TRONG COLLEGE

CHƯƠNG 2: TÌM KIẾM VÀ SẮP XẾP
BÀI 1: TÌM KIẾM (SORT)

Mục tiêu:

Sau khi học xong người học hiểu được:

- Trình bày được ý tưởng của phương pháp tìm kiếm tuyến tính, tìm nhị phân;
- Mô phỏng và đánh giá được giải thuật của phương pháp tìm kiếm tuyến tính, tìm nhị phân;
- Cài đặt được các giải thuật tìm kiếm trên mảng.
- Vận dụng được các giải thuật tìm kiếm trong lập trình các ứng dụng

1. Bài toán tìm kiếm

INPUT:

- Dãy (a) gồm n phần tử $a_0, a_1, \ldots, a_{n-1}$.
- x là giá trị của phần tử cần tìm

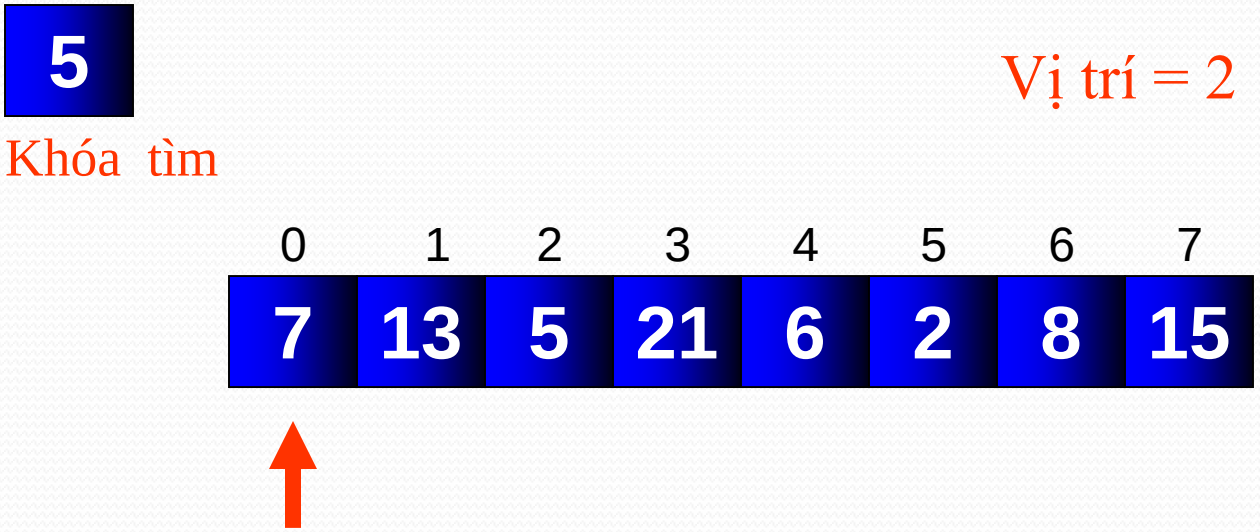
OUTPUT:

k ($0 \le k \le n-1$) nếu $a_k = x$, (k là vị trí xuất hiện của x trong dãy (a))

- 1 nếu trong dãy không có phần tử nào bằng x

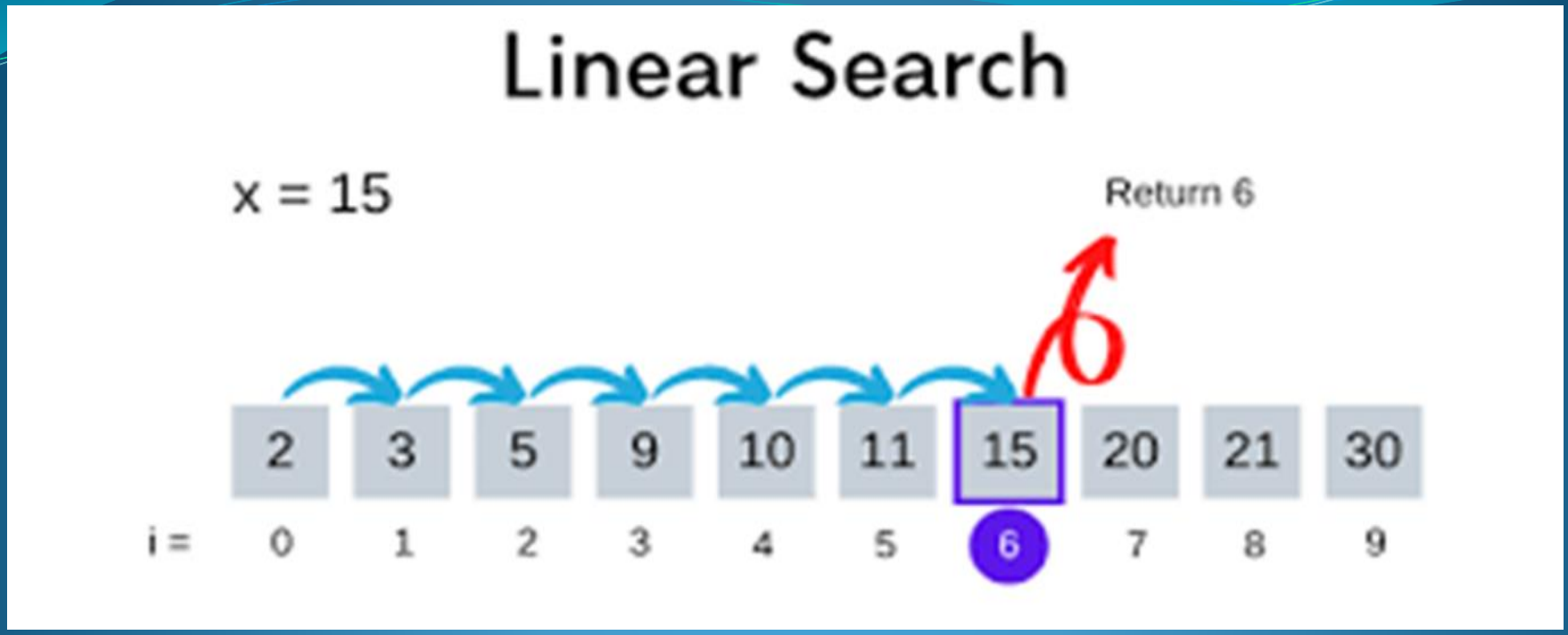
2. Tìm tuyến tính (Linear Search)

2.1. Ý tưởng của giải thuật



Tìm thành công

Số lần so sánh: 3



Lần lượt so sánh x với các a[i] (i=0,1,2,…,n-1)

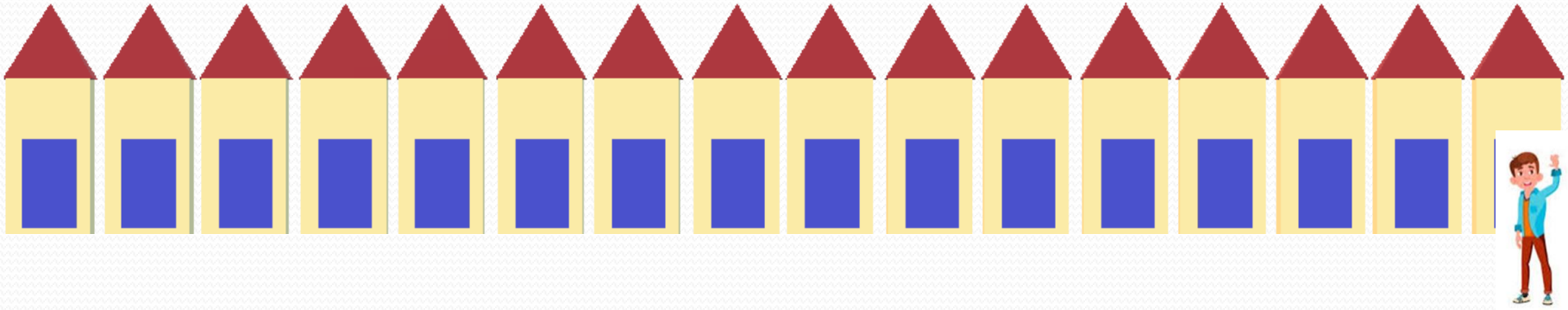
- Nếu có a[i] = x thì dừng và trả về i
- Nếu không có a[i] nào bằng x thì trả về -1

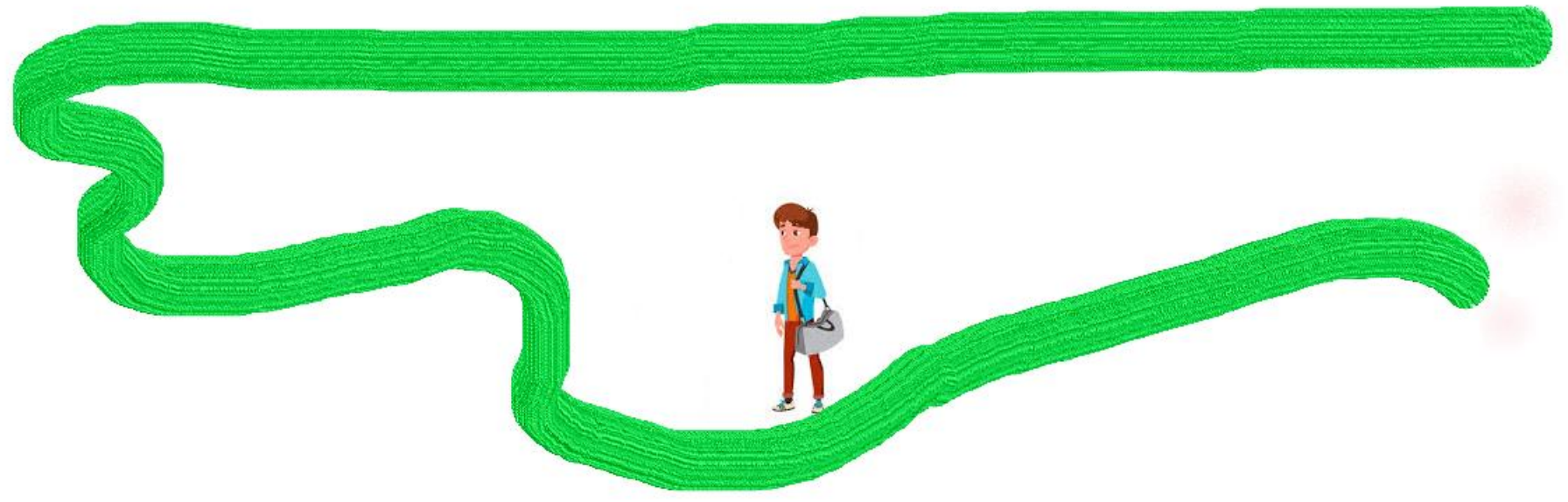
i	0	1	2	3	4	5	6	7(n-1)
a[i]	12	2	8	5	1	6	8	15

Lưu ý : Giải thuật tìm tuyến tính luôn trả về vị trí xuất hiện đầu tiên của x trong dãy số.

2.2. Cài đặt

```
int linearSearch (int a[], int n, int x)
{
      int i = 0;
      while (i<n&&a[i]!=x)
            i++;
      if( i==n)return -1;
      return i;
}
```





LY TU TRONG COLLEGE

2.3. Thuật toán cải tiến

```
int LinearSearch (int a[], int n, int x)
{
      int i = 0;
      a[n]=x;
      while (a[i]!=x)
            i++;
      if( i==n)return -1;
      return i;
}
```

LY TU TRONG COLLEGE

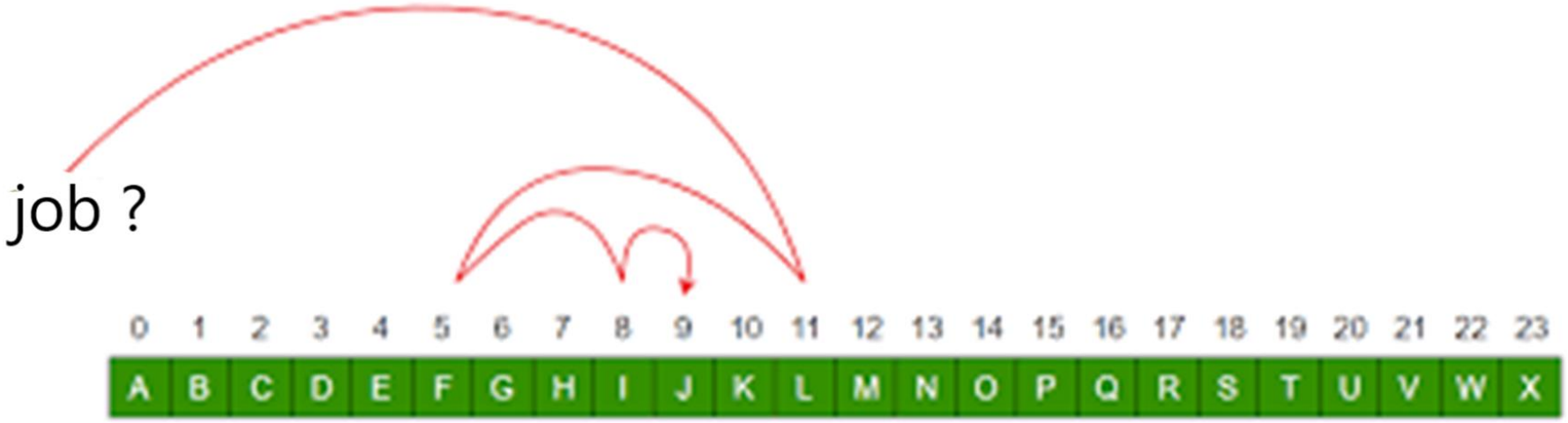
Đánh giá giải thuật:

Độ phức tạp của giải thuật là O(n)

Trường hợp	Số lần so sánh	Ghi chú
Tốt nhất	1	Phần tử đầu tiên có giá trị x
Xấu nhất	n + 1	Không có x trong mảng
Trung bình	(n+1)/2	Giả sử xác suất các phần tử trong mảng nhận giá trị x là như nhau.

3. Tìm nhị phân (Binary Search)

3.1. Ý tưởng của giải thuật



Điều kiện áp dụng: Dãy (a) đã được sắp thứ tự tăng dần

Xét dãy: a[l], a[l+1],…,a[r] (l ≤ r)

- Gọi m=(l+r)/2 và so sánh a[m] với x :
  + Nếu a[m]<x: Thực hiện lại giải thuật với l=m+1
  + Nếu a[m]>x: Thực hiện lại giải thuật với r=m-1
- Giải thuật dừng khi l> r và trả về -1

i	0	1	2	3	4	5	6	7(n-1)
a[i]	2	3	5	7	8	8	10	15

Tìm x=10 :

- l = 0, r = 7: m = 3, a[3] = 7 <x
- l = 4, r = 7: m = 5, a[5] = 8 < x
- l = 6, r = 7: m = 6, a[6] = 10 = x. Trả về 6.

i	0	1	2	3	4	5	6	7(n-1)
a[i]	2	3	5	7	8	8	10	15

Tìm x=5 ?

- l = 0, r = 7: m = 3, a[3] = 7 >x
- l = 0, r = 2: m = 1, a[1] = 3 < x
- l = 2, r = 2: m = 2, a[2] = 5 = x. Trả về 2

Tìm x=4 ?

- l = 0, r = 7: m = 3, a[3] = 7 >x
- l = 0, r = 2: m = 1, a[1] = 3 < x
- l = 2, r = 2: m = 2, a[2] = 5 >x. Trả về 2.
- l = 2, r = 1: l>r . Trả về -1

3.2. Cài đặt:

```
int binarySearch(int a[],int n,int x)
{
      int l=0,r=n-1,m;
      do
      {
            m=(l+r)/2;
            if (x==a[m])return m;
            if (x<a[m]) r=m-1;
            else l=m+1;
      }while (l<=r);
      return -1;
}
```

LY TU TRONG COLLEGE

Đánh giá giải thuật:

Số phần tử	Số lần chia để tìm x
2	1
4	2
8	3
16	4
n	$\log_2 n$

Độ phức tạp của giải thuật là $O(\log_2 n)$

Trường hợp	Số lần so sánh	Ghi chú
Tốt nhất	1	Phần tử giữa của mảng có giá trị x
Xấu nhất	$\log_2 n$	Không có x trong mảng
Trung bình	$\log_2 n/2$	Giả sử xác suất các phần tử trong mảng nhận giá trị x là như nhau

LY TU TRONG COLLEGE

BÀI TẬP

Viết tiếp chương trình **DanhSachSoNguyen.cpp**, thực hiện các công việc sau :

d. Cài đặt giải thuật tìm tuyến tính cải tiến và gọi hàm này để tìm số x có trong mảng không ? Nếu có thì trả về vị trí của số x, nếu không có thì xuất câu thông báo.

e. Cài đặt hàm tìm nhị phân và gọi hàm này để tìm số x có trong mảng không ? Nếu có thì trả về vị trí của số x, nếu không có thì xuất câu thông báo.

f. So sánh kết quả tìm kiếm của 2 hàm nếu x=7 và (a) : 1, 3, 7, 7, 8, 9,11. Giải thích kết quả.

LY TU TRONG COLLEGE

BÀI 2: TÌM KIẾM (Search)

Mục tiêu:

Sau khi học xong người học hiểu được:

- Trình bày được ý tưởng của phương pháp sắp xếp Chèn trực tiếp và Phân hoạch;
- Mô phỏng và đánh giá được giải thuật của phương pháp sắp xếp Chèn trực tiếp và Phân hoạch;
- Cài đặt được các giải thuật sắp xếp trên mảng.
- Vận dụng được các giải thuật sắp xếp trong lập trình các ứng dụng

LY TU TRONG COLLEGE

1. Bài toán sắp xếp

Giả sử cần sắp xếp tăng dần

INPUT:

- Dãy (a) gồm n phần tử a_0, a_1,..., a_{n-1}.

OUTPUT:

- Dãy (a) thỏa : $a_0 \leq a_2 \leq ... \leq a_{n-1}$

2. Phương pháp đổi chỗ trực tiếp (Interchange Sort)

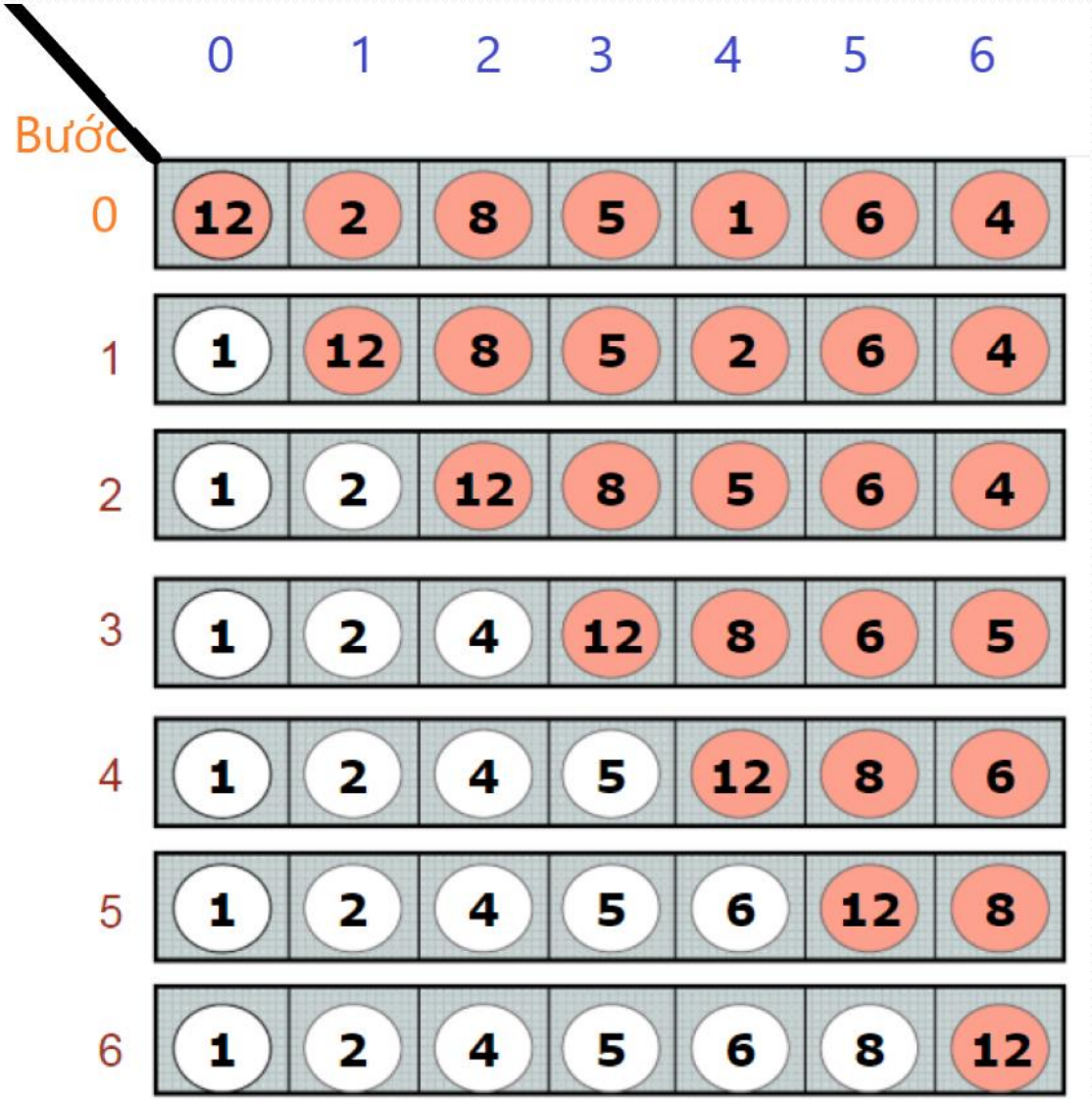
2.1. Ví dụ: Sắp tăng dần dãy số 4, 6, 5, 1, 2, 9, 3

Bước	0	1	2	3	4	5	6
Bước 0	4	6	5	1	2	9	3
Bước 1	1	6	5	4	2	9	3
Bước 1	1	5	6	4	2	9	3
Bước 1	1	4	6	5	2	9	3
Bước 2	1	2	6	5	4	9	3
Bước 2	1	2	5	6	4	9	3
Bước 2	1	2	4	6	5	9	3
Bước 3	1	2	3	6	5	9	4
Bước 3	1	2	3	5	6	9	4
Bước 4	1	2	3	4	6	9	5
Bước 4	1	2	3	4	6	9	5
Bước 5	1	2	3	4	5	9	6
Kết quả	1	2	3	4	5	6	6

Trình bày rút gọn:

Bước	0	1	2	3	4	5	6
Bước 0	4	6	5	1	2	9	3
Bước 1	1	6	5	4	2	9	3
Bước 2	1	2	6	5	4	9	3
Bước 3	1	2	3	6	5	9	4
Bước 4	1	2	3	4	6	9	5
Bước 5	1	2	3	4	5	9	6
Kết quả	1	2	3	4	5	6	6

Sắp xếp dãy số sau tăng dần: 12, 2, 8, 5, 1, 6, 4



2.2. Ý tưởng của giải thuật:

Giải thuật gồm n-1 bước (từ bước i= 0 đến bước i= n-2).

Ở bước 0, ta phải sắp xếp sao cho a[0] sẽ là phần tử nhỏ nhất dãy a[0],…,a[n-1].

Ở bước 1, ta phải sắp xếp sao cho a[1] sẽ là phần tử nhỏ nhất dãy a[1],…,a[n-1]

Tổng quát:

Ở bước i, ta phải sắp xếp sao cho a[i] sẽ là phần tử nhỏ nhất dãy a[i],…,a[n-1], bằng cách :

So sánh a[i] với các a[j] (i<j≤n-1), nếu thấy a[i]>a[j] thì hóan vị a[i] với a[j].

Giải thuật gồm n-1 bước (bước i= 0,... n-2).

Ở bước i, ta phải sắp xếp sao cho a[i] sẽ là phần tử nhỏ nhất dãy a[i],…,a[n-1], bằng cách :

So sánh a[i] với các a[j] (i<j≤n-1), nếu thấy a[i]>a[j] thì hóan vị a[i] với a[j].

2.3. Cài đặt:

```
void interchangeSort(int a[], int n)
{
    for (int i=0; i<n-1;i++)
        for (int j=i+1;j<n ;j++)
            if (a[i]>a[j])
                swap(a[i],a[j]);
}
```

```
void interchangeSort(int a[], int n)
{
      for (int i=0; i<n-1;i++)
            for (int j=i+1;j<n ;j++)
                  if (a[i]>a[j])
                        swap(a[i],a[j]);
}
```

Đánh giá giải thuật:

- Vòng lặp for j lặp (n - i- 1) lần
- Số lần lặp của 2 vòng lặp for lồng nhau là:

$\sum_0^{n-2}(n-i-1)$

= (n-1)+(n-2)+(n-3)+….+3+2+1

$= \frac{[(n-1)+1](n-1)}{2} = \frac{n(n-1)}{2}$

Độ phức tạp của giải thuật là : $O(n^2)$

3. Phương pháp phân hoạch (Quick Sort)

3.1. Ý tưởng của giải thuật

a [left] ---------- x ----------a[right]

a[i]<x a[j]>x

left ----------→ i

j←----------right

LY TU TRONG COLLEGE

Giả sử ta cần sắp tăng dần dãy a[l],...,a[r]

Bước 1: Chọn x=a[(l+r)/2] làm mốc, i=l, j=r

Bước 2:

Trong khi a[i]<x thì tăng i

Trong khi a[j]>x thì giảm j

Nếu i<=j thì hóan vị a[i], a[j] rồi tăng i, giảm j

Nếu i<=j thì quay lại bước 2, ngược lại qua bước 3

Bước 3:

Nếu l<j thì thực hiện lại giải thuật với r=j

Nếu i<r thì thực hiện lại giải thuật với l=i

3.2. Ví dụ:

	0	1	2	3	4	5	6
l=0, r=6,	i	i		j	j	j	j
x=4	**2**	**9**	**5**	**(4)**	**7**	**6**	**8**
		j	i j		j	j	j
	2	**(4)**	**5**	**9**	**7**	**6**	**8**
l=0, r=1,	i j	i j					
x=2	**(2)**	**4**					
l=2, r=6,			i	i		j	j
x=7			**5**	**9**	**(7)**	**6**	**8**
				j	i j	i	
			5	**6**	**(7)**	**9**	**8**

	0	1	2	3	4	5	6
l=2, r=3, *x=5*		j	i j **(5)**	i j **6**			
l=5, r=6, *x=9*						i **(9)**	j **8**
						j **8**	i **(9)**
Kết quả	2	4	5	6	7	8	9

LY TU TRONG COLLEGE

3.3. Cài đặt:

```
void quickSort(int a[],int l,int r)
{
     int  x=a[(l+r)/2],i=l,j=r;
     do
     {
          while (a[i]<x) i++;
          while (a[j]>x) j--;
          if (i<=j)
          {    swap(a[i],a[j]); i++;j--;}
     }while (i<j);
     if (l<j) quickSort(a,l,j);
     if (i<r) quickSort(a,i,r);
}
```

LY TU TRONG COLLEGE

Đánh giá giải thuật:

Độ phức tạp của giải thuật là O (nlogn)

BÀI TẬP

Bổ sung chương trình `DanhSachSoNguyen.cpp`, thực hiện các công việc sau bằng 2 phương pháp : Interchange Sort và Quick Sort

g. Sắp xếp danh sách tăng dần.

h. Sắp xếp danh sách giảm