CS2100

Searching and Sorting
Recap:

- HW due today - via git
- Lab due today
- Extra credit due Monday
- Posted 1st HW (due Sunday March 18?)

I am traveling
(get me questions next week)
Searching:

Given a value $x$ of data structure $S$, output true if $x$ is in $S$.

Often also want an iterator to the value, or an index (if array-based).

Two ways:

- binary search (fast, but needs sorting)
- linear search (look at everything)
Coding & runtimes:
Linear Search.

- You've actually done the code for this (or nearly have) in both SlinkedList & Vector!

A simple loop to run through the data:

- return true if ever found (or iterator/loc)
- return false if not found

Run time:
- Lists $\mathcal{O}(n)$
- Vectors $\mathcal{O}(n)$
Binary Search:

- Check middle entry, val.

\[
\begin{cases}
\text{If } x < \text{val, } & \text{search right half} \\
\text{Else} & \text{search left half}
\end{cases}
\]

(if equal, report "true")

Note: Need to be careful!

Issues: Recursive can be problematic - don't want O(n) time copy

Need: left & right index

Runtime: \[B(n) = 5 + B(\frac{n}{2})\]

Vectors: \(O(\log n)\)

Lists: \(O(n) \leq \text{BAD (bad of operator \[\]})\)
Next: Sorting!

Algorithms?

- Bubble Sort
- Merge Sort
- Insertion Sort
- Quick Sort
- Radix Sort
- Heap sort
An easy one: Bubble Sort

Idea: \[ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9 \]

for \( i = 1 \) to \( n \)
for \( j = \frac{1}{2} \) to \( n - (i - 1) \)
    swap them

Details:
- linked structure - deal
- \( n \) pointers & move iterators around
- versions of this go from \( n \) to \( i \) & swap down

Runtime: \( O(n^2) \)
\[
\sum_{i=1}^{n} \sum_{j=1}^{\frac{n}{2}} 4 = \sum_{i=1}^{n} 4^i - 4 \sum_{i=1}^{n} (n - 1) = O(n^2)
\]
Quick Sort:

Idea: Choose “pivot” + divide array

$5 \ 6 \ 2 \ 11 \ 13 \ 3$

Then repeat on each side if $> 1$.

Recursively:

Quicksort both sides

Issues:

- Use reference to pass entire list
- Need left & right index of current sublist
“Code”:

Quick Sort \((\text{list } A, \text{ length } n)\)

for \((i = 2 \text{ to } n)\)

if \(A[i] > A[i]\)

move \(A[i]\) to front

else

move \(A[i]\) to back

Quick Sort (first "half")

Quick Sort (second "half")

Details:

linked vs vector

easy for this alg

need to track indices

Runtime:

Worst case \(O(n^2)\)

Expected: \(O(n \log n)\)
Merge Sort:
if length of A is > 2
    divide in half
    Merge sort (left)
    Merge sort (right)
    Merge (left and right)
else // (list of length 0 or 1)
    Merge (left, right)
    create empty list L
    i ← 1, j ← 1
    while (i or j
Ex: 5 11 3 2 6 8 7 4

\[
\begin{array}{cccc}
5 & 11 & 3 & 2 \\
\hline
2 & 5 & 3 & 1 \\
\hline
2 & 3 & 5 & 1 \\
\hline
\end{array}
\]
Runtimes:

Quicksort:

Mergesort:
Others: