Announcements

- No class tomorrow (lab still due Sunday)
- Check point tomorrow
- HW due Sunday
Operator < (in vectors)

\[ [a_1, a_2, \ldots, a_n] \quad [b_1, \ldots, b_m] \]

compare \( a_i \) \& \( b_i \),

\[ A[i] > B[i] \]
Vectors versus lists

Q: What would operator \( \sum \) look like in a list?

- Start at \( - \text{sent} \)
- Loop up to input \( \leq 0(n) \)
- Move to next
<table>
<thead>
<tr>
<th>Operation</th>
<th>Vectors</th>
<th>Lists</th>
</tr>
</thead>
<tbody>
<tr>
<td>operator [J]</td>
<td>$O(1)$</td>
<td>$O(n)$</td>
</tr>
</tbody>
</table>
| find        | $O(n)$  | $O(n)$| ←
| insert      | $O(n)$  | $O(1)$|
| erase/remove| $O(n)$  | $O(1)$|
Searching

What is linear search?

Hunt element by element.

Binary search?

Fundamental divide and conquer:

$B(n) = \text{running time of input of size } n$

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

$= O(\log_2 n)$

in list: $B(n) = \frac{n}{2} + B\left(\frac{n}{2}\right)$
Sorting

Name some sorting algorithms.

- Bubble sort
- Merge sort
- Insertion sort
- Quick sort
- Marriage sort
- Bucket sort
Bubble sort

Move down list, comparing neighbors.
If in wrong order, swap them.
At end of list, largest element must be at end.

(Nested for loops)

Easy to code.

$O(n^2)$
Merge sort

Base case: size 1 or 2:
  Merge sort first half
  Merge sort second half

Claim: At most n comparisons until sorted.
Merge sort (list of size 8):

5 3 8 1 2 6 7 4

merge sort (3)

5 3 8

merge sort (4)

1 3 5 8

4 6 7
Quicksort

10 2 3 11 6 9 -1 7 12
2 3 6 9 -1 7 10 11 12

Call Quicksort on this half Quicksort

In worst case, get bad pivot.
**Merge sort** : \( M(n) = M\left(\frac{n}{2}\right) + M\left(\frac{n}{2}\right) + O(n) \)

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

\[= O(n \log n)\]

**Quicksort** : \( Q(n) \leq \frac{n^2}{2} + Q(n-1) \)

\[= \sum_{i=1}^{n-1} i = O(n^2)\]

\( n-1 \)
Insertion:

$k-1$ are in sorted order

$k$th

for $k = 2$ to $n$

$log_2 + log_3 + ... + log n$

$log (2 \cdot 3 \cdot 4 \cdots n)$

$log (n!)$

$O(n \log n)$

$O(n^2)$