Linear Search

```python
def linear_search(v, b):
    # Precond: b a list of number, v a number
    # Loop variable
    i = 0
    while i < len(b) and b[i] != v:
        i = i + 1
    if i == len(b):
        # not found
        return -1
    return i
```

How many entries do we have to look at?
All of them!

Binary Search

```python
def binary_search(v, b):
    # Loop variable(s)
    i = 0, j = len(b)
    while i < j and b[i] != v:
        mid = (i+j)//2
        if b[mid] < v:
            j = mid
        elif b[mid] > v:
            i = mid
        else:
            return mid
    return -1
```

Requires that the data is sorted!
But few checks!

The Sorting Challenge

- **Given**: A list of numbers
- **Goal**: Sort those numbers using only
  - Iteration (while-loops or for-loops)
  - Comparisons (< or >)
  - Assignment statements
- **Why?** For proper analysis.
  - Methods/functions come with hidden costs
  - Everything above has no hidden costs
  - Each comparison or assignment is “1 step”

Horizontal Notation

- Want a pictoral way to visualize this sorting
  - Represent the list as long rectangle
  - We saw this idea in divide-and-conquer
  - Do not show individual boxes
    - Just dividing lines between regions
    - Label dividing lines with indices
    - But index is either left or right of dividing line

Visualizing Sorting

Start: b

Goal: b

In-Progress: b

Insertion Sort

```python
i = 0
while i < n:
    # Push b[i] down into its
    # sorted position in b[0..i]
    i = i+1
```

Remember the restrictions!
Insertion Sort: Moving into Position

```python
def push_down(b, i):
    j = i
    while j > 0:
        if b[j-1] > b[j]:
            swap(b, j-1, j)
        j = j - 1
```

<table>
<thead>
<tr>
<th>Operation</th>
<th>Complexity</th>
<th>n=10</th>
<th>n=100</th>
<th>n=1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>log n</td>
<td>0.003 s</td>
<td>0.006 s</td>
<td>0.01 s</td>
<td></td>
</tr>
<tr>
<td>n</td>
<td>0.01 s</td>
<td>0.1 s</td>
<td>1 s</td>
<td></td>
</tr>
<tr>
<td>n log n</td>
<td>0.016 s</td>
<td>0.32 s</td>
<td>4.79 s</td>
<td></td>
</tr>
<tr>
<td>n²</td>
<td>0.1 s</td>
<td>10 s</td>
<td>16.7 m</td>
<td></td>
</tr>
<tr>
<td>n³</td>
<td>1 s</td>
<td>16.7 m</td>
<td>11.6 d</td>
<td></td>
</tr>
<tr>
<td>2ⁿ</td>
<td>1 s</td>
<td>4x10⁹ y</td>
<td>3x10²⁹ y</td>
<td></td>
</tr>
</tbody>
</table>

Selection Sort

```python
b = [0, sorted, ... ≤ b[i-1] ≤ b[0..i-1]]

i = 0
while i < n:
    # Find minimum in b[i..]
    # Move it to position i
    i = i + 1
```