Falling glass

(a)when we drop glass from X floor, there can be two case :

(1) glass breaks (2) glass doesn't break

if glass breaks after dropping from X floor, then we only need to check for floors lower than X with remaining glasses.So the problem will reduces to X-1 floors and n-1 glasses.

if the glass doesn't break after dropping from X floor. Then we only need to check for floors highner than X. So our problem will reduce to k-X floors and n glasses.

k will be # of floors and n will be # of

glasses glass drop(n,k)==》minimum number of trials which we needed to find the critical floor in worst case.

glassdrop(n,k)= 1+min{max (glassdrop (n-1, x-1), glassdrop(n, k-x))

x will be {1，2，...., k}}

（b) recurrence tree for give (floors = 4 , sheets =2) apply for two cases (break or not)

E(2,4)

x=1 x=2 x=3 x=4

E(1,0) E(2,3) E(1,1) E(2,2) E(1,2) E(2,1) E(1,3) E(2,0)

x=1 x=2 x=3 ...

E（1,0）E（2,2）

(d)

how many distinct subproblems do you end up with given 4 floors and two sheets.

There will many repeated subproblems when you draw the complete recursion tree.since same subproblems are clled again, this problem has Overlapping subproblems property. So it has both properties of a dynamic programming problem.

2^4-1 = 8

e) How many distinct subproblems for n floors and m sheets?

2^k-1

(f) Describe how you would memoize GlassFallingRecur

We can do this with the memorize function. memorize takes a function as input and returns another function which caches results as they are calculated and on subsequent calls with the same parameters just returns the previous calculation.