- **Mục đích**
- **Thuật toán tìm kiếm tuyến tính**
- **Thuật toán tìm kiếm nhị phân**

Tìm kiếm trong mảng

Mục đích

- Cùng với chức năng sắp xếp, tìm kiếm là chức năng không thể tránh khỏi ở hầu hết các ứng dụng
- Tìm kiếm cho phép ta lọc ra những thông tin mong muốn
- Ví dụ tìm điểm thi của một thí sinh theo mã dự thi
- Tìm danh sách những người có tài khoản tiền gửi >= 1 tỉ vnđ
- Nội dung bài này sẽ trình bày hai thuật toán tìm kiếm: linear search & binary search

Thuật toán tìm kiếm tuyến tính

- Linear search là thuật toán tìm kiếm đơn giản.
- Gọi x là giá trị cần tìm, i là biến chạy của chỉ số mảng. Ý tưởng của thuật toán:

Cho biến chạy i chạy từ chỉ số đầu đến chỉ số cuối của mảng
Nếu phần tử tại vị trí i bằng x

Trả về chỉ số i

Mặc định trả về -1

Thực thi thuật toán

- Thực thi chi tiết:

```
// ham tim kiem phan tuyen tinh
int linearSearch(const int arr[], const size_t size, const int key)
{
	for (size_t i = 0; i < size; i++)
	{
		if (key == arr[i]) { // neu tim thay
			return i;	// tra ve vi tri i
		}
	}
	return -1; // mac dinh neu khong tim thay tra ve -1
}
```

Thuật toán tìm kiếm nhị phân

- Việc tìm kiếm tuyến tính chỉ phù hợp với lượng nhỏ dữ liệu
- Với các bài toán có dữ liệu lớn, ta thực hiện tìm kiếm với thuật toán binary search
- Các bước thực hiện thuật toán:
- Mảng dùng để tìm kiếm sẽ được sắp xếp theo thứ tự tăng dần
- Gọi chỉ số phần tử giới hạn đoạn chứa nội dung cần tìm kiếm là low và height

```
Khi (low <= height):
    middle = (low + height) / 2;
    Nếu a[middle] = key:
        trả về middle
    Nếu a[middle] < key:
        low = middle + 1
    Nếu a[middle] > key:
        height = middle - 1;
Mặc định trả về -1
```

Thực thi thuật toán

- Sau đây là mã nguồn thực thi thuật toán trên:

```
int binarySearch(int arr[], size_t size, int key, int low, int height)
{
	while (low <= height) {
		// determine the middle element index
		size_t middle = (low + height) / 2;
		// if key matched middle element
		if (key == arr[middle]) {
			return middle; // return middle as index
		} // then end this task
		// search the left part of middle element
		else if (key < arr[middle]) {
			height = middle - 1;
		}
		else { // search on the right part of middle element
			low = middle + 1;
		}
	}
	return -1; // nothing found
}
```

So sánh hai thuật toán

Tiêu chí đánh giá	Tìm kiếm tuyến tính	Tìm kiếm nhị phân
Độ phức tạp	O(n)	O(log(n))
Áp dụng khi	Bộ dữ liệu đối sánh nhỏ, không quan tâm đến tốc độ	Bộ dữ liệu đối sánh thường lớn, cần ưu tiên tốc độ
Logic code	Đơn giản	Phức tạp
Bước tiên quyết	Không áp dụng	Mảng cần tìm phải được sắp xếp

Tiếp theo

Mảng hai chiều