DATA STRUCTURES

By

Dr. Yasser Abdelhamid

الجامعة المصرية
للتعلم الإلكتروني الأهلية

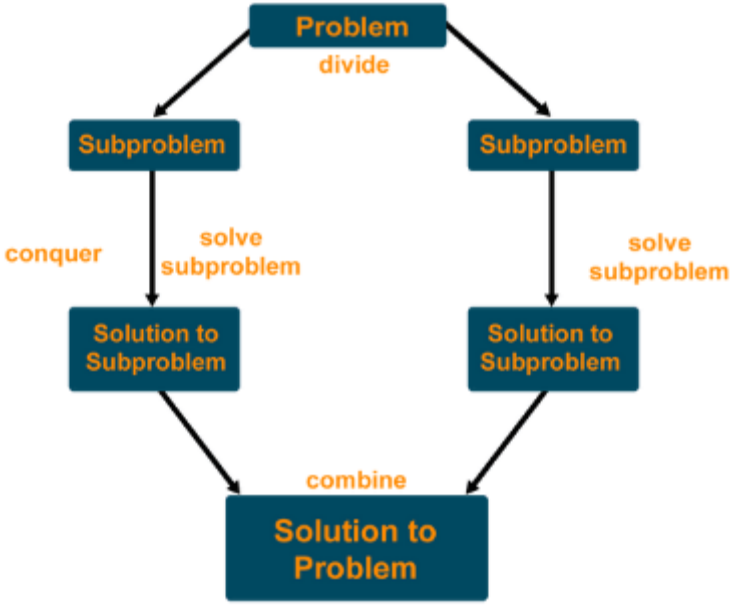
EELU
THE EGYPTIAN E-LEARNING UNIVERSITY

OUTLINE

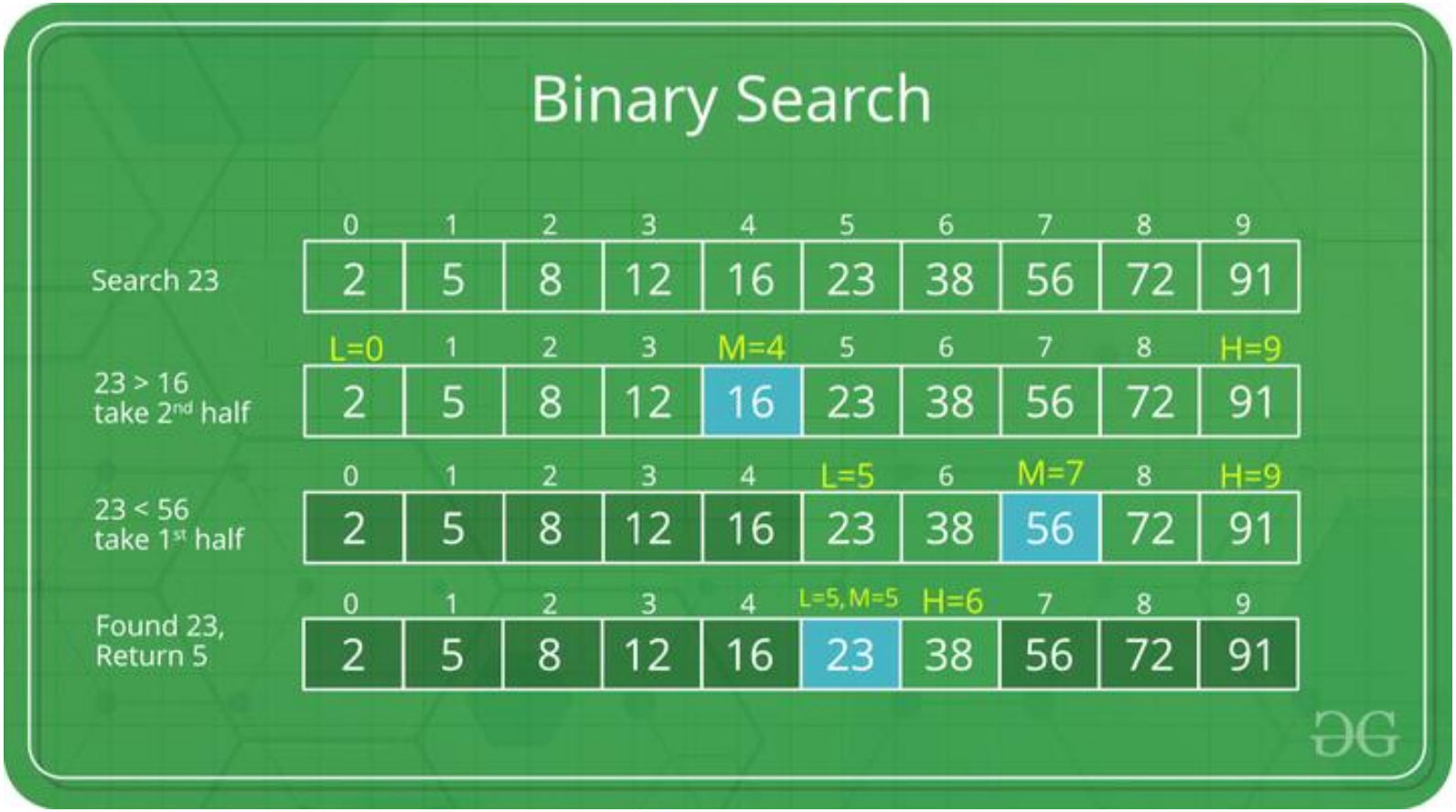
- Searching algorithms Cont.
  - Binary Search
    - Iterative
    - Recursive

BINARY SEARCH

- Binary search algorithm is an efficient algorithm that run in the $O(\log_2 n)$.
- It uses the divide and conquer strategy for solving the search problem.
- It is only applicable to sorted lists.

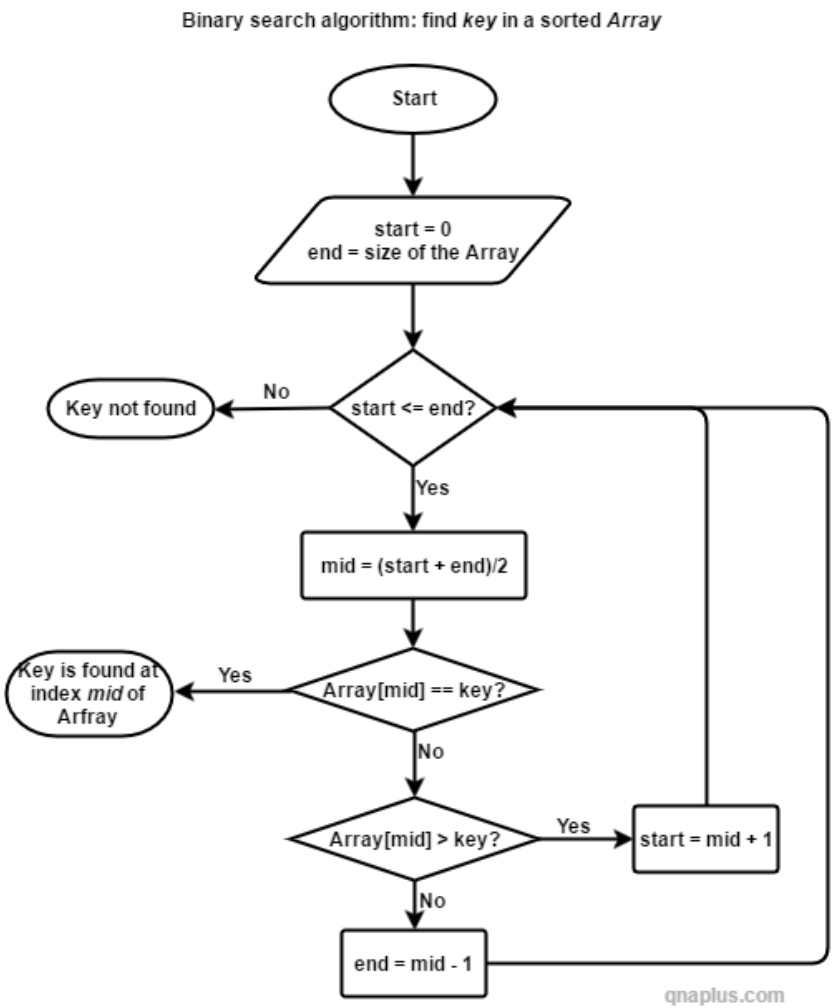


BINARY SEARCH



BINARY SEARCH

- If list is empty, search is complete with fail.
- Compare key to the middle element of list
- If key < middle element of list, search is restricted to first half of the list
- If key > middle element of list, search second half of the list
- If key = middle element, search is complete with success.



BINARY SEARCH ITERATIVE METHOD

```
public static void binarySearch(int arr[], int first, int last, int key){
  int mid = (first + last)/2;
  while( first <= last ){
    if ( arr[mid] < key ){
      first = mid + 1;
    }else if ( arr[mid] == key ){
      System.out.println("Element is found at index: " + mid);
      break;
    }else{
       last = mid - 1;
    }
    mid = (first + last)/2;
  }
  if ( first > last ){
    System.out.println("Element is not found!");
  }
}
```

BINARY SEARCH ALGORITHM PERFORMANCE

- A list of 11 elements takes 4 tries
- A list of 32 elements takes 5 tries
- A list of 250 elements takes 8 tries
- A list of 512 elements takes 9 tries
- $32 = 2^5$ $\log_2(32) = 5$
- $512 = 2^9$ $\log_2(512) = 9$
- $8 < 11 < 16$ $2^3 < 11 < 2^4$
- $128 < 250 < 256$ $2^7 < 250 < 2^8$
- So for a list of n, number of comparisons is in $O(\log_2 n)$

BINARY SEARCH RECURSIVE METHOD

```
public static int binarySearch(int [ ] list, int first, int, last, int key)
{
  if(first > last) return FALSE;                          c1

  mid = (first + last) / 2;                               c2

  if(key==list[mid]) return TRUE;                         c3

  if(key<list[mid])
    return binarySearch(list, first, mid-1, key);         T(n/2)

  if(key>list[mid])
    return binarySearch(list, mid+1, last, key);          T(n/2)
}
```

RECURRENCE EQUATION

- The worst case in Binary search occurs when we keep dividing the li st until we get only one element in the list, and either it matches our key value or not.
- $T(n) = c_1+c_2+c_3+T(n/2)$
- $T(n) = c + T(n/2)$
- By recursively expanding T(n)
  - $T(n) = c + T(n/2)$
  - $= c + (c+T(n/4))$
  - $= c + (c + (c + T(n/8)))$
  - ...
  - $= c.i + T(\frac{n}{2^i})$ eq.(1)
- We reach T(1) when $(\frac{n}{2^i}) = 1$ i.e. when $i = \log_2 n$
- By Replacing i in the eq.(1)
- $T(n) = c.\log_2 n + T(1)$ that asymptotes to $O(\log_2 n)$

THANK YOU

الجامعة المصرية
للتعلم الإلكتروني الأهلية

EELU
THE EGYPTIAN E-LEARNING UNIVERSITY