CS98-52 Lecture 3

Wednesday, September 18th, 2019

Church Numerals

- Church numerals are a way of representing numbers with functions, as in lambda calculus.

```
def zero(f):
    return lambda x: x

def successor(n):
    return lambda f: lambda x: f(n(f)(x))

def one(f):
    """Same as successor(zero)"""
    return lambda x: f(x)

def two(f):
    """Same as successor(successor(zero))"""
    return lambda x: f(f(x))
```

- We can convert church numerals to Python integers with:

```
def church_to_int(n):
    return n(lambda y: y+1)(0)
```

- By now, you should have a clear understanding that Church numerals are simply functions applied to the same `x` over and over again.

```
def add_church(m,n):
    """Adds two church numerals M and N.

    >>>church_to_int(add_church(two,three))
    5
    """
    return lambda f: lambda x: n(f)(m(f)(x))

def mul_church(m,n):
    return lambda f: lambda x: m(n(f))(x)
    #This code tells us to apply F N times, then do that M times again.

def pow_church(m,n):
    return lambda f: lambda x: m(n)(f)(x)
    #This code tells us to apply F M*N times.
```

Strategies

What if instead of just averaging and collecting data, we could just get the probability of something working out before doing it?

Check slides for background info

- What's the chance of rolling a 1 in python?

```
def roll_a_one(n):
    if n==0:
        return 0
    return dice(1) + (1-dice(1))*roll_a_one(n-1)
```

- This tells the code that your chance of rolling a 1 is either you roll a 1, or you roll some number other than 1, then a dice numbered 1.

```
def roll_no_ones(total,n):
    if total==0 and n==0:
        return 1
    elif n==0:
        return 0
    else:
        chance, outcome = 0, 2
        while outcome <= 6:
            chance += dice(outcome)*roll_no_ones(total-outcome,n-1)
            outcome += 1
        return chance
```

Hog: The End Game

- Let's say that both players have 99 points, and let's focus only on the pig out rule. It doesn't matter what you do. If it's your turn, you win.
- Take this alternate scenario. You have 88 points, and your opponent has 99. How do you maximize your chance of winning?
- Or what if you have 80?
- You ask: what is the chance you score `n` points rolling `k` dice?

```
def roll_at_least(k,n):
    """P(At least K points with N dice)"""
    total, chance = k, 0
    while total <= k, 6*n:
        chance += roll_dice(total,n)
        total += 1
    return chance
```

- This function calculates your chance of getting a number, or any number between that and the maximum.
- This gets very slow as you roll a higher number of dice, because the functions calls a lot of other functions in the process.
- We can improve this speed by using what many people call memoize.

Memoize

- Python calls this the `lru_cache`, or least recently used cache.

```
from functools import lru_cahce
memoize = lru_cahce(None)
```

- We can then use decorators to call memoize on `roll_dice`.

```
@memoize
def roll_dice(n):
    ...
```

Twenty-One the game

- Two players alternate turns, on which they can add 1, 2, or 3 to the current total.
- The total starts at 0.
- The game ends whenever the total is 21 or more.
- The last player to add to the total loses.
- Some states are good; some are bad:
  - 17, 18 and 19 are good states, because you can push the total to 20 and force your opponent to lose.
- So what's your strategy to win?
- Any strategy should be able to look at the total, and return whether you should roll 1, 2 or 3.