

Searching

We have a list (array) L of n items, $L(1), L(2), L(3), \dots, L(n)$. For example, if $L = [5, 27, 13, 71]$ then $n = 4$ and $L(1) = 5, L(2) = 27, L(3) = 13, L(4) = 71$. We want to search this list for some item X . Of course X may not be in the list at all, it may be in the list once, or it may be in the list multiple times.

Linear (Sequential) Search Just search from left to right in the list:

- 1 Compare X with $L(1)$
- 2 Compare X with $L(2)$
- 3 Compare X with $L(3)$
- 4 ...
- 5 Compare X with $L(n)$

If we know X occurs in the list once, then on average we will have to look through $n/2$ items to find it (**worst case:** n comparisons, **best case:** 1 comparison). So this search is $\mathbf{O}(n)$.

Binary Search **This search algorithm only applies to ordered (sorted) data**

- 1 Compare X to the middle value in the list (i.e. $L(n/2)$).
- 2 If $X = L(n/2)$, we are finished, terminate the algorithm with “X found”.
- 3 If $X > L(n/2)$ do a binary search to the right of the mid-point (i.e. a binary search of the elements $L(n/2 + 1), L(n/2 + 2), \dots L(n)$), from step 1.
- 4 If $X < L(n/2)$ do a binary search to the left of the mid-point (i.e. a binary search of the elements $L(1), L(2), \dots L(n/2 - 1)$), from step 1.

- 5 If in either step 3. or step 4. there is no “right” or “left” of the current point, terminate the algorithm with “X not found”.

Binary Search- Example 1

Search $L = [23, 27, 45, 47, 51, 86, 97]$ for the number $X = 51$

1

- $X = L(n/2)$?
- i.e. $51 = L(4)$?
- i.e. $51 = 47$? No!

2 $51 > 47$? Yes!

3 So do binary search to the right, of the list $[51, 86, 97]$

4 $51 = 86$? No!

5 $51 > 86$? No!

6 So do binary search to the left, i.e. binary search of the list $[51]$

7 $51 = 51$ Yes!

8 Terminate with "X found".

Binary Search- Example 2

Search $L = [23, 27, 45, 47, 51, 86, 97]$ for the number $X = 50$

1

- i.e. $50 = L(4)$?
- i.e. $50 = 47$? No!

2 $50 > 47$? Yes!

3 So do binary search to the right, of the list $[51, 86, 97]$

4 $50 = 86$? No!

5 $50 > 86$? No!

6 So do binary search to the left, i.e. binary search of the list $[51]$

7 $50 = 51$ No!

8 $50 > 51$ No!

9 So do binary search to the left: But there is no left.

10 Terminate with "X not found".

Efficiency

Let $N(n)$ denote the number of comparisons we need to make in a binary search of a list of length n . After the first “iteration”, we have made one comparison, and we have a (new) problem to do a binary search of a list of size (approx.) $n/2$, so we can write

$$N(n) \approx 1 + N(n/2)$$

We can re-substitute repeatedly in this equation (representing further iterations) to get

$$\begin{aligned}N(n) &= 1 + N(n/2) \\ &= 1 + [1 + N(\frac{n/2}{2})] \\ &= 2 + N(n/4) \\ &= 2 + [1 + N(\frac{n/4}{2})] \\ &= 3 + N(n/8) \\ &= 3 + [1 + N(\frac{n/8}{2})] \\ &= 4 + N(n/16) \\ &= \dots \\ &= k + N(n/2^k)\end{aligned}$$

Clearly, $N(1) = 1$, so we terminate when

$$\frac{n}{2^k} = 1$$

or $k \approx \log_2 n$, which gives $N(n) \approx \log_2 n$. So the Binary Search Algorithm is $O(\log_2 n)$.

Pseudocode

Linear Search

```
BEGIN
  IN(L, n, X)
  {Linear Search of a List L of length n
   for item X}
  FOR i ← 1 TO n DO
    IF L(i)=X THEN OUT(i)
  END
```

OR

```
BEGIN  
  IN(L, n, X)  
  LS(L, 1, n, X)  
END
```

```
LS(M, i, j, Y)  
{Linear Search of a List M between  
index i and j for item Y}  
IF i>j THEN OUT("Y not found")  
  ELSE IF Y=M(i) THEN OUT(i)  
    ELSE LS(M, i+1, j, Y)
```

Binary Search

```
BEGIN
  IN(L, n, X)
  BS(L, 1, n, X)
END

BS(M, i, j, Y)
  {Binary Search of a List M between
  index i and j for item Y}
  IF i>j THEN OUT("Y not found")
  MP ← FLOOR((i+j)/2)
  IF Y=M(MP) THEN OUT("Y found at MP")
    ELSE IF Y>M(MP) THEN BS(M, MP+1, j, Y)
      ELSE BS(M, i, MP-1, Y)
```

Aside

FLOOR truncates numbers to their integer value: e.g.

- $\text{FLOOR}(3) = 3$
- $\text{FLOOR}(3.4) = 3$
- $\text{FLOOR}(3.8) = 3$

You can regard this as “built in” to the pseudocode language, or if necessary it can be easily calculated:

```
FLOOR(x)
```

```
i ← 0
```

```
WHILE  $i \leq x$  DO i ← i+1
```

```
FLOOR ← i-1
```

