Binary Search

Array A contains element from domain D

$x \in D$

Find out if $x \in A$ (x is present in the array or not?)

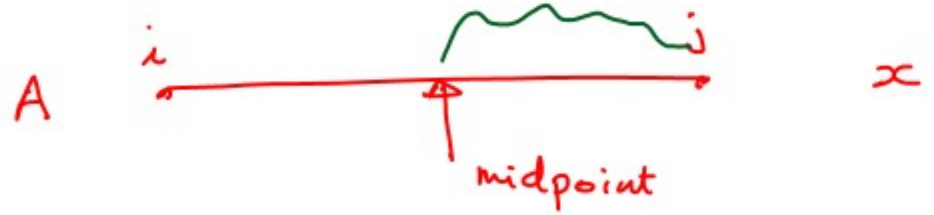
If yes output an i s.t. $A[i] = x$

Search

Search (A, x)
for $i = 1$ to A.length do
  if $A[i] == x$ then (return i)
return ("element not found")

Worst case time
$O(n)$
n : no. of elements in the array.

If A is sorted then search can be made more efficient $O(\log n)$



$A[\text{midpoint}] == x$
$A[\text{midpoint}] < x \Rightarrow$ search $A[\text{midpoint}+1, j]$
$A[\text{midpoint}] > x \Rightarrow$ search $A[i, \text{midpoint}-1]$

PsuedoCode

binsearch (A, i, j, x) // If $x \in A[i, j]$

if $i > j$ then return ("element not found")

// $i \le j$

$k = \left\lfloor \frac{i+j}{2} \right\rfloor$

If $A[k] == x$ then return k

If $A[k] < x$ then return binsearch(A, k+1, j, x)

If $A[k] > x$ then return binsearch(A, i, k-1, x)

Example

$A \quad [1 \quad 3 \quad 7 \quad 9 \quad 10 \quad 15] \qquad x = 12$

- $i = 1, \; j = 6 \qquad k = \lfloor \frac{7}{2} \rfloor = 3$

  $A[k] = 7 < 12$

  call binsearch with $i_1 = 4$, $j_1 = 6$

- $k_1 = \lfloor \frac{i_1 + j_1}{2} \rfloor = 5$, $A[5] = 10 < 12$, call binsearch with $i_2 = 6$, $j_2 = 6$

- $k_2 = \lfloor \frac{6+6}{2} \rfloor = 6 \qquad A[6] = 15 > 12$

  call binsearch with $i_3 = 6$, $j_3 = 5$

- $i_3 > j_3$ so this is an empty segment

  return("element not found").

Example

$A = [1 \quad 3 \quad 7 \quad 9 \quad 10 \quad 12] \qquad x = 9$

$i_1 = 1, \quad j_1 = 6 \qquad k_1 = 3$

$A[k_1] = 7 < 9$

$i_2 = 4, \quad j_2 = 6, \qquad k_2 = 5$

$A[5] = 10 > x$

$i_3 = 4, \quad j_2 = 4, \quad k_2 = 4$

$A[4] = 9 = x \qquad$ return (4) $\qquad A[4] = 9$

$\square$

Correctness

if $x \in A[i,j]$ then binsearch(A, i, j, x) returns ℓ, $i \leq \ell \leq j$, s.t. $A[\ell] = x$

if $x \notin A[i,j]$ then binsearch returns "element not found"

Case I $i > j$: $x \notin A[i,j]$

binsearch returns the correct message

Case II $i \leq j$

we prove correctness in this case by induction on

$j - i + 1$ $|[i, j]|$

Base Case $\quad j - i + 1 = 1 \qquad j = i, \qquad k = i$

(a) if $x \in A[i, j]$ then $A[i] == x$

binsearch returns i

(b) if $x \notin A[i, j]$ then $A[i] < x$ or $A[i] > x$

return binsearch$(A, i+1, i, x)$

returns "element not found"

Induction step $\quad i < j$

Ex $\quad i \leq k < j$

Ex $\quad k - i, \; j - k \leq \left\lfloor \frac{j - i + 1}{2} \right\rfloor$

If $x \in A[i, j]$

(a) $A[k] < x$

$x \in A[k+1, j]$

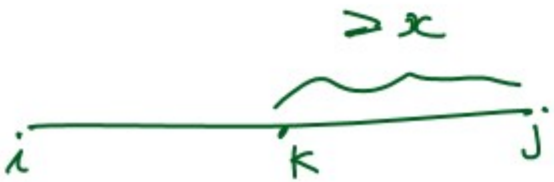
$$j - (k+1) + 1 = j - k < j - i + 1$$

By I.H. binsearch$(A, k+1, j, x)$ returns ℓ

s.t. $A[\ell] = x$, $k+1 \leq \ell \leq j$

$\Rightarrow i \leq \ell \leq j$

If $x \notin A[i, j]$



(a) $A[k] > x$

binsearch$(A, i, k-1, x)$ $\quad k - 1 - i + 1 = k - i < j - i + 1$

$x \notin A[i, k-1]$

Apply I.H. algorithm returns "element not found"

$\square$

Time Complexity

recursion tree method

$n_1 = j_1 - i_1 + 1$

n_1 c

$n_2 \le \lfloor \frac{n_1}{2} \rfloor$ c

$n_3 \le \lfloor \frac{n_2}{2} \rfloor$ c

$\vdots$

$n_i > 1$ c

$n_i \le 1$ d

i is $O(\log_2 n)$

$n = j - i + 1$

$\le c \log_2 n + d$

$O(\log_2 n)$

$\square$

Applications

Binary search is used in many different situations in computer science.

Example

Input n

Compute $\lfloor \sqrt{n} \rfloor$ (integer square root)

Ex: Show that $\lfloor \sqrt{n} \rfloor$ is x s.t.

$$x^2 \leq n \ , \ (x+1)^2 > n$$

(largest x s.t. $x^2 \leq n$)

binsqrt(i, j, n) // $i \le \lfloor \sqrt{n} \rfloor \le j$

if $i == j$ then return(i)

// $i < j$

$k = \left\lfloor \frac{i+j}{2} \right\rfloor$

if $k^2 == n$ then return(k)

if $k^2 \le n$ and $(k+1)^2 > n$ then return(k)

if $k^2 \le n$ and $(k+1)^2 \le n$ then binsqrt$(k+1, j, n)$

if $k^2 > n$ then binsqrt$(i, k-1, n)$

sqrt(n)
$=$ binsqrt$(1, n, n)$

__Correctness__: prove by induction on $j - i + 1$

__Base case__ $i == j$ . . . .

Induction Step $\quad i < j \qquad i \le k < j$

if $\quad k^2 \le n$ and $(k+1)^2 \le n$

$\Rightarrow \lfloor \sqrt{n} \rfloor \ge k+1 \qquad i \le \lfloor \sqrt{n} \rfloor \le j$

$\Rightarrow k+1 \le \lfloor \sqrt{n} \rfloor \le j$

binsqrt$(k+1, j, n)$ return the correct answer by induction hypothesis.

$\square$

Time complexity analysis is exactly the same as in the case of binary search assuming unit cost for operations like squaring etc.

$O(\log_2 n) \qquad O(|n|)$ If n is $\approx 10^6$ no. of step ≈ 20 $\square$

Ex Let $f : N \to N$ be a strictly monotone function.

Generalize the previous example, to compute

Input n

output largest x s.t. $f(x) \le n$

Ex Give an efficient algorithm for the following problem

Input i, n

output yes if n is ith power of some no.

no otherwise