"Organizing is what you do before you do something, so that when you do it, it is not all mixed up."

~ A. A. Milne
Why Sorting?

- Sorting is useful
  - Database indexing
  - Scheduling
  - Operations research
  - Compression
Some Sorting Algorithms

There are lots of ways to sort

- Insertion sort
- Selection sort
- Merge sort
- Quick sort
- And more …

There isn't one right answer

You need to be able to figure out the options and decide which one is right for your application.
Selection Sort

for each i:

    int m = index of min of b[i..];

    Swap b[i] and b[m];
# Recursive Selection Sort

```java
selectionSort(int[] b, int i){
    if(i > 0){
        // Recursively sort b[0..i-1]
        selectionSort(b, i-1);
    }

    // find index of min b[i..]
    int m = indexOfMin(b, i);

    swap b[i] and b[m]
}
```

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>30</td>
<td>10</td>
<td>26</td>
<td>34</td>
<td>18</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>30</td>
<td>10</td>
<td>26</td>
<td>34</td>
<td>18</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>30</td>
<td>26</td>
<td>34</td>
<td>18</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>14</td>
<td>26</td>
<td>34</td>
<td>18</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>14</td>
<td>18</td>
<td>34</td>
<td>26</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>14</td>
<td>18</td>
<td>26</td>
<td>34</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>14</td>
<td>18</td>
<td>26</td>
<td>30</td>
<td>34</td>
<td></td>
</tr>
</tbody>
</table>

- \( b \) sorted, smaller values
- larger values
Induction

• Mathematical proof technique

• To prove propositions:
  \( \forall n \geq c, P(n) \) is true

1) Prove Base Case:
   \( i = c \)

2) Prove Induction Case
   \( \forall i > c, P(i - 1) \Rightarrow P(i) \)
Induction Example

• Prove $\forall n \geq 0, \ n < 2^n$

• Prove $\forall n \geq 1, 1 + 2 + \cdots + n = \frac{n(n+1)}{2}$
Correctness of Recursive Selection Sort

```java
selectionSort(int[] b, int i){
    if(i > 0){
        //Recursively sort b[0..i-1]
        selectionSort(b, i-1);
    }

    // find index of min b[i..]
    int m= indexOfMin(b,i);

    // swap b[i] and b[m]
    int temp= b[i];
    b[i]= b[m];
    b[m]= temp;
}
```
Complexity of Recursive SelectionSort

• To compute the running time of this algorithm, we need to count the number of comparisons in each recursive call
• All of the comparison are in $\text{indexOfMin}(b, i)$
  • that makes __________ comparisons
• Selection sort takes time __________________________
## Performance

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Time</th>
<th>Space</th>
<th>Stable?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selection Sort</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Insertion Sort</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Merge Sort</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Heap Sort</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Quick Sort</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
More fun with induction...
RSA

Encrypt: $c = m^e \mod n$

Decrypt: $m = c^d \mod n$
Fast Power

```c
fastPower(int x, int n){
    if(n == 0){ return 1; }

    // handle odd values
    if(n % 2 == 1){
        return x * fastPower(x, n-1);
    }

    // handle even values
    return fastPower(x*x, n/2);
}
```
Strong Induction

Mathematical proof technique
To prove propositions:
\[ \forall n \geq c, \ P(n) \text{ is true} \]
1) Prove Base Case:
\[ i = c \]
2) Prove Induction Case:
\[ \forall i > c, \ (\forall c \leq j < i, \ P(j)) \Rightarrow P(i) \]
Correctness of Fast Power

- Base Case: \( i = 0 \)

- Induction Case: \( \forall i > c, \ (\forall c \leq j < i, \ P(j)) \Rightarrow P(i) \)
  
  - If \( i \) is odd:

  - If \( i \) is even:
We can iteratively compute exponentiation with square (and optional multiply) and bit shifting!
Iterative Square-and-Multiply

- Example: compute $5^{13}$ (13 is 1101 in binary)

1.
2.
3.
4.
5.
Side Channel Attacks