## CS10: The Beauty and Joy of Computing

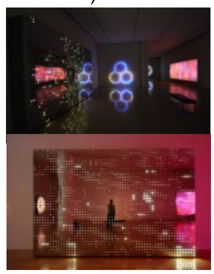
**Lecture #7: Algorithm Complexity**

TA Jon Kotker (2010-09-27)

**LEDs + Math = Art**

Leo Villareal combines modern LED control systems to produce contemporary modern art. The exhibit is on display at the San Jose Museum of Art.

http://news.cnet.com/8301-13772_3-20017310-52.html



## BIG IDEA

- Many ways to do the same thing = many *algorithms* to accomplish the same task.
- Example: Distributing candy!
- Example: Searching through a list of numbers to find a specific number.
  - *Linear* search (list is unsorted): Go through the list number by number and check if each number is The One.
  - *Binary* search (list is sorted): Look at the middle of the list. If it is not The One, break the list into two smaller halves and ignore the irrelevant half.
  - Any other algorithms?



## MAKING A DECISION

How do we decide which algorithm to use?

- Which is easier to implement?
- Which takes less time?
- Which uses up less space (memory)?
- Which gives a more precise answer?
- Which of the above questions even *matter*?



## WHAT DO *YOU* THINK?

Which of the factors below is most important in making a choice between two algorithms?

A. Which is easier to implement?
B. Which takes less time?
C. Which uses up less space (memory)?
D. Which gives a more precise answer?
E. I don't know / I don't have an opinion.



## RUNTIME ANALYSIS

One commonly used criterion in making a decision is **runtime** – how much time does the algorithm take to run and finish its task?

Computers are most useful for large inputs, so find the runtime of the algorithm on large inputs.

*How do we do that?*



## RUNTIME ANALYSIS

Time the algorithm with a stopwatch! But…

- Different computers will have different runtimes. ☹
- Same computer can have different runtime on the same input. ☹
- Need to implement the algorithm first so that we can run it. o_O;

*Solution*: Need to somehow *abstract* the computer away from the algorithm.

## RUNTIME ANALYSIS

*Idea*: Do not focus on how long the algorithm takes on one input. Instead, focus on how the **worst-case** runtime of the algorithm *scales* as we scale the input.

*Why?* Abstracts the computer out. A good algorithm should work well, no matter what the computer = a good recipe should produce a great dish, no matter what the kitchen.



## RUNTIME ANALYSIS

*Idea*: Do not focus on how long the algorithm takes on one input. Instead, focus on how the **worst-case** runtime of the algorithm *scales* as we scale the input.

*Why?* Computers are mainly used for large sets of data. The runtime of an algorithm should scale "reasonably" as we make the dataset even larger, or else we need to improve/discard that algorithm.




Dangerous for Math majors.

**A *LOT* OF APPROXIMATION AHEAD!**

Courtesy http://t5bg.roblox.com/bdc77779cfe00dc3291623e373969193



## ARITHMETIC OPERATIONS


*Key Idea*: As the input scales, arithmetic operations take approximately the same time. Arithmetic operations are **constant-time** operations.

*Another Key Idea*: We only care about how the runtime of the block scales as the input *scales*!



## LAUNDRY

*It Must Be Done*



## WASHING LOADS


If each load takes about the same time to launder, how does the runtime scale as the number of loads doubles? Triples?

## WASHING LOADS


*Key Idea*: The runtime of the algorithm *scales by the same amount* as the size of its input scales.

Doing laundry is a **linear-time** operation *in the number of loads*.



## FINDING CLOTHES TO WEAR


How does the **worst-case** total time to find a good pair scale as the number of shirts and pants doubles?

1. Stays the same.
2. Doubles.
3. Halves.
4. Quadruples.
5. Buh?

*Hint*: For each shirt that I own, how many pants do I have to match against it?



## FINDING CLOTHES TO WEAR


*Key Idea*: If I have 3 shirts and pants, there are 9 different combinations that I can try, because for *each* shirt, I can try on 3 pants to match with it.

Double it: If I have **6** shirts and pants, there are **36** different combinations that I can try.

If I double the number of shirts and pants that I have, then the number of different combinations that I can try **quadruples**.



## FINDING CLOTHES TO WEAR


*Key Idea*: The runtime of the algorithm scales by the **square** of the amount that the input scales by.

Finding a good pair of clothes to wear is a **quadratic-time** algorithm *in the number of shirts and pants*.



## LAUNDRY

*It Has Been Done*



## RUNTIME ANALYSIS

What is the runtime of this script?


1. Constant in `num`.
2. Linear in `num`.
3. Quadratic in `num`.
4. Buh?

## RUNTIME ANALYSIS

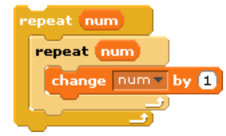
What is the runtime of this script?


1. Constant in `num`.
2. Linear in `num`.
3. Quadratic in `num`.
4. Buh?



## RUNTIME ANALYSIS

What is the runtime of this script?




1. Constant in `num`.
2. Linear in `num`.
3. Quadratic in `num`.
4. Buh?



## RUNTIME ANALYSIS

What is the runtime of this script?


1. Constant in `num`.
2. Linear in `num`.
3. Quadratic in `num`.
4. Buh?



## IT'S ALL APPROXIMATE!

Which is better: a **linear-time** algorithm or a **quadratic-time** algorithm?

| Input Size (N) | 10 | 100 | 1000 | 10000 | 100000 |
|---|---|---|---|---|---|
| Linear (msec) | C | 10C | 100C | 1000C | 10000C |
| Quadratic (msec) | C | 100C | 10000C | $10^6$C | $10^8$C |

As the input size increases, the quadratic-time algorithm takes so much more time than the linear-time algorithm that the linear-time algorithm is *negligible* in comparison.



## IT'S ALL APPROXIMATE!

Which is better: a **linear-time** algorithm or a **quadratic-time** algorithm?

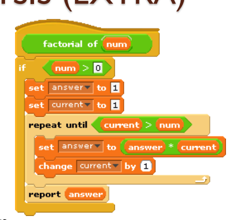
| Input Size (N) | 10 | 100 | 1000 | 10000 | 100000 |
|---|---|---|---|---|---|
| Linear (msec) | C | 10C | 100C | 1000C | 10000C |
| Quadratic (msec) | C | 100C | 10000C | $10^6$C | $10^8$C |

Since we only consider large sized inputs, expressions like $N^2 – N$ or $N^2 + N$ are considered approximately equal to N and thus **quadratic-time**; the linear-time part is ignored.



## RUNTIME ANALYSIS (EXTRA)

What is the runtime of this algorithm to find the factorial of a number?




1. Constant in the number.
2. Linear in the number.
3. Quadratic in the number.
4. Buh?