Data Search

O() notation

- Used in study of algorithmic complexity: beyond scope of this course
- Rough approximation: “order of magnitude”, “approximately” etc.
- Main concepts here:
  - O(1) - constant time independent of input size - excellent!
  - O(log N) - logarithmic in input size - grows slowly with input - very good
  - O(N) - linear in input size - often the baseline - would like to do better
  - $O(N^k)$ - polynomial (quadratic, cubic etc.) - not good as input size grows
  - $O(k^N)$ - exponential - VERY bad: won’t work even for reasonably small inputs

Searching for element in memory

Unsorted data in a linked list

- Start from beginning
- Proceed stepwise, comparing each element
- Stop if found and return LOCATION
- If end-of-list, return NOTFOUND

O(N)

Searching for element in memory

Unsorted data in array

- Start from beginning
- Proceed stepwise, comparing each element
- Stop if found and return LOCATION
- If end-of-list, return NOTFOUND

O(N)

Searching for element in memory

Sorted data in array

- Start from beginning
- Proceed stepwise, comparing each element
- Stop if found and return LOCATION
- If end-of-list, return NOTFOUND

O(N) but...

Searching for element in memory

Sorted data in array

- Look at middle element in array:
  - greater than target - search in lower half
  - lesser than target - search in upper half
- Switch focus to new array: half the size of original
  - Repeat

O(log N)

Problems with arrays

- Size must be fixed ahead of time
- Adding new entries requires resizing - can try oversize, but eventually ...
- Maintaining sorted order O(N):
  - find location to insert
  - move all further elements by 1 to create a gap
  - insert
- Deleting
  - find location, delete
  - move all entries down by 1 step

Alternatives

- Binary search tree
  - Maintaining sorted order is easier: growth of tree

Alternatives

- Binary search tree
  - Maintaining sorted order is easier: growth of tree
- Self-Balancing
  - BST can easily tilt to one side and grow downwards
  - Red-black, AVL, B-tree… more complex, but still reasonable

Alternatives

- Binary search tree
  - Maintaining sorted order is easier: growth of tree
- Self-Balancing
  - BST can easily tilt to one side and grow downwards
  - Red-black, AVL, B-tree… more complex, but still reasonable
- Hash tables
  - Compute an index for an element: O(1)
  - Hope the index for each element is unique!
    - Difficult but doable in many cases