Class 17 - Notes

Upcoming Schedule

By 11:59am tomorrow (Thursday), submit the Pre-Break Course Survey.

Project 4 is due Wednesday, 16 March (but if you don't get started this week, you should expect to work on it over spring break).

By Friday, March 4, everyone should have read Chapter 7: *Cost* of the course book and completed Lesson 5: *How Programs Run* of the Udacity course.

Cost of Addition

```
def mba_add(a, b):
    if b == 0:
        return a
    else:
        return 1 + mba_add(a, b - 1)

def mba_add(a, b):
    for _ in range(b):
        a = a + 1
    return a
```

How does the amount of work required to execute `mba_add` (either version) scale with the *magnitude* of the inputs?

How does the amount of work required to execute `mba_add` (either version) scale with the *size* (length) of the inputs? (Does it matter if the inputs are written using decimal digits or binary bits?)

Third-Grade Addition

```
def generate_addition_table():
    entries = []
    for a in range(10):
        for b in range(10):
            val = (a + b) % 10
            carry = (a + b) > 10
            entries.append("('" + str(a) + "', '" + str(b) + "'): ('" +
                           str(val) + "', " + str(carry) + ")")
    return "{" + ', '.join(entries) + "}"

ADDITION_TABLE = { ('0', '0'): ('0', False), ..., ('9', '9'): ('8', True)}

NEXT_DIGIT = {'0': '1', '1': '2', '2': '3', ..., '7': '8', '8': '9'}

def add_one(a, b, carry):
    value, newcarry = ADDITION_TABLE[(a, b)]
    if carry:
        if value == '9':
            value = '0'
            assert not newcarry
            newcarry = True
        else:
            value = NEXT_DIGIT[value]
    return value, newcarry

def thirdgrade_add(a, b):
    adigits = [digit for digit in list(str(a))]
    bdigits = [digit for digit in list(str(b))]
    adigits.reverse()
    bdigits.reverse()
    maxlen = max(len(adigits), len(bdigits))
    while len(adigits) < maxlen: adigits.append('0')
    while len(bdigits) < maxlen: bdigits.append('0')
    assert len(adigits) == len(bdigits)
    result = []
    carry = False
    for digits in zip(adigits, bdigits):
        value, carry = add_one(digits[0], digits[1], carry)
        result.append(value)
    if carry:
        result.append('1')
    result.reverse()
    return ''.join(result)
```

Problems, Procedures, Algorithms, and Programs

A **problem** is defined by the set of possible inputs and the desired property of the output.

A **procedure** is a precise decription of an information process.

An **algorithm** is a *procedure* that solves a *problem*. To *solve* a problem, an algorithm must (eventually) produce the correct output for any problem input. (This means it must always finish!)

A **program** is a description of a procedure that can be executed by a computer. A **Python 3 program** is a description of a procedure that can be executed by a Python 3 interpreter.

Addition

What is the Addition problem?

Inputs:

Output:

Cost

The **cost of a problem** is the cost to execute the least expensive algorithm that can solve the problem. Knowing the cost of a problem precisely is extremely difficult since it means knowing the *best possible* way of solving that problem.

The **cost of an algorithm** is the cost to execute that algorithm on some computer (or abstract computing model). Knowing the cost of an algorithm is just a matter of understanding what the algorithm does on all inputs (which may still be hard, but is reasoning about a concrete description of a procedure).

Cost of Addition

What is the cost of the `mba_add` algorithm?

What is the cost of the Addition problem?

Merkle's Puzzles

Protocol:

N is the number of puzzles.
w is the amount of work to solve each puzzle.

How much work does the legitimate receiver need to do?

How much work does the eavesdropper need to do?

Links

Ralph Merkle's history of public-key cryptography, including his (rejected) undergraduate course proposal, and rejection letter (that is actually quite savvy, talking about the work advantage being too little).

Applied Cryptography section on Merkle's Puzzles (this is a direct link to the videos, or you can watch them with the interactive quizzes in the Udacity player

Cryptography Pioneers Win Turing Award, New York Times, 1 March 2016.

Turing Award Announcement, 29 February 2016.

