Session 31: Searching Algorithms

- Linear Search Algorithm
- Binary Search Algorithm

Searching Problems

Task: Given a list $S = a_1, a_2, a_3, \ldots, a_n$ of distinct elements and some x, if $x \in S$ return i such that $a_i = x$, else return 0.

Examples

- Find a word in a dictionary
- Find a name in a customer table
- Find an amount in a bank transaction table

Linear Search Algorithm

The linear search algorithm locates an item in a list by examining elements in the sequence one at a time, starting at the beginning.

Algorithm

1. First compare *x* with a_1. If they are equal, return the position 1.
2. If not, try a_2. If $x = a_2$, return the position 2.
3. Keep going, and if no match is found when the entire list is scanned, return 0.

Linear Search Algorithm

```
procedure linear search($x$: integer, $a_1, a_2, \ldots, a_n$: distinct integers)
$i := 1$
while ($i \leq n$ and $x \neq a_i$)
    $i := i + 1$
    if $i \leq n$ then $location := i$ else $location := 0$
return $location$
```

Example

Sequence 3 5 1 7 2 1

Searching x = 2

$x \neq a_i$

location

Searching x = 4

$x \neq a_i$

location

Binary Search

Assume the input is a list of items in increasing order.

- The algorithm begins by comparing the element to be found with the middle element.
    - If the middle element is lower, the search proceeds with the upper half of the list.
    - If it is not lower, the search proceeds with the lower half of the list (including the middle position).
- Repeat this process until we have a list of size 1.
    - If the element we are looking for is equal to the element in the list, the position is returned.
    - Otherwise, 0 is returned to indicate that the element was not found.

Example

Binary search for 19 in the list:

1 2 3 5 6 7 8 10 12 13 15 16 18 19 20 22

Binary Search

```
procedure binary search($x$: integer, $a_1, a_2, \ldots, a_n$: increasing integers)
  $i := 1$                        {i is the left endpoint of interval}
  $j := n$                        {j is right endpoint of interval}
  while $i < j$                   {at least two elements in the list}
      $m := \lfloor (i + j)/2 \rfloor$                 {take the midpoint}
      if $x > a_m$ then $i := m + 1$ else $j := m$
  if $x = a_i$ then location $:= i$ else location $:= 0$
  return location
```

Summary

- Search is a fundamental operation for data
- Linear and Binary Search
- Binary Search is more efficient, but requires sorting