CS2100

Searching & 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 & 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/location)
  - return false if not found

  → tricky bit is avoid seg fault

Run time:

- Lists
- Vectors

] $O(n)$

Binary Search:

- Check middle entry, val.

If $x < val$, search right half

Else search left half

(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:

$$B(n) = 5 + B\left(\frac{n}{2}\right)$$

Vectors: $O(\log n)$

Lists: $O(n)$ ← BAD

(b/c 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~~ 6 ~~6~~ 5 ~~5~~ 4 ~~8~~ 9 9

```
for i = 1 to n
    for j = 1 to n-(i-1)
        if A[j-1] > A[j]
            swap them
```

Details:

- linked structure — decl w/ pointers & move iterators around
- versions of this go from n to i & swap "down"

Runtime: $O(n^2)$

$$\sum_{i=1}^{n} \sum_{j=1}^{i} 4 = \sum_{i=1}^{n} 4i$$

$$= 4 \sum_{i=1}^{n} i = 4 \frac{n(n-1)}{2} = O(n^2)$$

Quick Sort:

Idea: Choose "pivot" + divide array

pivot

5 6 2 11 13 3

11 6 13

2 3 5 13 11 6

0 1 3 5

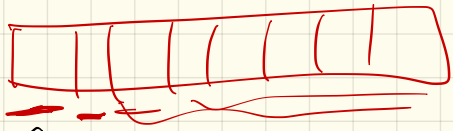
Then repeat on each side. if > 1.

recursely quicksort both sides

Issues:

Use reference + pass entire list

- Need left + right index of current sub list



"Code":

```
QuickSort (list A, length n)
  for (i = 2 to n)
      if A[1] > A[i]
          move A[i] to front
      else
          move A[i] to back
  QuickSort (first "half")
  QuickSort (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

Mergesort (left)

Mergesort (right)

Merge (left + right)

return list

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

5 11 3 2

5 11 3 2

2 3 5 11 4 6 7 8

2 3 4 5 6 7 8

Runtimes:

Quicksort:

Mergesort:

Others: