Lecture 8

More recursion/memoization examples:

Example 1: Let `numWays`(n) be the number of ways to write a nonnegative integer n as the sum of positive integers. For example, there are 8 ways of writing 4: $1+1+1+1$, $2+1+1$, $1+2+1$, $1+1+2$, $2+2$, $1+3$, $3+1$, and 4. One can show by induction that `numWays`(n) $= 2^{n-1}$, but let's see how to calculate it using recursion and memoization.

Recursive implementation without memoization:

```
def numWays(n):
    if n==0:
        return 1
    ans = 0
    for i in xrange(1, n+1):
        ans += numWays(n-i)
    return ans
```

Recursive implementation with memoization:

```
def memNumWays(n, mem):
    if n==0:
        return 1
    elif mem[n] != -1:
        return mem[n]
    mem[n] = 0
    for i in xrange(1, n+1):
        mem[n] += memNumWays(n-i, mem)
    return mem[n]

def numWays(n):
    mem = [-1]*(n+1)
    return memNumWays(n, mem)
```

Example 2: What if we want to compute a function `distinctNumWays`(n) which doesn't differentiate between different orderings of the same sum? For example, it treats $1+1+2$ and $2+1+1$ as the same sum. So, there would only be 5 ways to sum up to the number 4: $1+1+1+1$, $1+2+2$, $2+2$, $1+3$, 4.

We can calculate `distinctNumWays`(n) recursively as well, by generating all ways of forming n where the integers in the sum are generating in nondecreasing order. That is, we would not generate $2+1+1$ or $1+2+1$ since the integers do not appear in nondecreasing order; we would only generate $1+1+2$. That way, we never count each sum exactly once.

Recursive implementation without memoization:

```
# how many ways are there to sum up to n, not counting different
# orderings of the sum, when the smallest number must be at least
# atLeast
def recurse(n, atLeast):
    if n==0:
        return 1
    ans = 0
    for i in xrange(atLeast, n+1):
        ans += recurse(n-i, i)
    return ans

def distinctNumWays(n):
    return recurse(n, 1)
```

Recursive implementation with memoization:

```
def recurse(n, atLeast, mem):
    if n==0:
        return 1
    elif mem[n][atLeast] != -1:
            return mem[n][atLeast]
    mem[n][atLeast] = 0
    for i in xrange(atLeast, n+1):
        mem[n][atLeast] += recurse(n-i, i, mem)
    return mem[n][atLeast]

def distinctNumWays(n):
    mem = [[-1]*(n+1)]*(n+1)
    for i in xrange(n+1):
        x = []
        for j in xrange(n+1):
            x += [-1]
        mem += [x]
    return recurse(n, 1, mem)
```