Searching

Rosen 6th ed., §3.1-3.3

Searching

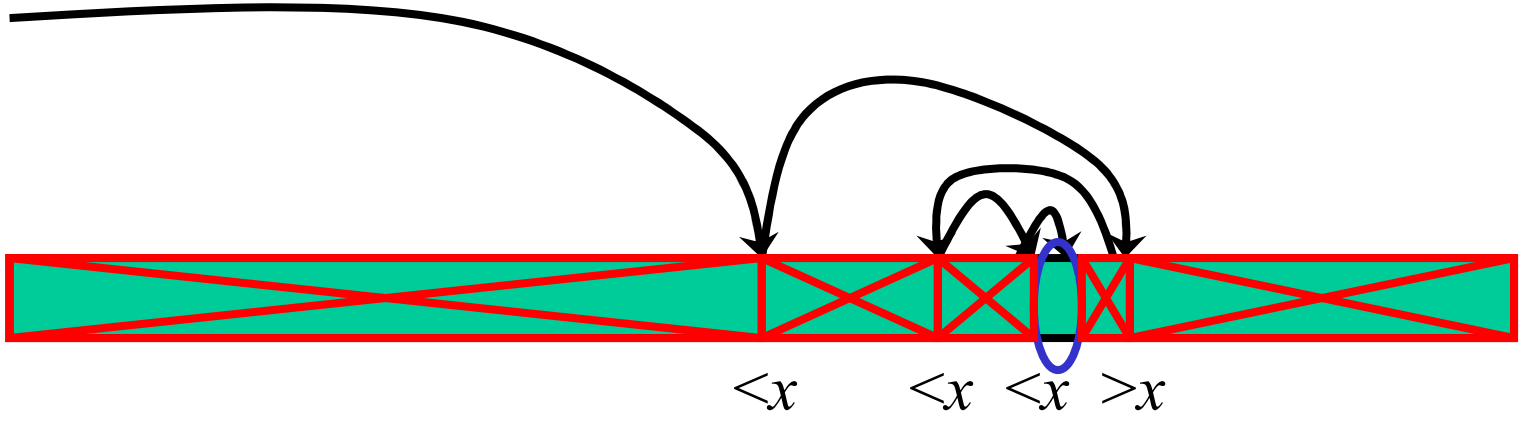
- Problem of *searching an ordered list*.
  - Given a list L of n elements that are sorted into a definite order (*e.g.*, numeric, alphabetical),
  - And given a particular element x,
  - Determine whether x appears in the list, and if so, return its index (position) in the list.

Search alg. #1: Linear Search

function *linear search(x,* ***a****)*
(*x*: integer, $a_1, a_2, \ldots, a_n$: distinct integers)
$i := 1$
while $(i \leq n \wedge x \neq a_i)$ **do**
$i := i + 1$
if $i \leq n$ **then** *location* $:= i$
else *location* $:= 0$
return *location* {index or 0 if not found}

Search alg. #2: Binary Search

- Basic idea: On each step, look at the *middle* element of the remaining list to eliminate half of it, and quickly zero in on the desired element.



Search alg. #2: Binary Search

function *binary search (x,* ***a****)*
(*x*:integer, $a_1, a_2, \ldots, a_n$: distinct integers)
$i := 1$ {left endpoint of search interval}
$j := n$ {right endpoint of search interval}
while $i<j$ **do** {while interval has >1 item}
$m := \lfloor (i+j)/2 \rfloor$ {midpoint}
if $x>a_m$ **then** $i := m+1$ **else** $j := m$
endwhile
if $x = a_i$ **then** *location* $:= i$ **else** *location* $:= 0$
return *location*

Is Binary Search more efficient?

- **Number of iterations**:
  - For a list of n elements, Binary Search can execute at most $\log_2 n$ times!!
  - Linear Search, on the other hand, can execute up to n times !!

Is Binary Search more efficient?

- **Number of computations per iteration**:
  - Binary search does more computations than Linear Search per iteration.
- **Overall**:
  - If the number of components is small (say, less than 20), then Linear Search is faster.
  - If the number of components is large, then Binary Search is faster.