COMPSCI 105 Lecture 19:
Simple Searching and Sorting

Also relevant to CMPSCI 145

©2015-2019 Dr. William T. Verts

Why do we Care?

• It is estimated that around 70% of what any computer does over its entire operational lifetime is either searching or sorting.
• Searching: Looking for an item in a group of items. Can be a simple list or something much more complicated.
• Sorting: Arranging items into some order (numerical or alphabetical), either for our convenience or to make searches faster.

Big-O Notation

• There are N items in the list to search or sort.
• Mathematically, we describe the “goodness” or worst-case performance of a technique as an expression involving N, called “big-O” notation.
  – O(1) = constant performance, regardless of N.
  – O(N) = performance directly proportional to N.
  – O(log₂(N)) = performance proportional to the logarithm base 2 of N.
  – O(N²) = performance proportional to the square of N.
  – Etc., etc., etc.

Big-O Example

• For example, suppose we can describe the performance of a technique precisely as:
  \[ f(N) = 3N^2 - 4N + 2 \]
  
  • As N grows, the dominant term is the 3N² (grows much faster than the 4N term).
  • The coefficient 3 is largely irrelevant (could be 5 or 2 and doesn’t change the N² effect).
  • This is considered an O(N²) technique.

Linear Search

• Items can be in any order,
• Have to examine first record, then second record, then third record, etc., until item is found or all items have been examined,
• Worst case search time (item not found) is O(N) for N items,
• Search time grows linearly as a function of N.

Self-Organizing Lists

• What if we move every item searched closer to the front of the list, so it can be found faster next time?
• Several self-organizing list techniques:
  – Swap with front (fast to move, no clustering)
  – Move to front (slow to move on simple lists, fast on more complicated structures, excellent clustering)
  – Promote by one slot (fast to move, slow clustering)
  – Promote by half the list length (OK but not great move and clustering performance)
• Can improve average performance, but not worst case. Still O(N) worst case!
### Binary Search

- Items must be sorted on search field,
- Examine middle record, stop if found, but if not found discard half of list known to not contain item, repeat until found or list empty,
- Worst case search time (item not found) is \( O(\log_2(N)) \) for \( N \) items,
- Search time grows logarithmically as a function of \( N \),
- \( O(\log_2(N)) \) grows more slowly than \( O(N) \), so binary search is much faster than linear search for large \( N \).