Data Structures and Algorithms

MSc. KhangVQH

Computer Science

Lecture #2

Searching Techniques

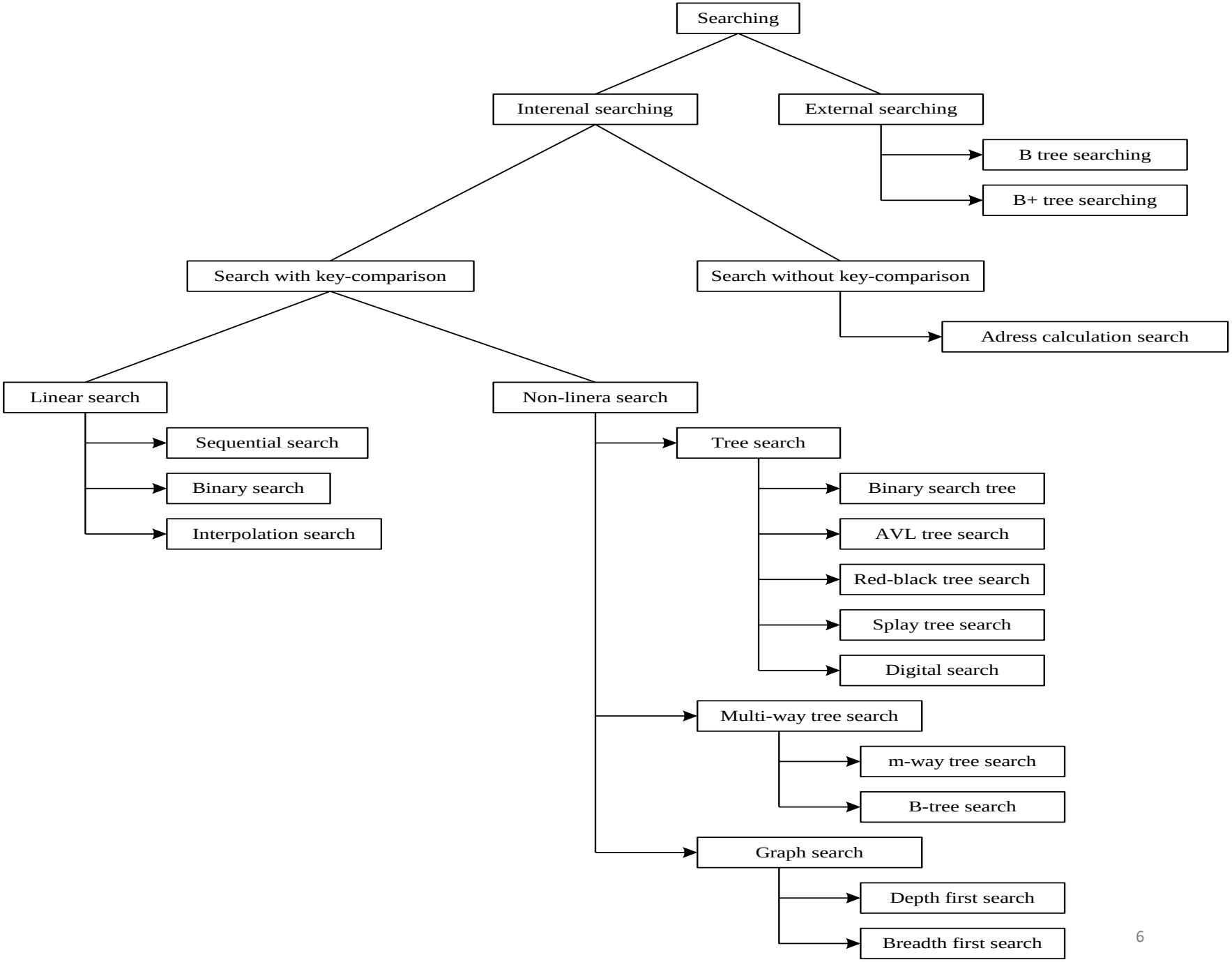
Today’s discussion…

- Searching Techniques
- Sequential search with arrays
- Binary search with arrays

Searching Techniques

Searching Techniques

- Tìm kiếm là một nhu cầu rất thường xuyên trong đời sống hàng ngày cũng như trong tin học.
- Tìm kiếm có trong hầu hết trong các hệ thống thông tin
- Ví dụ:
  - Tìm kiếm một sinh viên trong lớp
  - Tìm kiếm một tập tin, thư mục trong máy tính
  - …


Searching
Interenal searching
External searching
B tree searching
B+ tree searching
Search with key-comparison
Search without key-comparison
Adress calculation search
Linear search
Non-linera search
Sequential search
Binary search
Interpolation search
Tree search
Binary search tree
AVL tree search
Red-black tree search
Splay tree search
Digital search
Multi-way tree search
m-way tree search
B-tree search
Graph search
Depth first search
Breadth first search

Linear Search

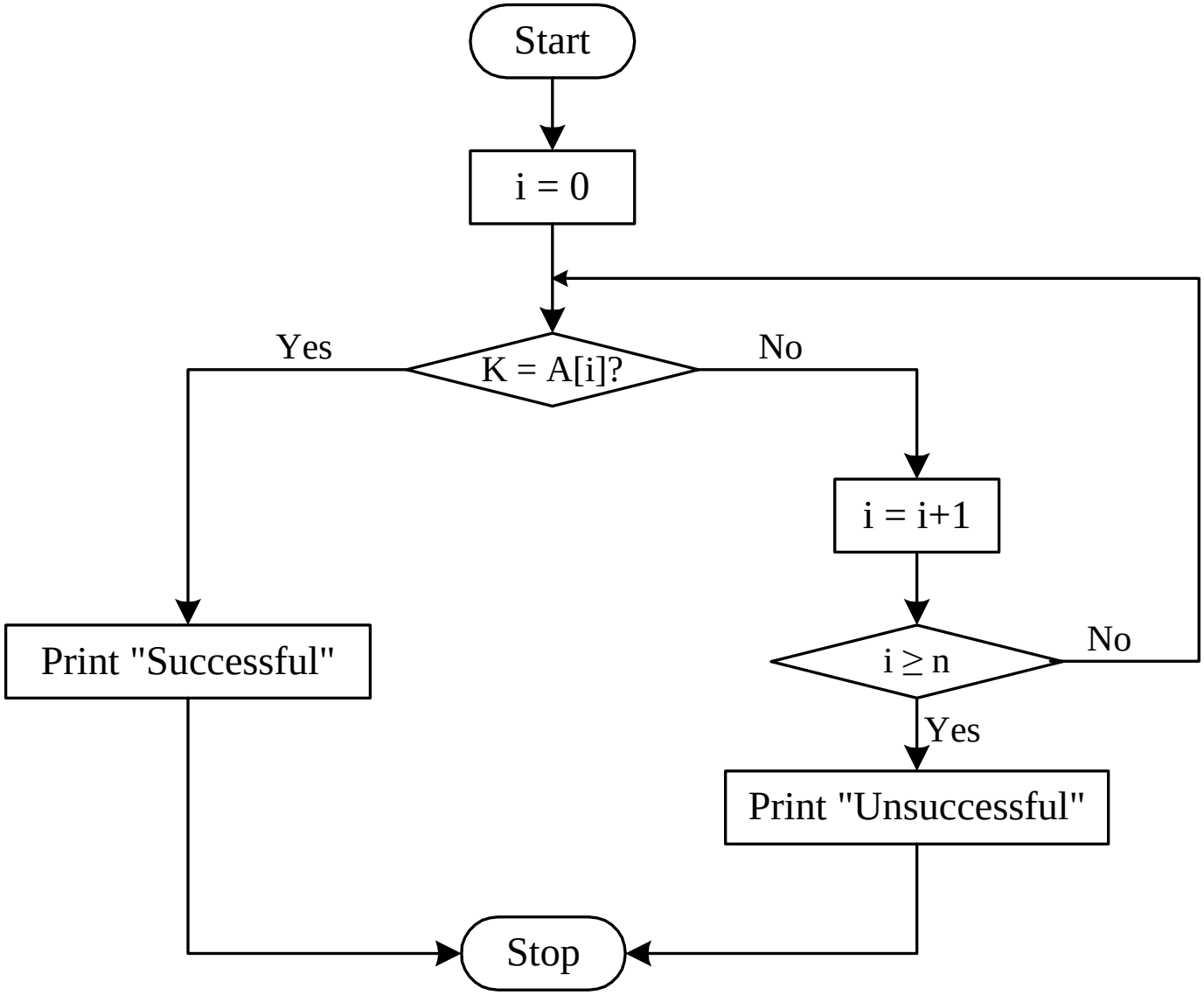
Sequential Search with Arrays

Tìm tuyến tính (Linear Seach)

The Technique: Bắt đầu từ phần tử đầu tiên của mảng, so sánh lần lượt từng phần tử với giá trị K cần tìm.

- Nếu có phần tử bằng K, thuật toán dừng lại (tìm thấy)
- Nếu đến cuối mảng mà không có phần tử nào bằng X, thuật toán dừng lại (không tìm thấy)

Flowchart: Sequential Search with Array



Example: Sequential Search with Array

```
int main()
{
    int A[10], i, n, K, flag = 0;
    printf("Enter the size of an array: ");
    scanf("%d",&n);

    printf("Enter the elements of the array: ");
    for(i=0; i < n; i++)
         scanf("%d",&A[i]);
    printf("Enter the number to be searched: ");
    scanf("%d",&K);
    for(i=0;i<n;i++){
         if(a[i] == K){
             flag = 1; break;
         }
    }
    if(flag == 0)
        printf("The number is not in the list");
    else
        printf("The number is found at index %d",i);
    return 0;
}
```

Complexity Analysis

- Case 1: The key matches with the first element
  - T(n) = 1
- Case 2: Key does not exist
  - T(n) = n
- Case 3: The key is present at any location in the array

$$T(n) = \sum_{i=1}^{n} p_i \cdot i$$

$$T(n) = \frac{1}{n}\sum_{i=1}^{n} i \qquad p_1 = p_2 = \cdots p_i = \cdots p_n = \frac{1}{n}$$

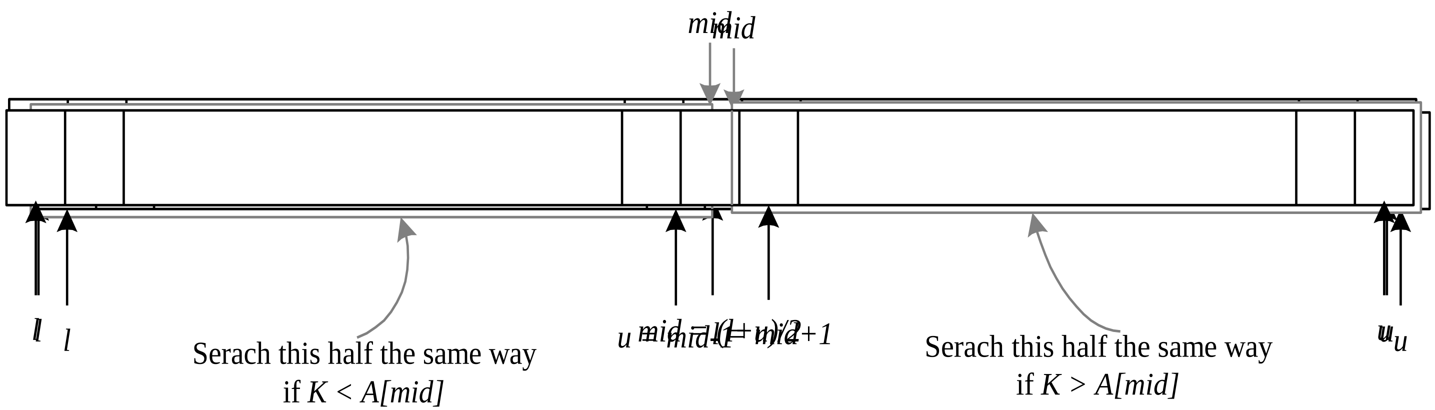
$$T(n) = \frac{n+1}{2}$$

Complexity Analysis : Summary

Case	Number of key comparisons	Asymptotic complexity	Remark
Case 1	$T(n) = 1$	$T(n) = O(1)$	Best case
Case 2	$T(n) = n$	$T(n) = O(n)$	Worst case
Case 3	$T(n) = \frac{n+1}{2}$	$T(n) = O(n)$	Average case

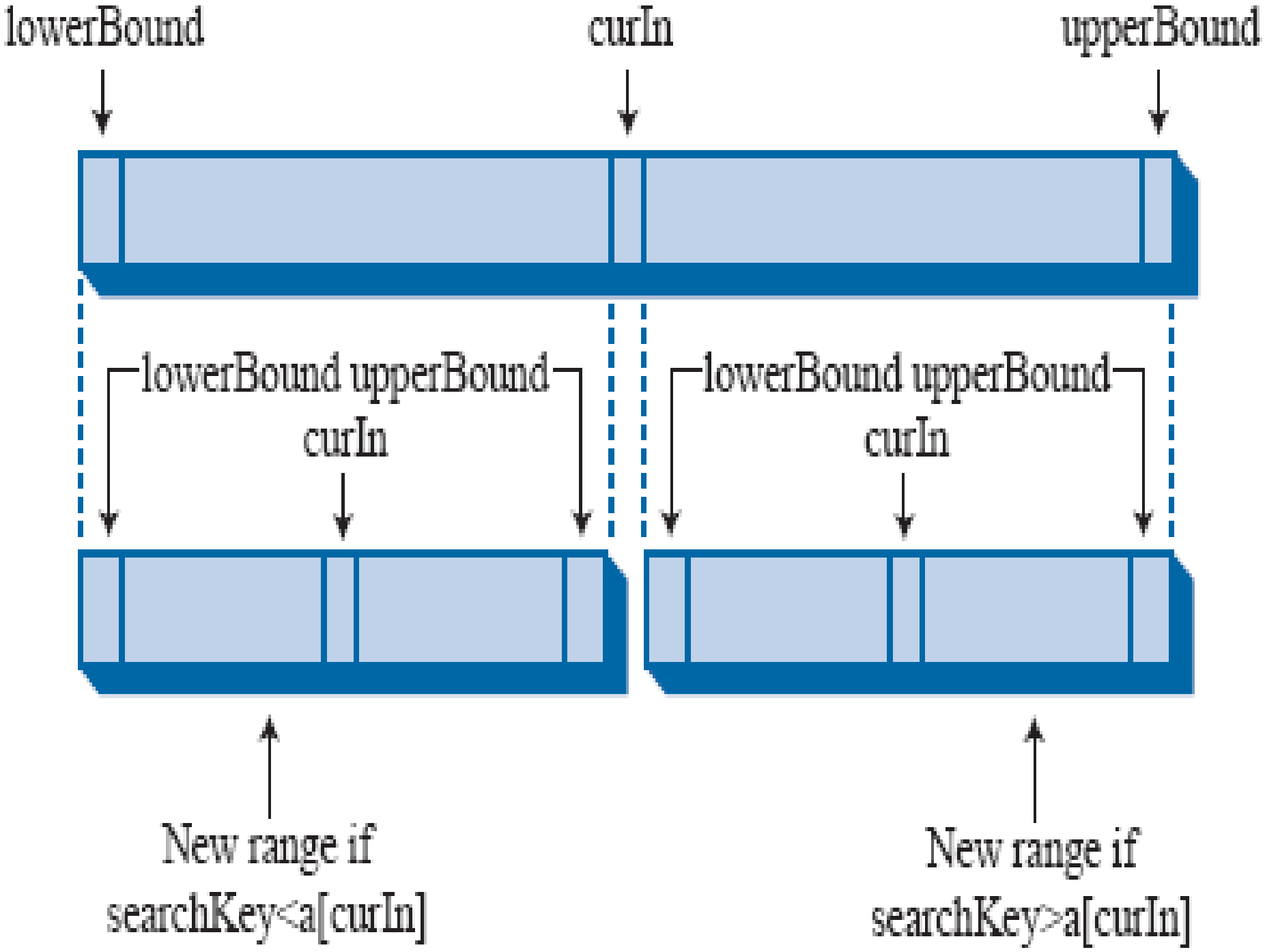
Binary Search

The Technique

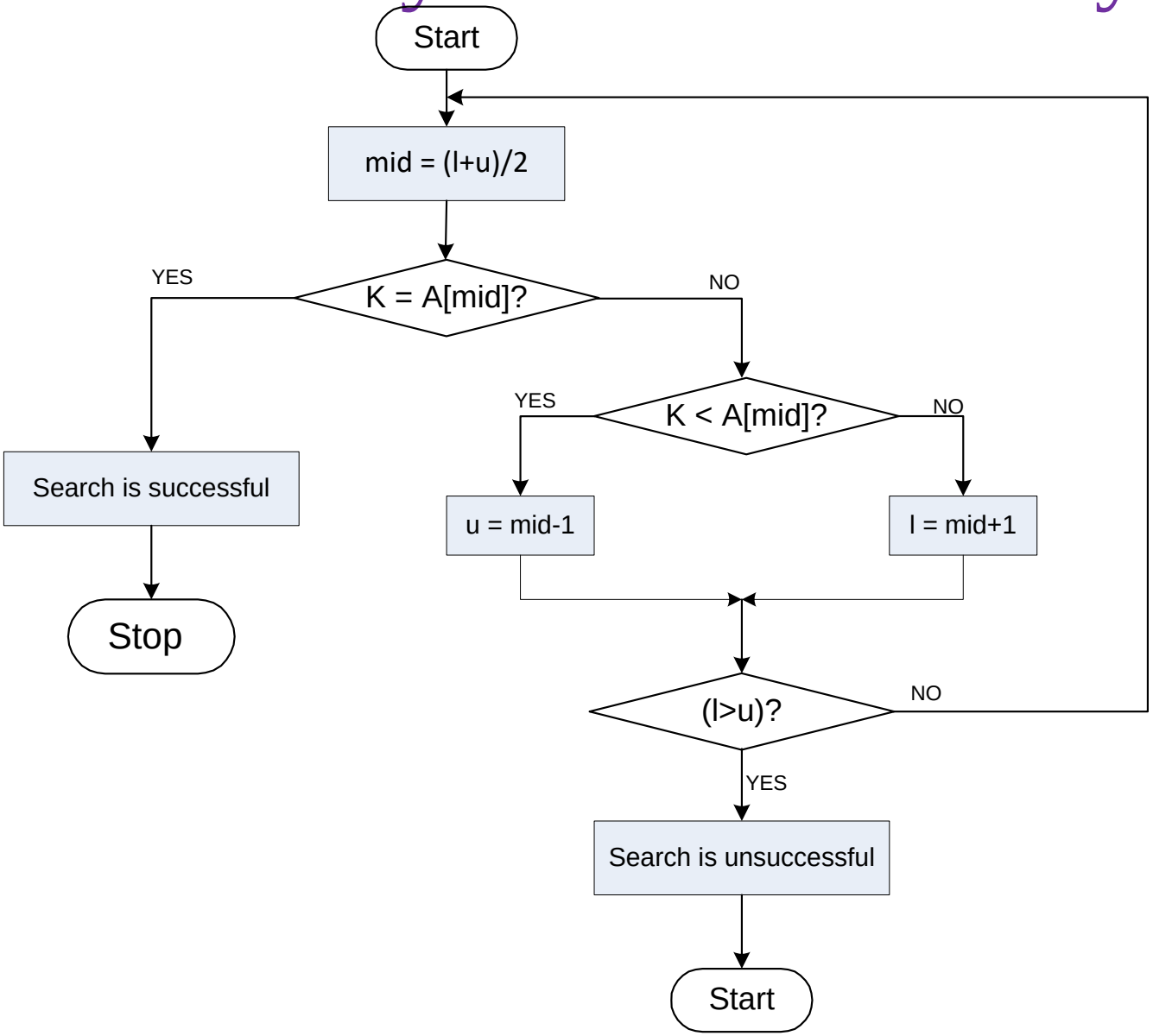


(c) Search the entire list turns into the searching of right-half only

(b) Search the entire list turns into the searching of left-half only


lowerBound
curIn
upperBound
lowerBound upperBound
curIn
lowerBound upperBound
curIn
New range if
searchKey<a[curIn]
New range if
searchKey>a[curIn]

Flowchart: Binary Search with Array



Binary Search (with Iteration)

```
#include <stdio.h>

int main()
{
   int i, l, u, mid, n, K, data[100];

   printf("Enter number of elements\n");
   scanf("%d",&n);

   printf("Enter %d integers in sorted order\n", n);

   for (i = 0; i < n; i++)
      scanf("%d",&array[i]);

   printf("Enter value to find\n");
   scanf("%d", &K);

   l = 0;
   u = n - 1;
   mid = (l+u)/2;
```

Contd…

Binary Search (with Iteration)

```
while (l <= u) {
      if (data[mid] < K)
         l = mid + 1;
      else if (data[mid] == K) {
         printf("%d found at location %d.\n", search, mid+1);
         break;
      }
      else
         u = mid - 1;

      mid = (l + u)/2;
   }
   if (l > u)
      printf("Not found! %d is not present in the list.\n", K);

   return 0;
}
```

Binary Search (with Recursion)

```
#include<stdio.h>
int main(){

    int data[100],i, n, K, flag, l, u;

    printf("Enter the size of an array: ");
    scanf("%d",&n);

    printf("Enter the elements of the array in sorted order: " );
    for(i=0;i<n;i++)
         scanf("%d",&a[i]);

    printf("Enter the number to be search: ");
    scanf("%d",&K);

    l=0,u=n-1;
    flag = binarySearch(data,n,K,l,u);
    if(flag==0)
         printf("Number is not found.");
    else
         printf("Number is found.");

    return 0;
 }
```

Contd…

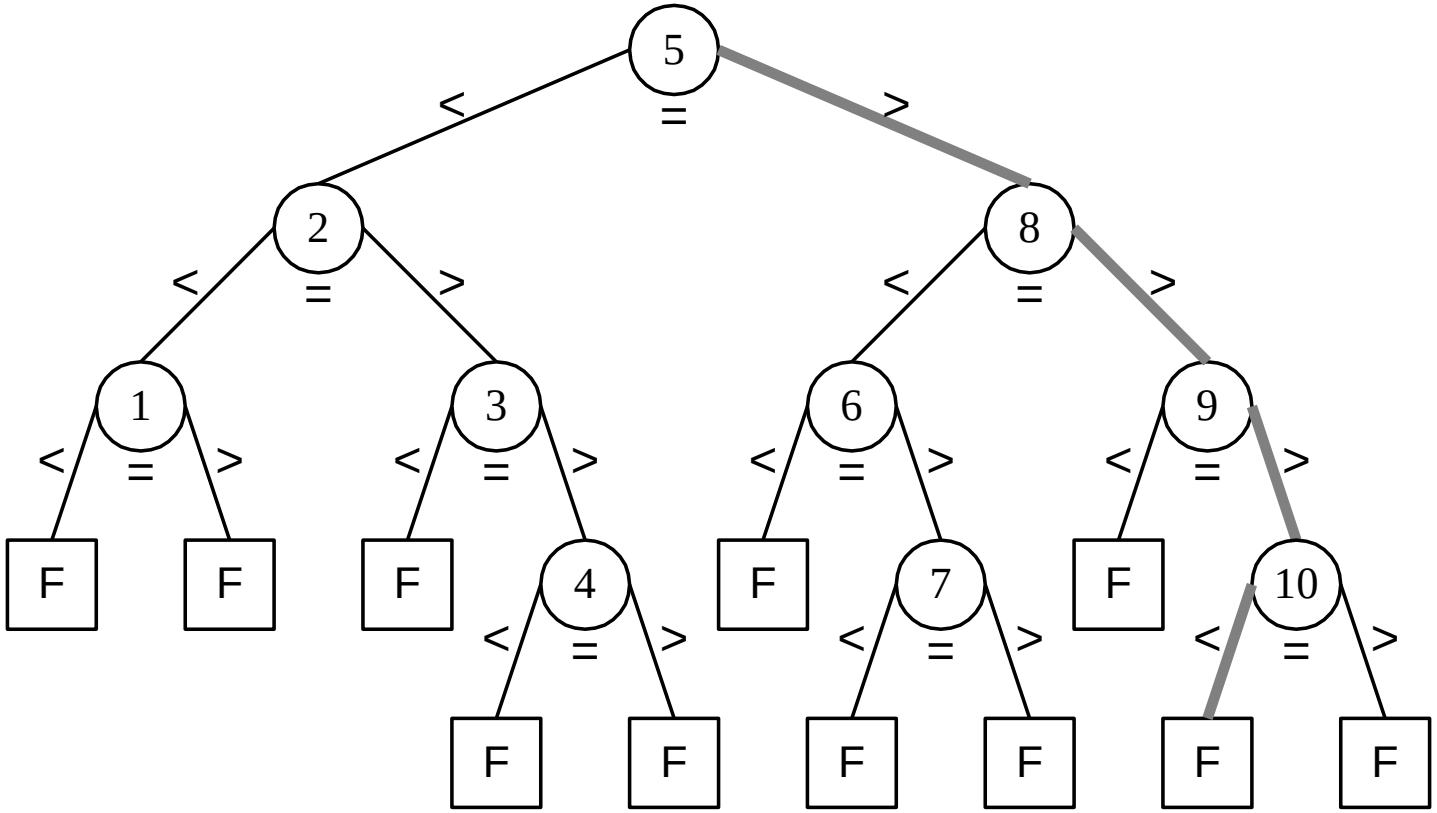
Binary Search (with Recursion)

```
int binary(int a[],int n,int K,int l,int u){

      int mid;

      if(l<=u){
            mid=(l+u)/2;
            if(K==a[mid]){
                return(1);
            }
            else if(m<a[mid]){
                return binarySearch(a,n,K,l,mid-1);
            }
            else
                return binarySearch(a,n,m,mid+1,u);
      }
      else return(0);
}
```

Complexity Analysis



Complexity Analysis

- Best case

  T(n) = 1

- Worst case

  T(n) = $\lfloor \log_2(n+1) \rfloor$

Nhận xét

- Tìm Tuyến Tính không phụ thuộc vào thứ tự của các phần tử, do vậy đây là phương pháp tổng quát nhất để tìm kiếm trên một dãy bất kỳ
- Tìm Nhị Phân dựa vào quan hệ giá trị của các phần tử mảng để định hướng trong quá trình tìm kiếm, do vậy chỉ áp dụng được cho những dãy đã có thứ tự
- Giải thuật Tìm Nhị Phân tiết kiệm thời gian hơn rất nhiều so với giải thuật Tìm Tuyến Tính do:

$$T_{\text{nhị phân}}(n) = O(\log_2 n) < T_{\text{tuyến tính}}(n) = O(n)$$

COMPLEXITY OF ALGORITHMS

- **Linear Search**
  - **–O(n).**
- **Binary Search**
  - **–O(log2 N)**

n	$\log_2 n$
10	3
100	6
1,000	9
10,000	13
100,000	16

Any question?

Problems to ponder…

1. What will be the time complexity of linear search with array if the array is already in sorted order?
2. What will be the outcome, if Binary search technique is applied to an array where the data are not necessarily in sorted order?
3. Whether binary search technique can be applied to search a string in a list od strings stored in an array? If so, revise the Binary search algorithm accordingly.

Problems to ponder…

4. A structure is defined as follows.

```
struct Record {
     int RollNo;
     char name[20];
     struct Date { int dd; int mm; int yy; } dob;
}
```

suppose, 100 records of type `Record` are stored in an array say, `struct Record students[100];`

We have to find the student(s), whose date of birth is given in the form dd/mm/yy. How you can search the array `students`?