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 to operator [] (int index)?
### Vectors versus lists (cont)

<table>
<thead>
<tr>
<th>Running times:</th>
<th>Vectors</th>
<th>Lists</th>
</tr>
</thead>
<tbody>
<tr>
<td>operator [ ]</td>
<td>$O(1)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>find</td>
<td></td>
<td>$O(n)$</td>
</tr>
<tr>
<td>insert</td>
<td>$O(n)$</td>
<td></td>
</tr>
<tr>
<td>erase/remove</td>
<td>$O(n)$</td>
<td>$O(1)$</td>
</tr>
</tbody>
</table>
Searching

What is linear search?

1. Go through data element by element.
2. Check if data is present.

Binary Search?

1. Comparison in sorted list eliminates 1/2 of list.

\[
\begin{align*}
B(n) &= 1 + B\left(\frac{n}{2}\right) \\
B(1) &= 1
\end{align*}
\]

\[\Rightarrow B(n) = O(\log_2 n)\]
Practical Considerations

Which is better?

- Binary search is faster.
- You need to compare with $A_{\frac{\text{size}}{2}}$

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

Name some sorting algorithms.

- Bubble sort
- Insertion sort
- Quicksort
- Merge sort
Insertion Sort

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

$\Rightarrow O(n^2)$
Bubble Sort

```plaintext
for i = n down to 1
    for j = 1 to i

\[ \sum_{i=1}^{n} \sum_{j=1}^{i-1} \frac{5}{i+1} = \frac{n}{i=1} \left( \frac{5 + 5 + 5 + \ldots + 5}{i} \right) = \frac{n}{i=1} \left( \frac{5 \sum_{i=1}^{n} i}{i} \right) = \frac{5n(n+1)}{2} = \Theta(n^2) \]
```
Merge Sort - recursion

\[ \frac{17}{5} \frac{10}{11} \]

\[ \frac{25}{11} \frac{16}{16} \]

5, 7, 11, 16 → 3, 4, 5, 7, 9, 10

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

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

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

\[ 62 \]

\[ \text{pivot} \]

\[ \begin{array}{cccc}
1 & 1 & 2 & 6 & 13 \\
\end{array} \]

\[ \begin{array}{cccc}
\leq \text{pivot} & > \text{pivot} \\
\end{array} \]

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

Experimentally, 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 $m = \# \text{ of inversions}$ (or out of order elements)

Conclusion: depends

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