

Algorithms

CSE 120 Winter 2020

Instructor:

Sam Wolfson

Teaching Assistants:

Yae Kubota

Eunia Lee

Erika Wolfe

Quiz 1

- You will have 20 minute to complete the quiz.
- No outside materials are allowed – just your mind 😊
- If you have a question, raise your hand.
- Finish: 3:51

Administrivia

- ❖ Monday (1/20) is a holiday – no class!
 - Yae may hold informal “office hours” – email her if you are interested (ykubota@uw.edu)

- ❖ Assignments:
 - Logo Design [submit] is due Monday (1/20)
 - You should submit four different sketches of your logo as well as a detailed mockup.
 - Lego Family [submit] due on Friday (1/24)
 - Make sure you read the rubric *before* submitting!

Definition

- ❖ An **algorithm** is “any *well-defined* computational procedure that takes some value, or set of values, as input and produces some value, or set of values, as output.”

“An algorithm is thus a *sequence of computational steps* that transform the input into the output.”

- From textbook *Introduction to Algorithms* ([link](#))



Computational Problems

- ❖ Can think of an algorithm as a tool for solving a **computational problem**
 - The problem statement specifies desired input/output (I/O) relationship
 - The algorithm describes a specific computational *procedure* that gives you the desired input/output (I/O) relationship
- ❖ Example: Sorting is a computational problem
 - Problem statement: Given a sequence of numbers, put them in order
 - Example I/O: $[1, 9, 3, 6, 2] \rightarrow [1, 2, 3, 6, 9]$

Early Algorithms

- ❖ The concept of algorithms pre-dates computers
 - Dances, ceremonies, recipes, and building instructions are all *conceptually similar* to algorithms
 - Mathematical algorithms go way back:
 - Babylonians defined many fundamental procedures ~3600 years ago, more formal algorithms in Ancient Greece
 - [Al-Khwarizmi](#) laid out many algorithms for computation using decimal numbers
 - You implicitly use hundreds of numerical algorithms!
 - Nature runs algorithms (*e.g.* genes and development)

Properties of Algorithms

- ❖ Algorithms can be combined to make new algorithms
 - It helps to know standard algorithms
 - Building from correct algorithms helps ensure correctness
- ❖ There are many algorithms to solve the same computational problem
- ❖ Developing a new algorithm to solve a problem can lead to insight about the problem

Algorithms You've Seen

- ❖ Create a Taijitu from rectangles and ellipses
- ❖ Converting a binary number to decimal
- ❖ Make a character move into place on the screen
- ❖ ... and many more!

An Algorithm You've Seen Before

- ❖ Multiplying two numbers:

$$\begin{array}{r} 2 \\ 23 \\ \times 7 \\ \hline 161 \end{array}$$

- ❖ Another multiplication algorithm?
 - Common core “box” method:

$$\begin{array}{r} 20 + 3 \\ 7 \boxed{140} \boxed{21} \\ 140 + 21 = 161 \end{array}$$

More Famous Algorithms

- ❖ PageRank algorithm
 - Google's measure of the "reputation" of web pages
- ❖ EdgeRank algorithm
 - Facebook's method for determining what to show on your News Feed
- ❖ Luhn algorithm
 - Credit card number validation
- ❖ Deflate
 - Lossless data compression
- ❖ RSA Encryption
 - Encrypt (secure) data for transmission

Peanut Butter Jelly Time!



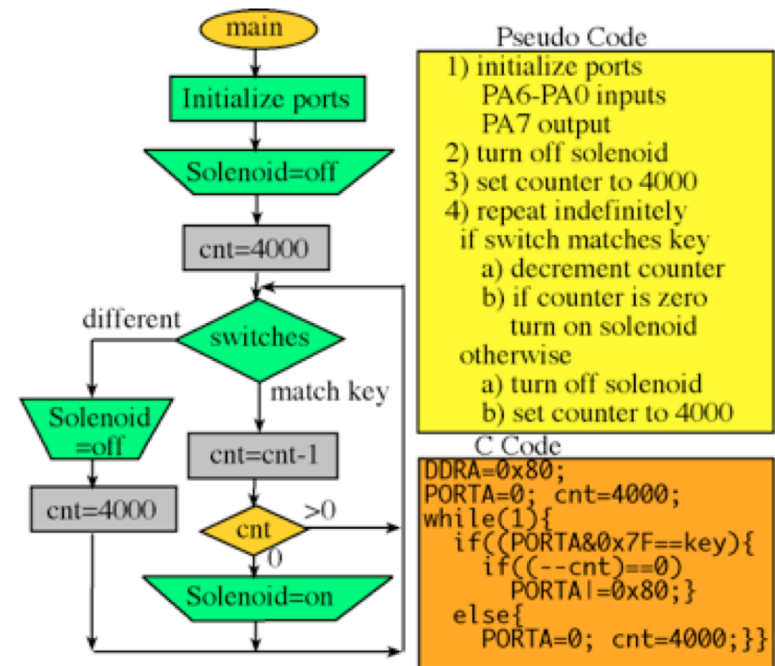
https://youtu.be/cDA3_5982h8

Be Specific!

- ❖ Algorithms need to be expressed in an *unambiguous* way for all participants
- ❖ Natural language is often ambiguous
 - “Time flies like an arrow, fruit flies like a banana”

Ways to Express Algorithms

- ❖ Many ways to specify an algorithm:
 - Natural Language (e.g. English, 中國) or Pseudocode
 - Easy for humans to understand, but can be ambiguous or vague
 - Visual and text-based programming languages (e.g. Scratch, Processing)
 - Have an exact meaning with no ambiguity
 - Can be run on a computer
 - Other information-conveying ways
 - e.g. flowcharts or pictures



Google Query

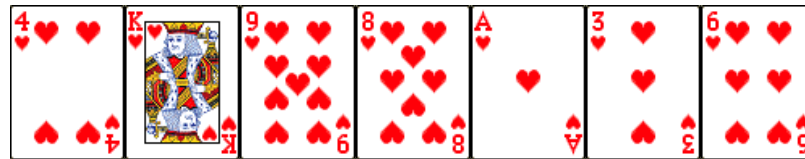
- ❖ From Larry Page and Sergey Brin's research paper *The Anatomy of a Large-Scale Hypertextual Web Search Engine*

1. Parse the query.
2. Convert words into wordIDs.
3. Seek to the start of the doclist in the short barrel for every word.
4. Scan through the doclists until there is a document that matches all the search terms.
5. Compute the rank of that document for the query.
6. If we are in the short barrels and at the end of any doclist, seek to the start of the doclist in the full barrel for every word and go to step 4.
7. If we are not at the end of any doclist go to step 4.
Sort the documents that have matched by rank and return the top k.

Figure 4. Google Query Evaluation

Specifying Algorithms Practice

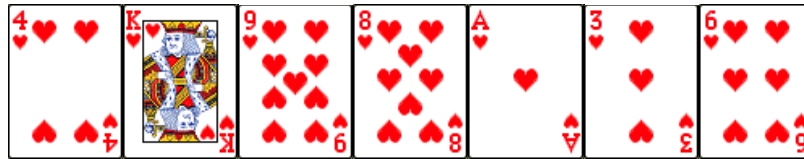
- ❖ The following algorithms will “operate” on playing cards



- Starting hand:
 - We will be playing the role of a computer
-
- ❖ Rules and restrictions:
 - Using values: $A \rightarrow 1$, $J \rightarrow 11$, $Q \rightarrow 12$, $K \rightarrow 13$
 - You can only *look* at one card/value at a time
 - Anything you need to *remember* must be stored in a “variable”
 - *Comparisons* can only be done between two values

1) Search an Unordered List

❖ Input:



, desired card

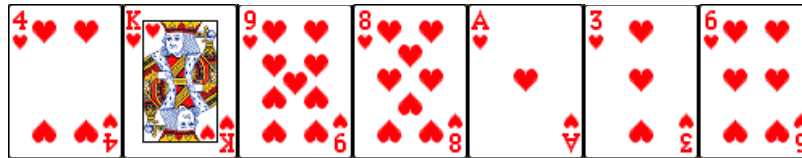
❖ Output: Yes if desired number is in the list,
No otherwise

❖ Algorithm:

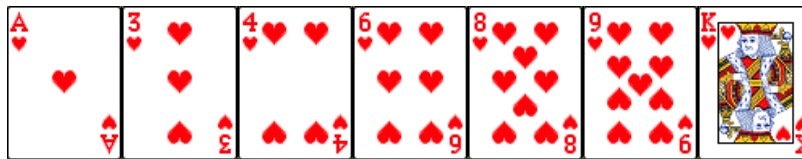
- Check each card starting from the left
 - If card is the one you're looking for, then report *Yes*
 - If a different card, then move on (don't report)
- If done with numbers, then report *No*

2) Sort an Unordered List (version 1)

❖ Input:



❖ Output: The same list, but now in numerical order

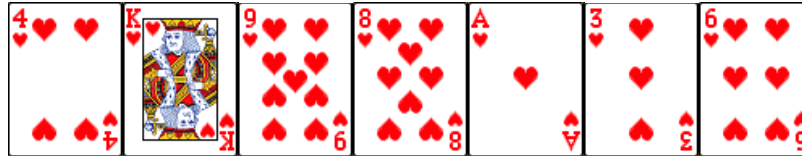


❖ Algorithm:

- Starting from the left, compare two adjacent cards and swap them if they are not in order. Move *one* card over and repeat until you reach the end of the list.
- Repeat until list is ordered.
- This algorithm is called *Bubble Sort*

3) Find the Smallest Number in a List

❖ Input:



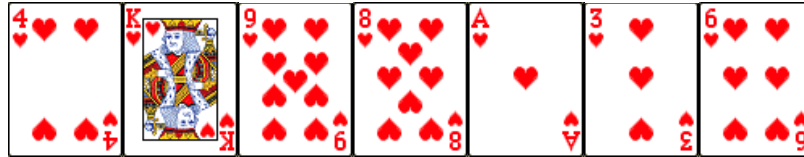
❖ Output: The *index/position* of the smallest number (5 in this case, since Ace is in the 5th position)

❖ Algorithm:

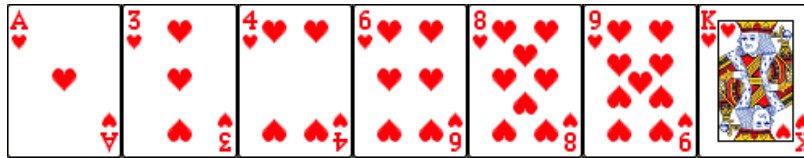
- Look at the first card and write down (or remember) the card value and the index 1.
- Check the rest of the cards 1-by-1: if card value is smaller, then write (or remember) the new value and index.

4) Sort an Unordered List (version 2)

❖ Input:



❖ Output: The same list, but now in numerical order




❖ Algorithm:

- Find the smallest number (algorithm 3) and move it to the front of the list.
- Repeat this entire procedure, but for the rest of the list.
- This algorithm is called *Selection Sort*

5) Find Median of a List

- ❖ **Note:** This is an actual job interview question!
- ❖ Problem: Given a list of numbers (odd length, no repeats), return the median

- ❖ Example:  should output 6

- ❖ Algorithm:
 - Sort the list (algorithm 2 or 4).
 - Take the number in the middle index $(N+1)/2$.

Sorting

- ❖ Sorting is a *mind-bogglingly* common task
 - Still difficult because it depends on (1) how many things you are sorting and (2) the initial ordering
- ❖ We showed some simple sorting algorithms, but there are a lot cleverer ones
 - <https://visualgo.net/bn/sorting>
 - <https://www.toptal.com/developers/sorting-algorithms>

Implementations

- ❖ If we specify an algorithm using code in a programming language, we say that the code **implements** the algorithm
 - A function or program that can be run on a computer

- ❖ Example: Find index of smallest in list

- Algorithm 3 of demo
 - Pseudocode/natural language
- Function in Processing
 - Implementation

we'll learn how to write things like this later

```
9 // returns the index of the smallest number in a list
10 int find_smallest(float[] list) {
11     int smallest = 0;
12     for(int i = 1; i < list.length; i=i+1) {
13         if(list[i] < list[smallest]) {
14             smallest = i;
15         }
16     }
17     return smallest;
18 }
```

Which Language to Use?

- ❖ Different languages are better suited for expressing different algorithms
 - Some languages are designed for specific domains, and are better for expressing algorithms in those domains
 - *e.g.* Unix shell scripts, SQL, HTML
 - https://en.wikipedia.org/wiki/Domain-specific_language
 - Language choice can affect things like efficiency, portability, clarity, or readability
 - Clarity and readability are VERY important considerations
 - Doesn't affect existence of algorithmic solution

Conceptual Question

- ❖ I hand you a cooking recipe with no name – is this equivalent to a *computational problem*, *algorithm*, or *implementation*?
 - What can and can't I do with just the recipe?
 - Talk with your neighbors!

A. **Computational Problem**

B. **Algorithm**

C. **Implementation**

Summary

- ❖ A **computational problem** is a problem statement with desired I/O relationship
- ❖ An **algorithm** *describes* a computational procedure that satisfies/solves a computational problem
- ❖ An **implementation** is an algorithm written out in a programming language (unambiguous, executable)
- ❖ Properties of algorithms:
 - Can be combined to make new algorithms
 - Knowledge of existing algorithms & correctness help
 - There are many algorithms to solve the same computational problem