KEY POINTS

Recursion: when a method calls itself

Binary search: O(n) for unsorted sequence.
: O(log n) otherwise

NAME/DATE/SUBJECT

CS126 - Recursion

NOTES

Examples of recursion:

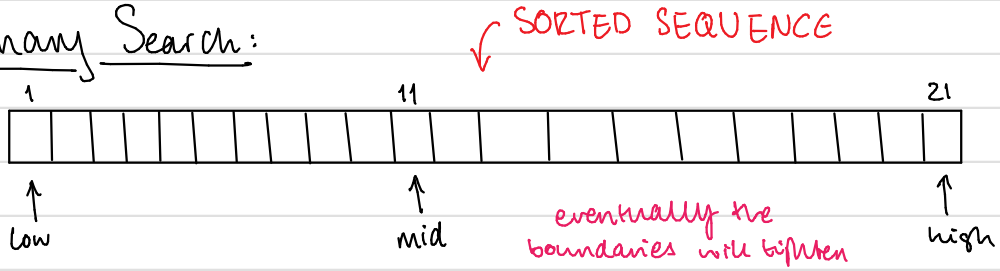
- factorial function

$$fc \quad \begin{cases} 1 & \text{if } n = 0 \\ n \cdot f(n-1) & \text{else.} \end{cases}$$

- binary search
  locate a target value within a sorted sequence.

Each recursive function has a base case(s), and then recursive calls to the function, which will lead to the base case.

Binary Search:



We check three things: (base case)

- target = data[mid] ? target located
- target < data[mid] ? recur on first half
- target > data[mid] ? recur on second half

SUMMARY

KEY POINTS

NAME/DATE/SUBJECT

Recursion - Analysing base case.

NOTES

(for DCS formula)

What is the worst case for input size n?

- Each call has a constant number of primitive operations.
- Time is proportional to the number of recursive calls.
- The algorithm runs in $O(\log n)$ time:

SUMMARY

KEY POINTS

- Linear rec -
  recur once
- Binary rec -
  recur twice for each non base case.
- Multiple rec
  multiple recursion possible.

NAME/DATE/SUBJECT

Calls within activation

NOTES

Linear Recursion uses a single recursive call, but there may be a test that decides which of several calls to make.

- linear sum : if $n = 0$? return 0 : return lsum $(A, n-1) + A[n-1]$
- reversing an array
- computing powers of a number :

$$\text{power}(x, n) \begin{cases} 1 & n = 0 \\ x\,(\text{power}(x, \frac{n-1}{2}))^2 & n > 0 \text{ odd} \\ (\text{power}(x, \frac{n}{2}))^2 & n > 0 \text{ even} \end{cases}$$

$O(\log n)$ - it halves each time

Binary Recursion

Similar to linear recursion - however, we make two recursive calls.

- fibonacci (return fib(n-1) + fib(n-2)

We can make this algorithm better (it is currently exponential) by using linear recursion - we return a pair.

Multiple Recursion

Makes potentially many recursive calls.

SUMMARY