CS180 - Sorting

Note Title 10/31/2011

Announcements

- Test Friday
  review tomorrow
- Next HW is posted - over lists
  due after fall break

Vectors versus lists

Q: What would operator [] look like in a list?

mylist[2] = "a";

How?

```
T& operator[](int index) {

}
```

Vectors versus lists (cont)

Running times:

	Vectors	Lists
operator[]	O(1)	O(n)
find	O(n)	O(n)
insert	O(n)	O(1)
erase/remove	O(n)	O(1)

Searching

What is linear search?

go through data element by element, check if data is present

Binary Search?

1 comparison in ~~sorted~~ list eliminates 1/2 ~~of list~~

$$B(n) = 1 + B\left(\frac{n}{2}\right)$$

$$B(1) = 1$$

$$\Rightarrow B(n) = O(\log_2 n)$$

Practical Considerations

Which is better?

binary search is faster.

↳ need to compare with $A\left[\frac{size}{2}\right]$

operator[] - fast in vector
sucks in a list!

Sorting

Name some sorting algorithms.

- Bubble sort
- Insertion sort
- Quicksort
- Merge sort

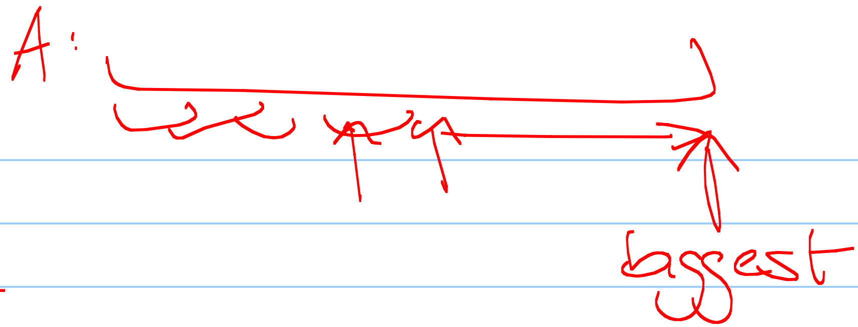
Insertion Sort

i – already sorted $i+1$

for $i = 1$ to $n-1$
    find where $i+1$ goes in first i sorted elements } $O(n)$

$\Rightarrow O(n^2)$

Bubble Sort



for $i = n$ down to 1

  for $j = 1$ to i

    compare $A[j]$ to $A[j+1]$

    swap if out of order

$$\sum_{i=1}^{n}\sum_{j=1}^{i} 5 = \sum_{i=1}^{n}\left(\underbrace{5+5+5+\cdots+5}_{i}\right)$$

$$= \sum_{i=1}^{n} 5i = 5\sum_{i=1}^{n} i = 5(1+2+3+\cdots+n) = \frac{5n(n+1)}{2}$$

$$= O(n^2)$$

Merge Sort - recursion

7 5 16 11 | ...

$n/2$

7 5 | 16 11

5, 7   11 16

5 7 11 16, 3 4 9 10

left   right

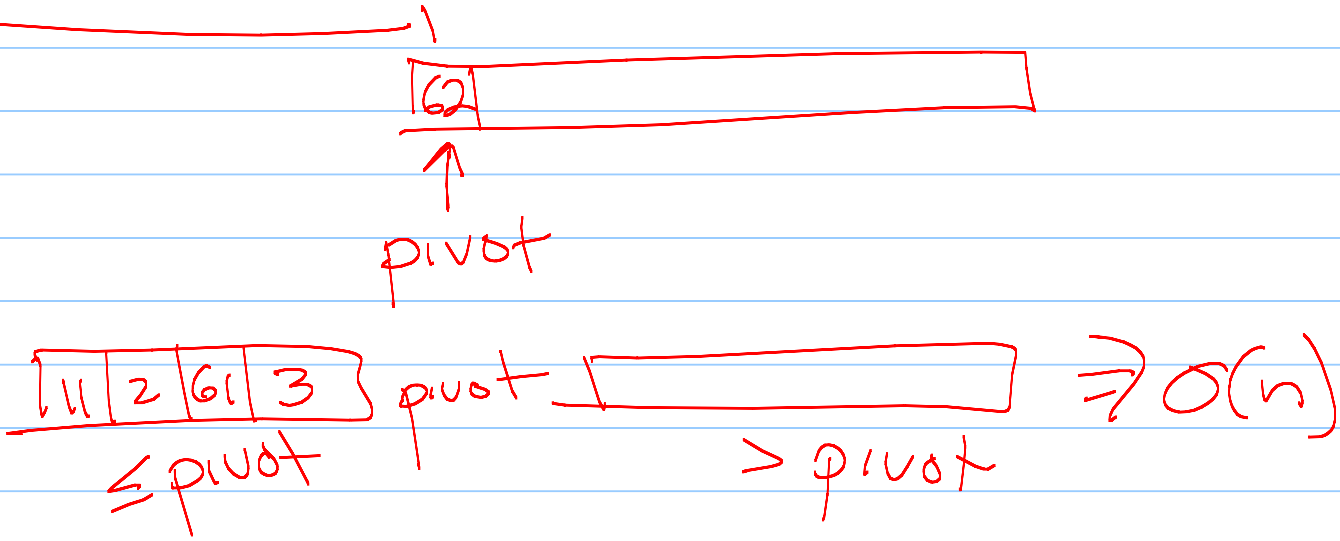
5, 7, 11, 16 → 3, 4, 5, 7, 9, 10,

$$M(n) = M\left(\frac{n}{2}\right) + M\left(\frac{n}{2}\right) + O(n)$$

$$= 2M\left(\frac{n}{2}\right) + O(n)$$

$$= O(n \log n)$$

Quick Sort



worst case – $O(n^2)$

expected time : $O(n \log n)$

Bucket Sort

n elements, each between 0 and N−1

Can we do better than $O(n \log n)$?

Radix Sort : for multiple-key sorting

Ex: (1, 5), (2, 1), (4, 2), (3, 3), (5, 4), (3, 1), (2, 2), (5, 1), (2, 4)

Sort lexicographically: (use repeated bucket sorts)

Practicalities

Expirimentally, quicksort runs faster than merge on small inputs.

Why?

- can do it "in place" (easier to code)

More practicalities

- If implemented well, the running time of insertion sort is $O(m+n)$, where m = # of <u>inversions</u> (or out of order elements)

<u>Conclusion:</u> depends

- If the range of values is small, bucket sort (or radix sort) are faster.