- Pair project probably due in 1-2 weeks
- Programming project 1 - East
- Lab 2 tomorrow
- HW due tomorrow

Announcements

CS 180 - Lecture 7
Input set matters
Language does
Compile
Disadvantage, compute matters

So how?

Define the complexity running time?

Ch3 - How to apply to running time?
Example: variable assignment / update

- Multiplication
- Subtraction
- Addition
- Printing
- Comparison

Computers, independent of language compilers, can identify high-level primitive operations.
Return Current Max

if Current Max > A[i] then

Current Max = A[i]

for i = 1 to n-1

Current Max = A[i]

Output: The maximum element of A

Input: An array A of n > 1 numbers

Algorithm Array Max(A[n]):

Write easily readable, independent code

Ex: (Pseudocode) to find max in an array
Return current max

```cpp
int currentMax = A[0];
```

if (currentMax > A[i])

```cpp
for (int i = 1; i < n / 2; i++)
```  
```cpp
int currentMax = A[i];
```  
```cpp
3
```  
```cpp
\text{Ex: (in C++)}
```

```
\text{Easily to read \& understand to any language}
```

```
\text{Independent of language}
```

```
\text{Advantage of pseudocode:}
```

```
\text{```cpp}
```
Algorithm Array Max\(\left( A, n \right)\):

1. For \(i \rightarrow 1\) to \(n-1\) comparisons:
   - Current Max \(\rightarrow A[0] \rightarrow A[0]

2. If \(\text{Current Max} < A[i]\):
   - \(\text{Current Max} \rightarrow A[i]\)

3. For \(i \rightarrow 1\) to \(n-1\) comparisons:

4. Return Current Max

Output: The maximum element of \(A\)

Input: An array of \(A\) of \(n \geq 1\) numbers

Counting Operations:
So how many operators in best (or worst)

(see previous)

Any to numerator

(see)
Worst case: $n + 2$

Average: $3n + 3$

Input

Another fine for worst possible average

Look at all possible inputs

Average case versus worst case
input size (usually n), running time of assorted algorithms. So we'll focus on big-picture or how the

independent of language or computer.

In general, any primitive statement depends on a small number of low-level operations. How important is the exact number of computations?

Asymptotic Notation
(c) is big-oh of g(n).

Such that f(n) ≤ C * g(n) for all n ≥ n₀. A constant C ≥ 0 and integer n₀ ≥ 0.

We say f(n) is O(g(n)) if there exist non-negative integers t and real numbers t₀ and t₁ such that f(n) ≤ t * g(n) for all n ≥ n₀.

Let f(n) and g(n) be two functions.

For example: Big-Oh notation.
If $n > 100$, then $4n + 2 < 5n$.

Let $n = 100$ and $n \neq 100$.

Let $n = d$, which $4n + 2 < 5n$.

Why?

Find $c + n$.

Ex: $4n + c = 15 (\text{cn}).$
Why? Just showed worst case running

\[
\text{Algorithm} \ \text{ArrayMax(A, n)}:
\]

\[
\begin{align*}
\text{Input}: & \text{ An array } A \text{ of } n > 1 \text{ numbers} \\
\text{Output}: & \text{ The maximum element of } A \\
\text{Running time of arrayMax is } O(n) \\
\end{align*}
\]

\[
\begin{align*}
\text{for } i \rightarrow 1 \text{ to } n-1 \\
\text{CurrentMax} \rightarrow A[i] \\
\text{if CurrentMax } \overset{?}{<} \text{ Max(A)} \rightarrow \text{Max(A)} \\
\text{return CurrentMax}
\end{align*}
\]
\[30n^3 + 10n \log n + 5 > 35n^3\]

Why? \(n = 35\) or \(n^6 = 2\)

Ex: \(30n^3 + 10n \log n + 5\)
Any polynomial: $a_n n^k + a_{k-1} n^{k-1} + \cdots + a_0$
$2,000 = 1 - 2,000$

Let $c = a, z_{100} + \frac{a}{n} = 1$

\[ \text{Ex: } c_{100} = 0(1) \]
\[ \log_a n = \frac{1}{\log_b n} \] for any constant \( c > 0 \).

- \( \log a \) is \( O(\log n) \).

From \( d(n) \) is \( O(n) \) and \( f(n) \) is \( O(g(n)) \),

\[ \text{Examples:} \]

1. \( \log \) in base 2: Rules
\[ c = \frac{1 - a}{2} \]

And if \( a < 1 \), then

\[ \sum_{n=0}^{\infty} a^n = \frac{1}{1 - a} \]

For any \( n \geq 1 \) and \( a < 1 \):

Loops often produce these:

\[ \sum_{n=0}^{\infty} f(c) = f(a) + f(a+1) + \ldots + f(b) \]

Useful things to remember:
for \( j \geq 1 \) to \( n \)
for \( i \geq 1 \) to \( n \)

A nested loops

When might this come in handy?

\[
\left( \frac{3}{n^2} \right) = 0 (n^2)
\]

\[
\frac{1}{n} = \frac{n}{n(n+1)}
\]

for any \( n \geq 1 \)

Another useful thing:
\[
\begin{align*}
\log_b c &= \log_b a + c \\
-6 \log_b c &= \log_b a \\
-6 \log_b c &= \log_b b^b \\
-6 \log_b c &= 6 \\
\log_b a &= 6 \\
\log_b a &= b^6 \\
\log_b a &= \sqrt[b]{x^6} \\
\log_b a &= \log_b (x^6) \\
\log_b a &= c \\
\log_b a &= \log_b (a/c) \\
\log_b a &= \log_b (a/c) - \log_b c \\
\log_b a &= \log_b (a/c) + \log_b c
\end{align*}
\]

Logarithms (see p. 115)