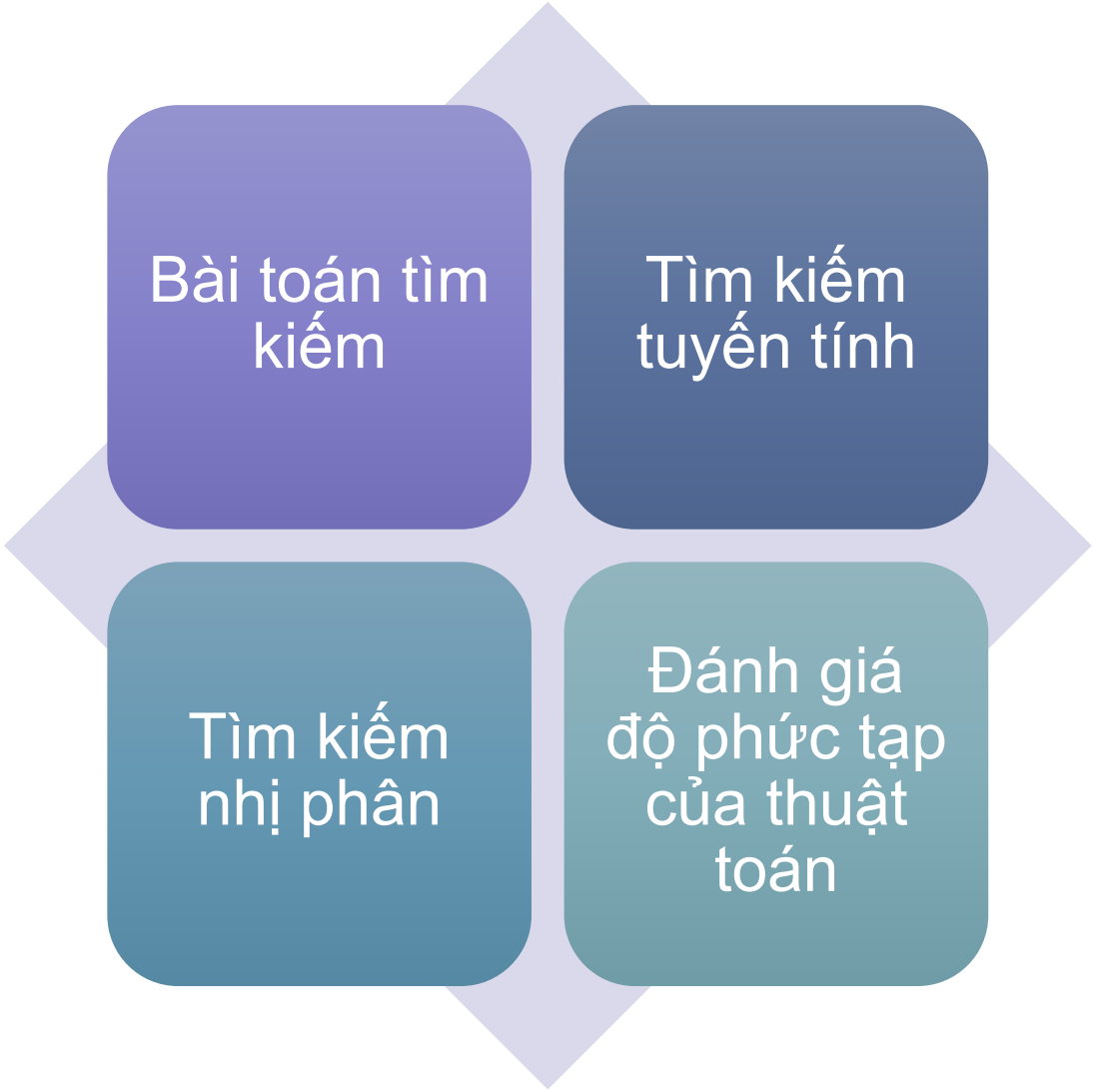
CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

CHƯƠNG 2

CÁC GIẢI THUẬT TÌM KIẾM

GV: Bùi Văn Thượng
Email: thuong.bui@ut.edu.vn

NỘI DUNG CHÍNH



BÀI TOÁN TÌM KIẾM

Tìm kiếm là một đòi hỏi rất thường xuyên trong các ứng dụng tin học. Tìm kiếm có thể định nghĩa là việc thu thập một số thông tin nào đó từ một khối thông tin lớn đã được lưu trữ trước đó.

Bài toán tìm kiếm có thể được phát biểu như sau:

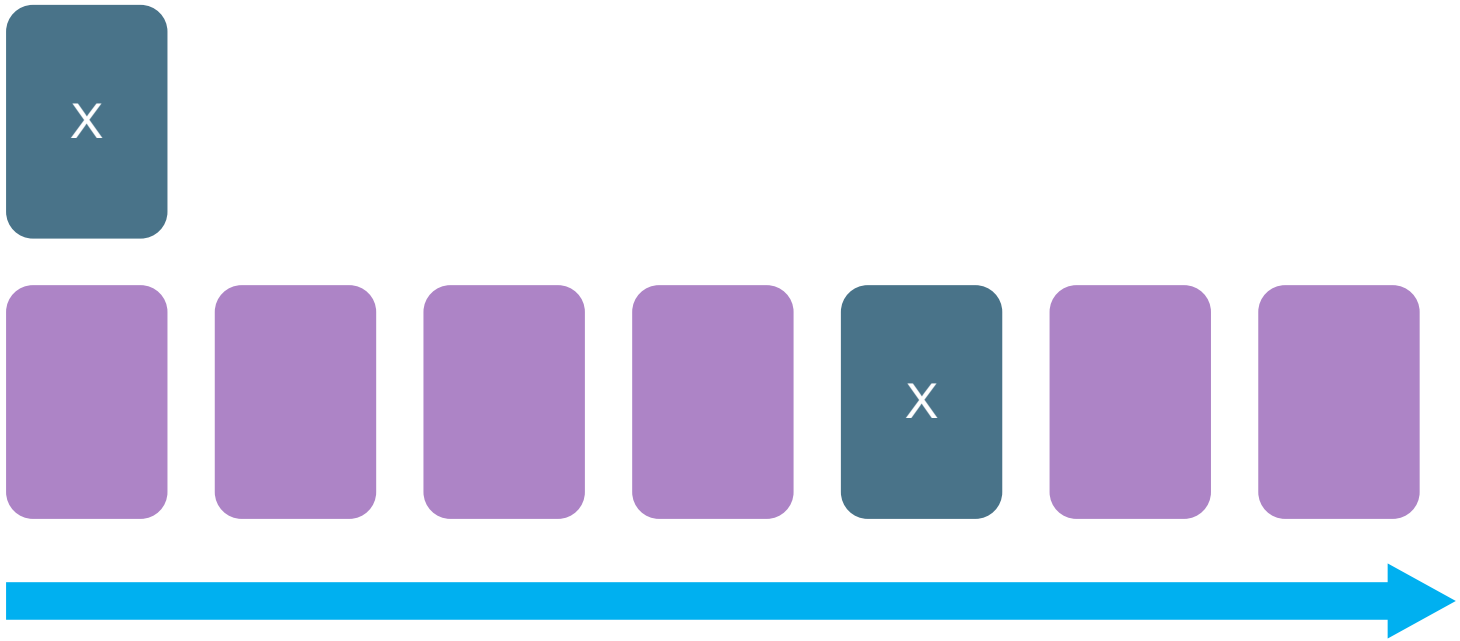
Cho một dãy gồm n bản ghi r[1…n]. Mỗi bản ghi r[i] tương ứng với một khóa k[i]. Hãy tìm bản ghi có khóa X cho trước. X được gọi là khóa tìm kiếm hay đối trị tìm kiếm (argument). Công việc tìm kiếm sẽ hoàn thành nếu như có một trong hai tình huống sau sảy ra:

- Tìm được bản ghi có khóa tương ứng bằng X, khi đó tìm kiếm thành công.
- Không tìm được bản ghi nào có khóa bằng X cả, phép tìm kiếm thất bại.

BÀI TOÁN TÌM KIẾM

Để đơn giản cho việc trình bày giải thuật thì bài toán có thể phát biểu như sau: Cho mảng A[n], hãy tìm trong A phần tử có khóa bằng X.

Hiện nay, có hai giải thuật tìm kiếm thường được sử dụng đó là tìm kiếm tuyến tính và tìm kiếm nhị phân.



GIẢI THUẬT TÌM KIẾM TUYẾN TÍNH

Ý tưởng để tìm phần tử có khóa X trong mảng A bằng giải thuật tìm kiếm tuyến tính như sau:

- Thực hiện tìm kiếm từ đầu cho đến cuối mảng (hoặc ngược lại).
- Nếu tìm thấy trả về vị trí của kết quả tìm kiếm.
- Nếu không tìm thấy trả về -1, bài toán không có giá trị thỏa mãn yêu cầu.

Các bước thực hiện:

- **Bước 1:** Duyệt mảng arr (n phần tử) từ vị trí đầu tiên i = 0.
- **Bước 2:** Thực hiện so sánh giá trị arr[i] và X. Nếu arr[i] == X, trả về i.
- **Bước 3:** Nếu như duyệt hết phần tử của mảng mà vẫn không tìm thấy thì trả về -1, khi đó kết quả của bài toán là không tìm thấy giá trị thỏa mãn yêu cầu.

GIẢI THUẬT TÌM KIẾM TUYẾN TÍNH

Đánh giá:

- Trong trường hợp tốt nhất, phần tử cần tìm nằm ngay ở vị trí đầu tiên, thuật toán sử dụng 1 lần so sánh.
- Trong trường hợp xấu nhất, phần tử cần tìm nằm ngay ở vị trí cuối hoặc không nằm trong mảng, thuật toán cần sử dụng n-1 lần so sánh.

Tìm kiếm tuyến tính là một giải thuật đơn giản nhưng khá hiệu quả với danh sách đủ nhỏ hoặc một danh sách chưa được sắp xếp.

GIẢI THUẬT TÌM KIẾM TUYẾN TÍNH

Hàm minh họa giải thuật tìm kiếm tuyến tính bằng vòng lặp while:

```
int SequentialSearch (int A[], int n, int x){
        int i = 0;
        while ((i<n) && (A[i] != x))
                i++;
        if (i == n) return -1; //Tìm không thấy x
        return i; //Tìm thấy
}
```

GIẢI THUẬT TÌM KIẾM TUYẾN TÍNH

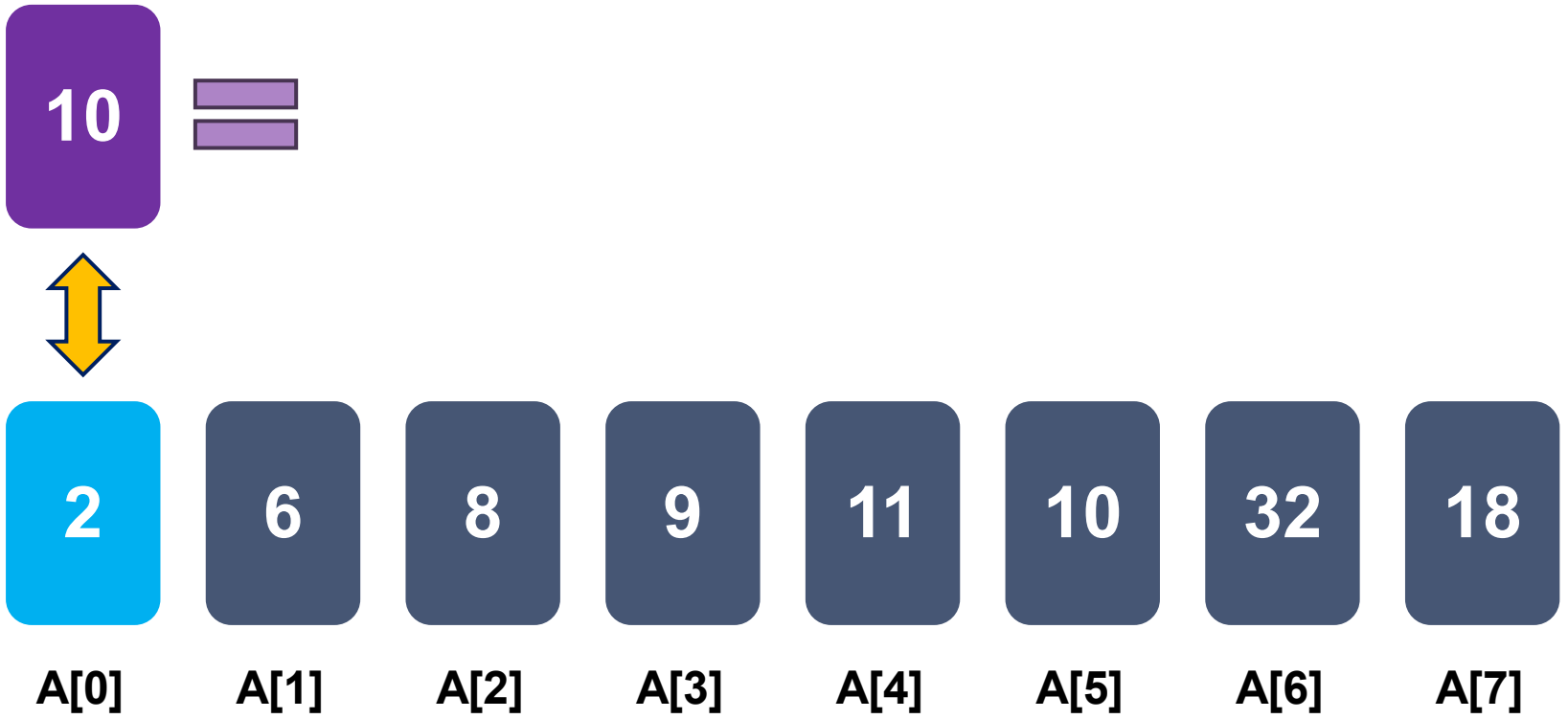
Thuật toán tìm kiếm tuyến tính cũng có thể cài đặt bằng for như sau:

```
int SequentialSearch (int A[], int n, int x){
        for (int i = 0; i < n; i++)
                if (A[i] == x) return i;
        return -1;
}
```

GIẢI THUẬT TÌM KIẾM TUYẾN TÍNH

Ví dụ: Tìm phần tử có giá trị bằng 10 trong mảng số nguyên A:

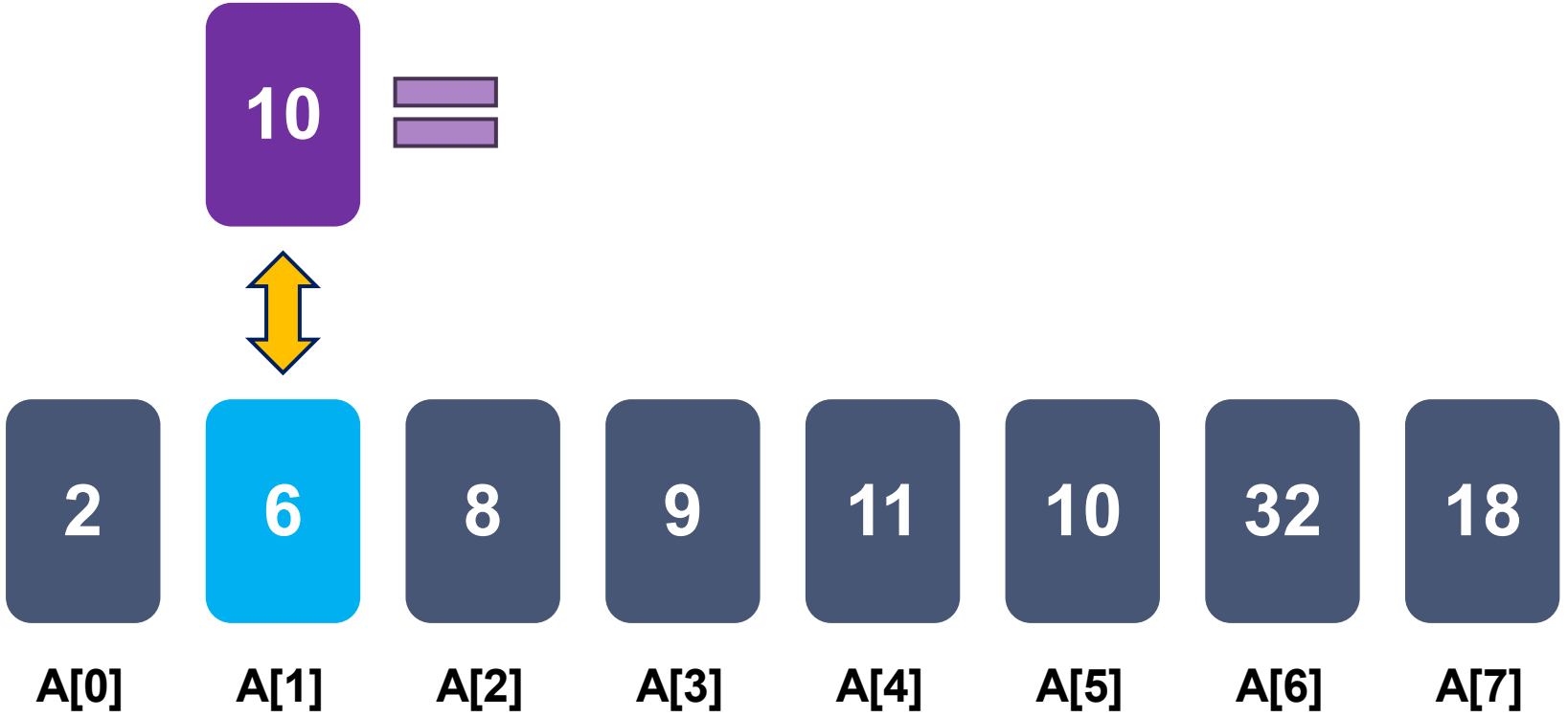
A[] = {2, 6, 8, 9, 11, 10, 32, 18}



GIẢI THUẬT TÌM KIẾM TUYẾN TÍNH

Ví dụ: Tìm phần tử có giá trị bằng 10 trong mảng số nguyên A:

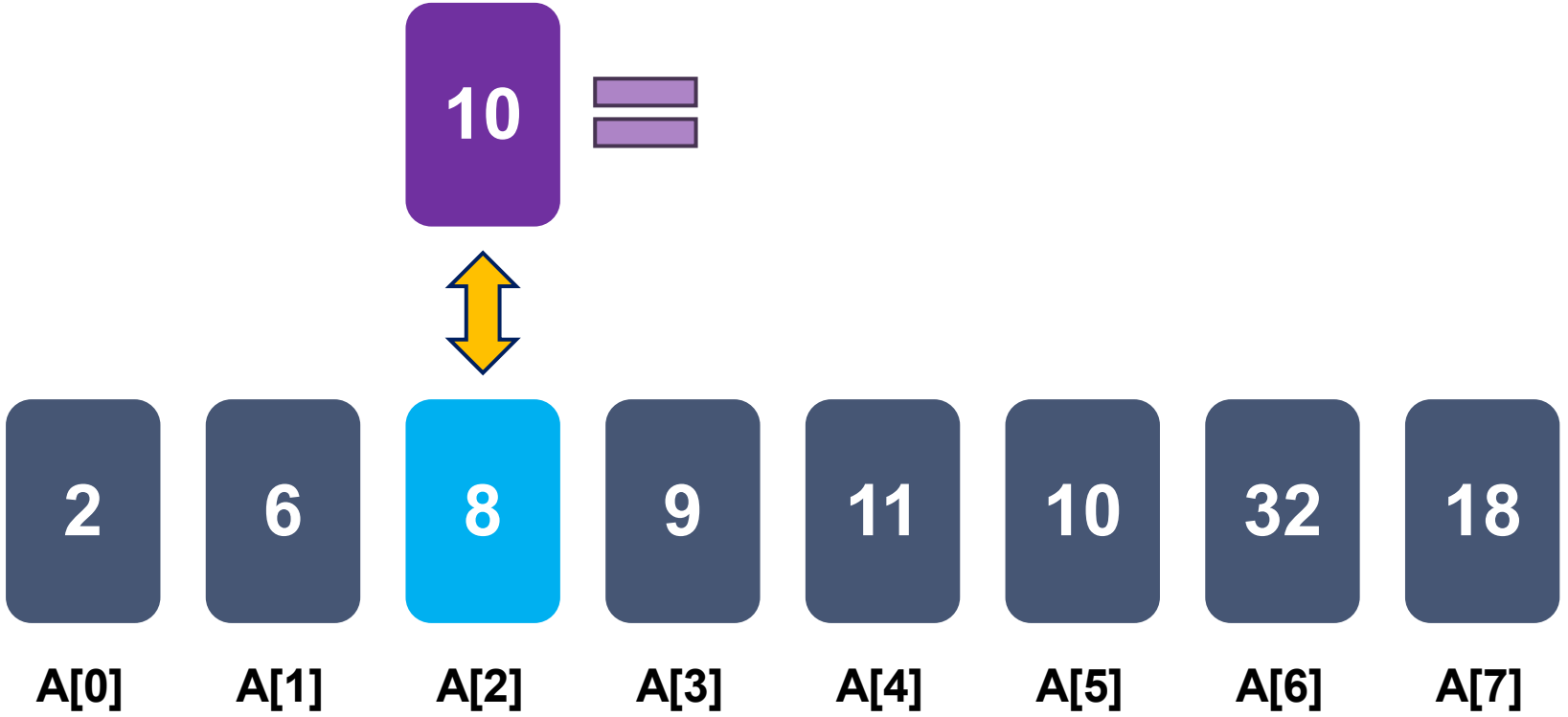
A[] = {2, 6, 8, 9, 11, 10, 32, 18}



GIẢI THUẬT TÌM KIẾM TUYẾN TÍNH

Ví dụ: Tìm phần tử có giá trị bằng 10 trong mảng số nguyên A:

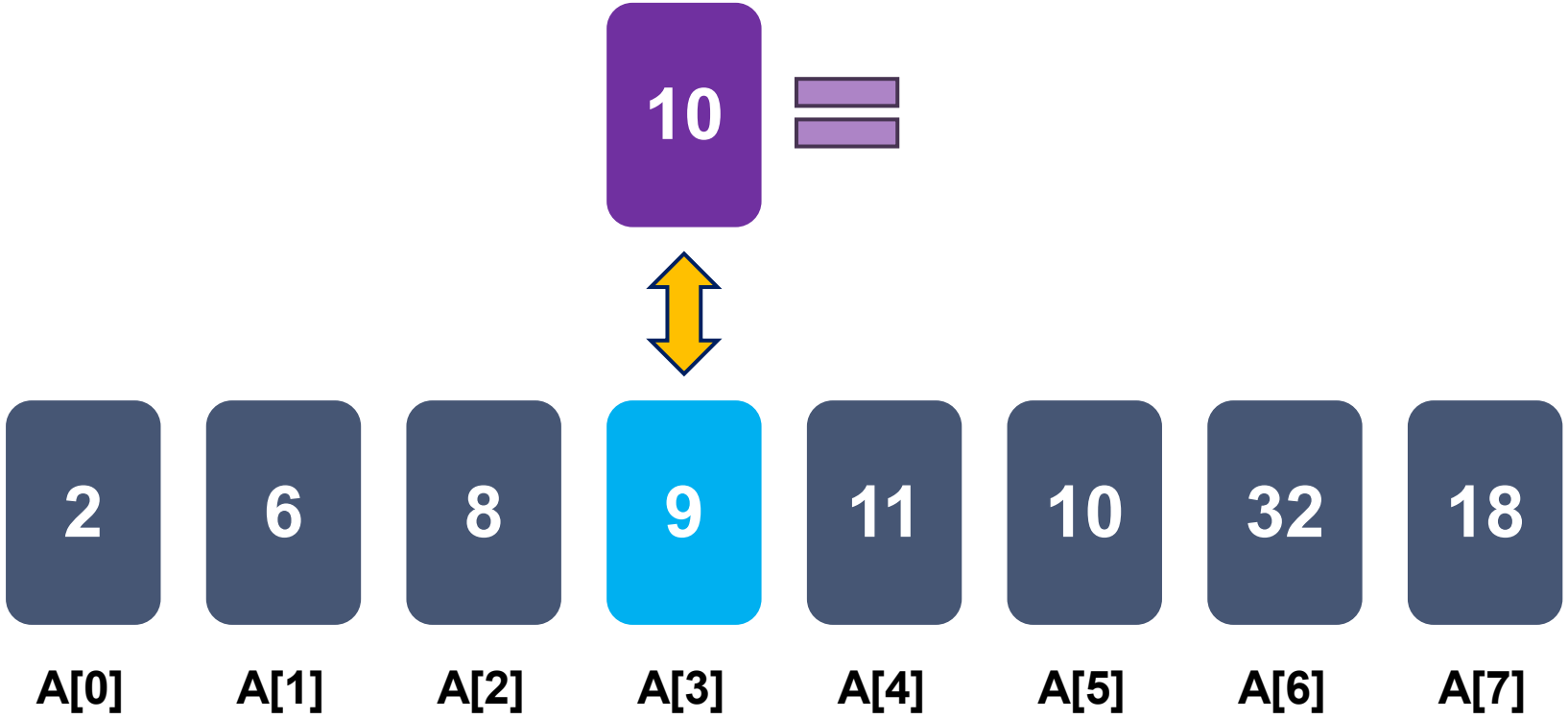
A[] = {2, 6, 8, 9, 11, 10, 32, 18}



GIẢI THUẬT TÌM KIẾM TUYẾN TÍNH

Ví dụ: Tìm phần tử có giá trị bằng 10 trong mảng số nguyên A:

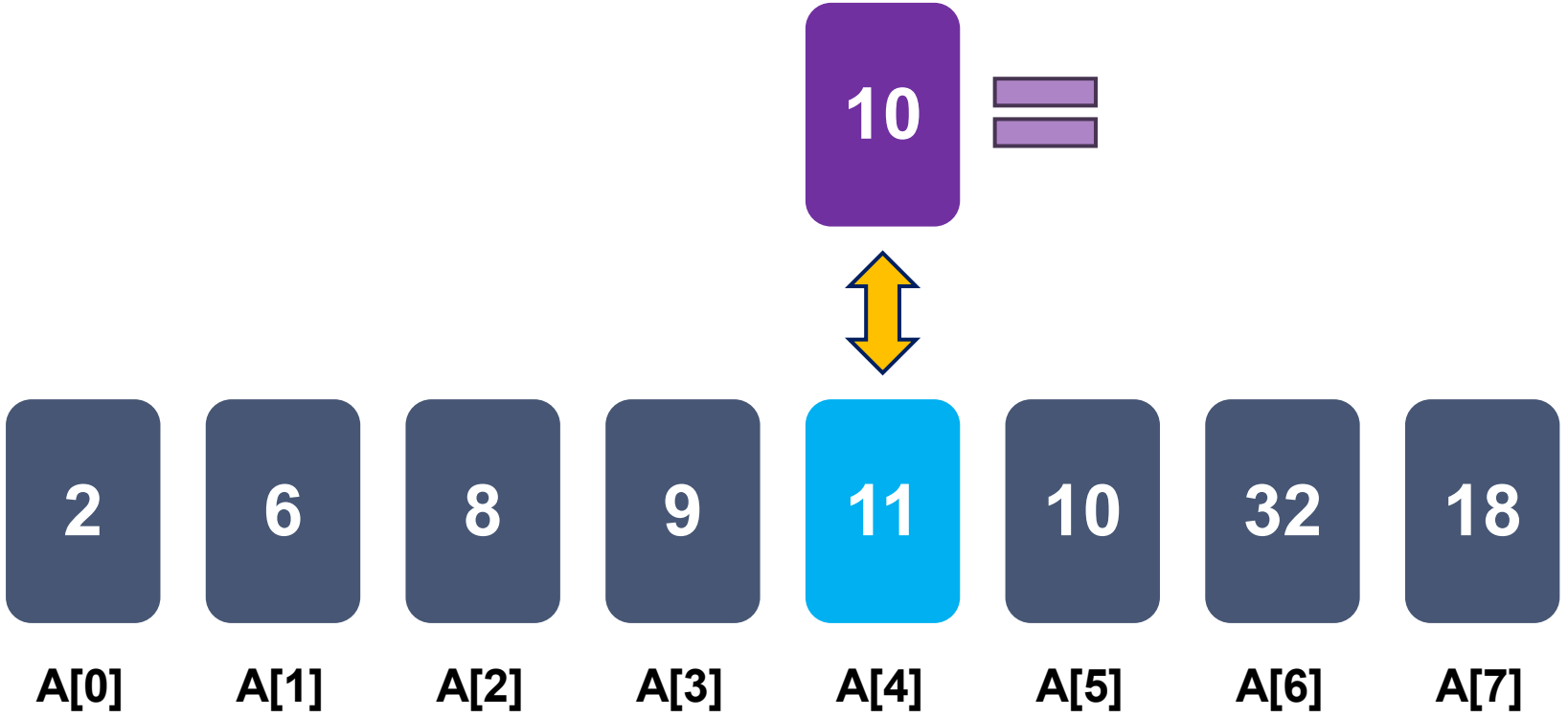
A[] = {2, 6, 8, 9, 11, 10, 32, 18}



GIẢI THUẬT TÌM KIẾM TUYẾN TÍNH

Ví dụ: Tìm phần tử có giá trị bằng 10 trong mảng số nguyên A:

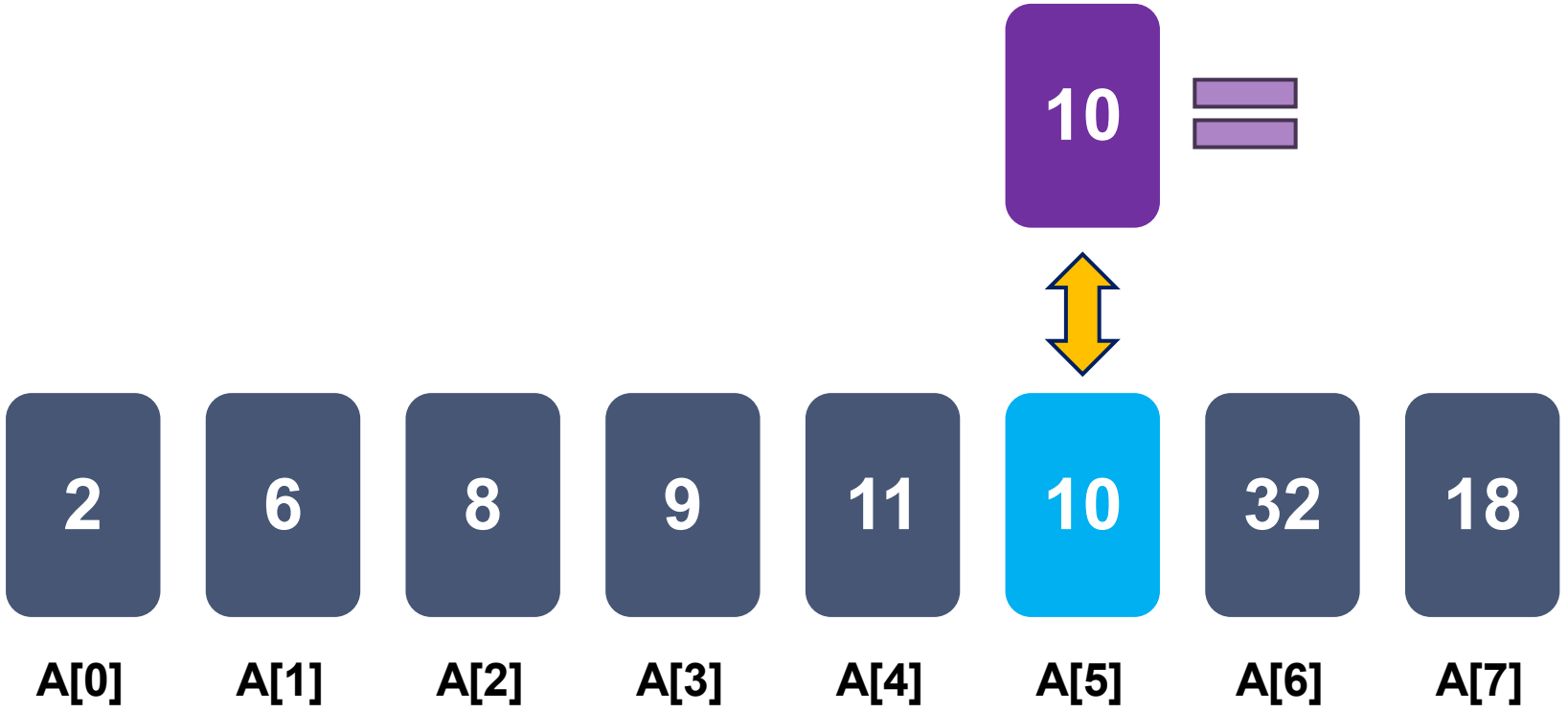
A[] = {2, 6, 8, 9, 11, 10, 32, 18}



GIẢI THUẬT TÌM KIẾM TUYẾN TÍNH

Ví dụ: Tìm phần tử có giá trị bằng 10 trong mảng số nguyên A:

A[] = {2, 6, 8, 9, 11, 10, 32, 18}



GIẢI THUẬT TÌM KIẾM TUYẾN TÍNH

Ví dụ: Tìm phần tử có giá trị bằng 10 trong mảng số nguyên A:

A[] = {2, 6, 8, 9, 11, 10, 32, 18}

GIẢI THUẬT TÌM KIẾM NHỊ PHÂN

Giải thuật tìm kiếm nhị phân (Binany Search) được áp dụng trên mảng đã sắp xếp thứ tự.

Giả sử mảng A[n] được sắp xếp tăng dần, khi đó ta có:

$$A[i-1] < A[i] < A[i+1]$$

Như vậy nếu X>A[i] thì X chỉ có thể nằm trong đoạn [A[i+1], A[n-1]], còn nếu X<A[i] thì X chỉ có thể nằm trong đoạn [A[0], A[i-1]].

Ý tưởng của giải thuật là tại mỗi bước ta so sánh X với phần tử đứng giữa trong dãy tìm kiếm hiện hành, dựa vào kết quả so sánh này mà ta quyết định giới hạn dãy tìm kiếm ở nửa dưới hay nửa trên của dãy tìm kiếm hiện hành.

GIẢI THUẬT TÌM KIẾM NHỊ PHÂN

Hàm minh họa giải thuật tìm kiếm nhị phân:

```
bool BinarySearch (int a[], int n, int x) {
        int left, right, mid;
                left = 0;  right = n-1;
        do {
                mid = (left+right)/2;
                if (a[mid] == x)   return true;
                else
                        if (a[mid]<x)   left = mid+1;
                        else   right = mid-1;
        } while (left <= right);
        return false;
}
```

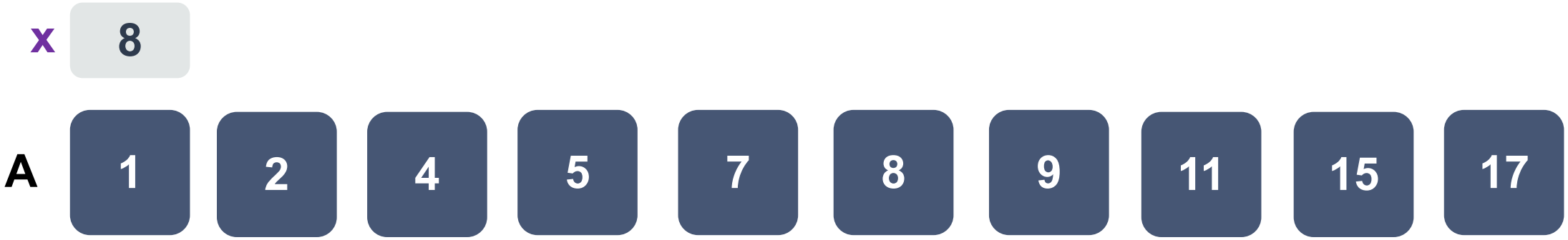
GIẢI THUẬT TÌM KIẾM NHỊ PHÂN

Ví dụ: Tìm phần tử trong mảng sắp xếp.

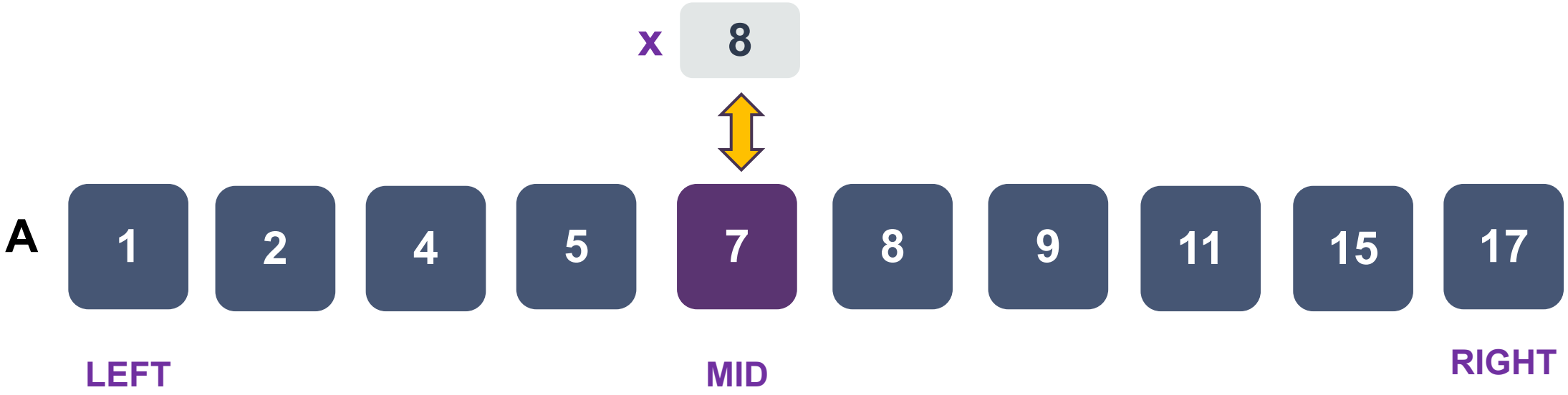
Cho mảng A[] = {1, 2, 4, 5, 7, 8, 9, 11, 15, 17}.

Hãy tìm vị trí của phần tử có giá trị bằng 8.

Vì mảng A đã được sắp xếp tăng dần nên bài toán này có thể áp dụng giải thuật tìm kiếm nhị phân để giải.

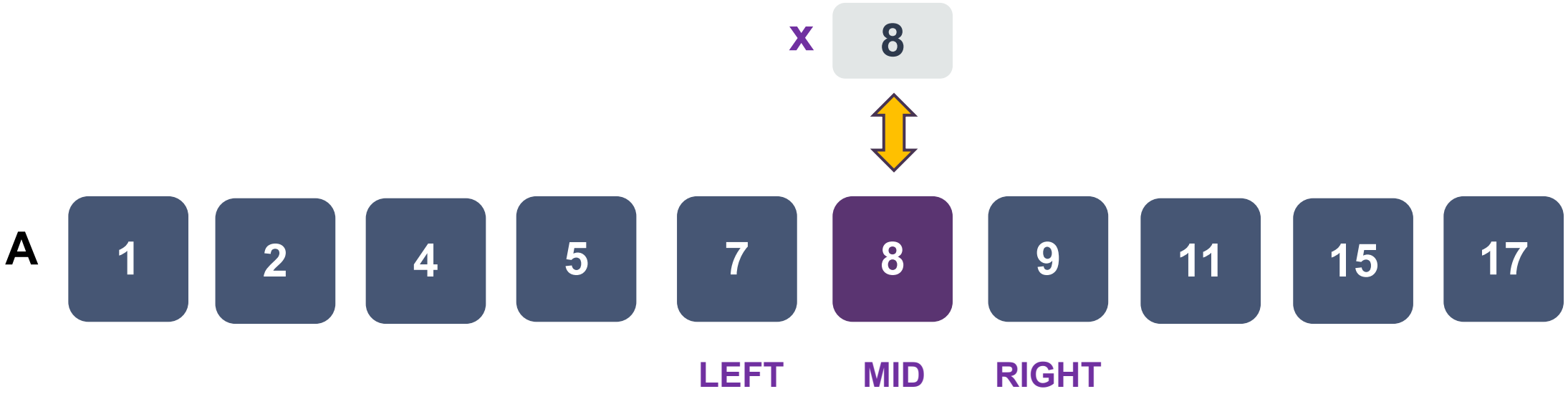


GIẢI THUẬT TÌM KIẾM NHỊ PHÂN

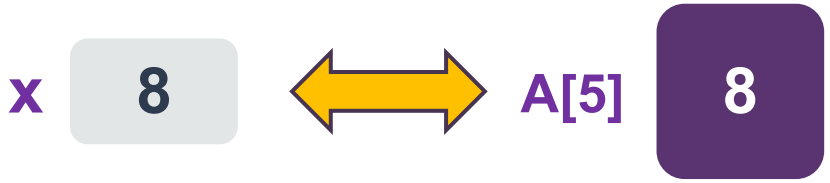


GIẢI THUẬT TÌM KIẾM NHỊ PHÂN

GIẢI THUẬT TÌM KIẾM NHỊ PHÂN



GIẢI THUẬT TÌM KIẾM NHỊ PHÂN



KẾT QUẢ: A[5] = 8

ĐÁNH GIÁ ĐỘ PHỨC TẠP CỦA THUẬT TOÁN

Tìm kiếm tuyến tính

- Giải thuật tìm kiếm tuyến tính tuy rất đơn giản và dễ cài đặt, tuy nhiên nhược điểm của nó nằm ở độ phức tạp.
- Trong trường hợp tốt nhất, giải thuật tìm kiếm tuyến tính có độ phức tạp là O(1), nhưng trong trường hợp xấu nhất độ phức tạp lên tới O(n). Vì vậy độ phức tạp tổng quát của giải thuật là O(n).
- Giải thuật tìm kiếm tuyến tính chỉ phù hợp với những bài toán có kích thước không gian tìm kiếm nhỏ.

ĐÁNH GIÁ ĐỘ PHỨC TẠP CỦA THUẬT TOÁN

Tìm kiếm nhị phân

- Trong trường hợp tốt nhất, giải thuật tìm kiếm nhị phân cho ta độ phức tạp O(1). Còn trong trường hợp xấu nhất, do tập tìm kiếm luôn luôn được chia đôi ra, nên số thao tác chỉ mất $O(\log_2(n))$. Vì thế, độ phức tạp tổng quát của giải thuật là $O(\log_2(n))$.
- Tuy nhiên, giải thuật tìm kiếm nhị phân chỉ có thể thực hiện trên một tập đã sắp xếp, chính vì thế chi phí sắp xếp cũng cần được tính đến.
- Nếu như dãy số bị thay đổi bởi các thao tác thêm, xóa hay sửa phần tử, thì việc sắp xếp cũng phải thực hiện lại liên tục, từ đó dẫn đến thời gian thực thi bị tăng lên.

ĐÁNH GIÁ ĐỘ PHỨC TẠP CỦA THUẬT TOÁN

Bảng so sánh độ phức tạp của các giải thuật tìm kiếm

Giải thuật	Trường hợp		
	Tốt nhất	Trung bình	Xấu nhất
Tìm kiếm tuyến tính	O(1)	O((N+1)/2)	O(N)
Tìm kiếm nhị phân	O(1)	$O(\log_2 N/2)$	$O(\log_2 N)$

BÀI TẬP CHƯƠNG 2

Câu 1. Viết chương trình nhập dữ liệu cho mảng số nguyên A[n], với 0<n<100. Hãy tìm trong A các phần tử là số lẻ và lưu vào mảng B.

Câu 2. Nhập dữ liệu và sắp xếp mảng A[n] tăng dần. Sử dụng giải thuật tìm kiếm nhị phân để tìm phần tử có giá trị bằng X ở trong mảng A sau đó xóa nó khỏi A nếu tìm thấy.

Câu 3. Nhập vào một chuỗi S bất kì. Đếm xem trong chuỗi S có bao nhiêu kí tự khoảng trống, bao nhiêu kí tự số, bao nhiêu kí tự là chữ cái in hoa ?

Câu 4. Viết chương trình xóa phần tử có giá trị bằng nhau trong mảng và giữ lại phần từ đâu tiên trong những phần tử trùng nhau đó