# Các Thuật Toán Tìm Kiếm

| | |
|---|---|
| MSSV | 20280083 |
| Tên | Lại Toàn Thắng |

2. Cho biết kết quả in ra màn hình với các trường hợp mảng A và x sau:

A = 3 6 8 9 0 -1

X = 3

A = 1 2 3 4

X = 5

A = 6 6 6 6 6 6

X = 6

```
Nhap so luong phan tu: 6
Nhap a[0]: 3 6 8 9 0 -1
Nhap a[1]: Nhap a[2]: Nhap a[3]: Nhap
a[4]: Nhap a[5]: Nhap gia tri phan tu
can tim kiem: 3
Vi tri x trong mang A la: 1
```

```
Nhap so luong phan tu: 4
Nhap a[0]: 1 2 3 4
Nhap a[1]: Nhap a[2]: Nhap a[3]: Nhap
gia tri phan tu can tim kiem: 5
Khong tim thay x trong mang A
```

```
Nhap so luong phan tu: 6
Nhap a[0]: 6
Nhap a[1]: 6
Nhap a[2]: 6
Nhap a[3]: 6
Nhap a[4]: 6
Nhap a[5]: 6
Nhap gia tri phan tu can tim kiem: 6
Vi tri x trong mang A la: 1
```

3. Nếu dòng

```
int i = LinearExhaustive(a, n, x);
```

trong hàm main ta thay hàm LinearExhaustive bằng hàm LinearSentinel. Cho biết kết quả in ra màn hình với A={1,2,3,4} và x=5.

```
Nhap so luong phan tu: 4
Nhap a[0]: 1
Nhap a[1]: 2
Nhap a[2]: 3
Nhap a[3]: 4
Nhap gia tri phan tu can tim kiem: 5
Vi tri x trong mang A la: 5
```

4. Sửa lại code trong hàm main để có thể chạy đúng khi gọi hàm LinearSentinel ở câu 3.

```c
int x;
printf("Nhap gia tri phan tu can tim kiem: ");
scanf("%d", &x);
int i = LinearSentinel(a, n, x);
if (i == n) {
      printf("Khong tim thay x trong mang A\n");
}
else {
      printf("Vi tri x trong mang A la: %d\n", i + 1);
      //Tại sao lại xuất i+1 ở đây
}
```

5. Hãy ghi chú các thông tin bằng cách trả lời các câu hỏi ứng với các dòng lệnh có yêu cầu ghi chú (`//Ghi chú`)

6. Ứng dụng hàm BinarySearch trong hàm main. Diễn giải ý nghĩa code của hàm này.

```c
cout << "\n Binary Search\n";
int j = BinarySearch(a, n, x);
if (j == -1) {
      cout << "Element not found\n";
}
else cout << x << " is present at index " << i;
```

Ý tưởng thuật toán: Giống như tên của function (Tìm kiếm nhị phân) thì nguyên lý tìm kiếm dựa trên một mảng đã sắp xếp. Cho left là vị trí 0 của mảng và right là n - 1.

mid = ( left + right)  / 2

Nếu giá trị của mảng tại vị trí mid mà bằng x thì return vị trí.

Nếu nhỏ hơn thì tìm kiếm bên trái

Và ngược lại thì tìm kiếm bên phải.

Nếu chưa tìm được thì return -1 để thông báo phần tử không có trong mảng

7. Viết lại hàm BinarySearch dùng đệ quy.

```
int BinarySearch_Recursion(int a[], int l, int r, int x) {
	if (l > r) return -1;
	int m = (l + r) / 2;
	if (a[m] == x)
		return m;
	if (x < a[m])
		return BinarySearch_Recursion(a, l, m - 1, x);
	else  return BinarySearch_Recursion(a, m + 1, r, x);
}
```

8. (Nâng cao) Đo thời gian tính toán của mỗi thuật toán tìm kiếm. Gợi ý: hàm clock_t của thư viện C/C++ (Xem code mẫu)

```
#include <time.h>

clock_t start, end;
double cpu_time_used;

start = clock();
... /* Do the work. */
end = clock();
cpu_time_used = ((double) (end - start)) / CLOCKS_PER_SEC;
printf("Thoi gian xu ly la: %f\n", cpu_time_used);
```

```
Nhap so luong phan tu: 6
Nhap a[0]: 3
Nhap a[1]: 4
Nhap a[2]: 6
Nhap a[3]: 8
Nhap a[4]: 9
Nhap a[5]: 34
Nhap gia tri phan tu can tim kiem: 34

Vi tri x trong mang A la: 6

Thoi gian xu ly la: 0.001000

 Binary Search
34 is present at index 6
Thoi gian xu ly la: 0.000000
```

```
Nhap so luong phan tu: 6
Nhap a[0]: 1
Nhap a[1]: 2
Nhap a[2]: 3
Nhap a[3]: 4
Nhap a[4]: 5
Nhap a[5]: 6
Nhap gia tri phan tu can tim kiem: 34

Khong tim thay x trong mang A

Thoi gian xu ly la: 0.003000

 Binary Search
Element not found

Thoi gian xu ly la: 0.001000
```

9. (Nâng cao) Xây dựng cấu trúc WORD trong từ điển (gồm tên của từ và nghĩa của từ) và áp dụng thuật toán tìm kiếm để xây dựng chương trình tra từ điển. Hàm so sánh chuỗi có thể dùng hàm strcmp (http://www.cplusplus.com/reference/clibrary/cstring/strcmp/)

```
struct WORD{
        char Name[256];
        char Meaning[512];
}
```

Cái này em sẽ làm sau. Còn bây giờ, em học mấy môn kia nữa. Em chào thầy 😊😊