How good is your algorithm?

• Now we know how to write algorithms in Python.
• But how do we know how efficient it is?
• Efficiency measured by:
  — Space – how much space does it take?
  — Time – how long does it take?
How do we measure time efficiency?

• Why not just see how long an algorithm takes?
• Problem:
  – Different operating systems, processors, programming languages run at different speeds
  – Doesn’t really measure efficiency of algorithm

Amount of Work

• To measure time efficiency we count the Amount of Work done by an algorithm
• Each operation in an algorithm is approximately one unit of work
• Sequential operations

```python
count = count + 1
print("The value of count is ", count)
Name = input("Enter your name: ")
```
Amount of Work

• Conditional operations
  
  if found == true:
      print("You found it!")
  else:
      print("Sorry, not found")

• Iterative operations
  
  while i<len(nameList):
      print("The name is ", nameList[i])
  
  i = i + 1

  for i in range(len(nameList)):
      print("The name is ", nameList[i])

Find Cheapest Film

• Given a list of film costs, find the cheapest film

  (1) cheapest = costs[0]
  (1) for i in range(len(costs)):
  
  5
  
  (1) if costs[i] < cheapest:
      (1) cheapest = costs[i]

  1 + 2 * 5 = 11   1 + 3 * 5 = 16

• Amount of work depends on the data!
  • Arrangement
Best Case / Worst Case Input

- Amount of work depends on arrangement of input
- **Best case input**
  - Input that allows algorithm to do the least amount of work
  - Ex: Cheapest film – costs sorted lowest to highest
- **Worst case input**
  - Input that allows algorithm to do the most work
  - Ex: Cheapest film – costs sorted highest to lowest

Find Cheapest Film

- Given a list of film costs, find the cheapest film

\[
\begin{align*}
&\text{cheapest} = \text{costs}[0] \\
&\text{for } i \text{ in range(len(costs))}: \\
&\quad \text{if } \text{costs}[i] < \text{cheapest}: \\
&\quad \quad \text{cheapest} = \text{costs}[i]
\end{align*}
\]

\[
1 + 3 \times 100 = 301
\]
Find Cheapest Film

• Given a list of film costs, find the cheapest film

\[
\text{cheapest} = \text{costs}[0] \\
\text{for } i \text{ in range(len(costs))}: \\
\quad \text{if costs}[i] < \text{cheapest}: \\
\quad \quad \text{cheapest} = \text{costs}[i]
\]

\[
1 + 3 \times 10000 = 30001
\]

Find Cheapest Film

• Given a list of film costs, find the cheapest film

\[
\text{cheapest} = \text{costs}[0] \\
\text{for } i \text{ in range(len(costs))}: \\
\quad \text{if costs}[i] < \text{cheapest}: \\
\quad \quad \text{cheapest} = \text{costs}[i]
\]

\[
1 + 3 \times n = 3n + 1
\]

• Amount of work depends on the data!
  • Size
Another example

```python
# Find most wins for baseball team league
for i in range(numTeams):
    for j in range(numTeams):
        if wins[i][j] > mostWins:
            mostWins = wins[i][j]
    print("The most wins = ", mostWins)
```

\[ n \times (3n + 1) + 1 = 3n^2 + n + 1 \]

<table>
<thead>
<tr>
<th>Team_1</th>
<th>Team_2</th>
<th>Team_3</th>
<th>Team_n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Team_1</td>
<td>0</td>
<td>13</td>
<td>14</td>
</tr>
<tr>
<td>Team_2</td>
<td>10</td>
<td>0</td>
<td>15</td>
</tr>
<tr>
<td>Team_3</td>
<td>11</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Team_n</td>
<td>12</td>
<td>14</td>
<td>16</td>
</tr>
</tbody>
</table>

Order of Magnitude

- Amount of work can be generalized to order of magnitude

- Cheapest film
- \(3n + 1\)
- Linear – \(O(n)\)

- Most wins
- \(3n^2 + n + 1\)
- Quadratic – \(O(n^2)\)
Order of Magnitude

Linear
- Amount of work increases proportionally to the size of the data
- $O(n)$
- Rule of thumb:
  - Any algorithm that goes through the data one item at a time

Quadratic
- Amount of work increases quadratically with the size of the data
- $O(n^2)$
- Rule of thumb:
  - Any algorithm that has nested loops going through a list of $n$ items $n$ times