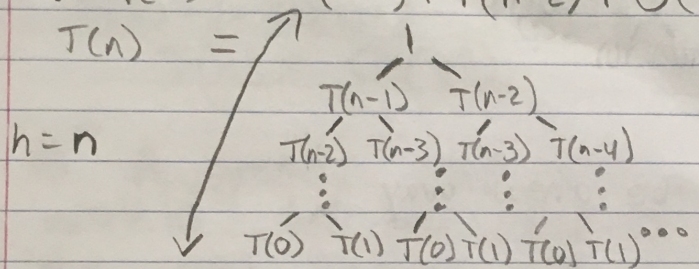
Jake Schinto

Homework #5

1.
```
fib(n)
1. if (n=0)
2.     return 0
3. else if (n=1)
4.     return 1
5. else if (n≥2)
6.     return fib(n-1) + fib(n-2)
7. return 0 // if n<0
```

2. $T(n) = T(n-1) + T(n-2) + \Theta(1)$

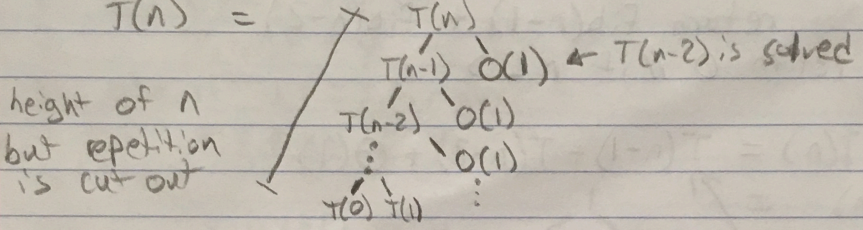


the height of the tree is n and doubles at each level with a run time of $\Theta(1)$ at each level, so at the bottom there is oproximatly 2^n leaf nodes making $T(n) = O(2^n)$

3.
```
fib(n, r)   ← r is array[0...n] originally containing -∞
1. if (r[n] ≥ 0)
2.     return r[n]
3. if (n=0)
4.     q = 0
5. else if (n=1)
6.     q = 1
7. else      // assuming n ≥ 2
8.     q = fib(n-1, r) + fib(n-2, r)
9. r[n] = q
10. return q
```

4. it is worst case $O(n)$ because when fib solves for each value $\leq n$ it saves the solution and becomes $O(1)$ to retreive it. Therefore, it only needs to run through each value of n once making it $O(1) \cdot n = O(n)$

$T(n) =$



5.
```
fib(n,r)
1. let r[0...n] be a new array
2. r[0] = 0
3. r[1] = 1
4. for j = 2 to n
5.     r[j] = r[j-1] + r[j-2]
6. return r[n]
```

6. It is $O(n)$ because there is only a single for loop that iterates to n and no recursion, therefore, the algorithm must run in $O(n)$ time

7. a. The cut-rod function has a for loop that iterates n times and calls the cut-rod function again with $n-i$ each time. This makes $T(n) = T(n-1) + T(n-2) + T(n-3) \ldots T(0) + 1$

↑ One for each iteration of the for loop

run time of rest of function

b. $T(n) = 2^n$

$T(n) = c2^{n-1} + c2^{n-2} + \ldots c2^0$

$= c2^n \left(\sum_{i=1}^{n} \frac{1}{2^i}\right)$ ← ≤ 1

$\leq c2^n$

$n_0 \geq 0$

$T(0) = 1 \quad T(0) = 2^0 = 1 \quad c = 1$

$T(1) = 2 \quad T(1) = 2^1 = 2$

$T(2) = 1+2+1 = 4 \quad T(2) = 2^2 = 4$

$T(3) = 1+4+2+1 = 8 \quad T(3) = 2^3 = 8$

$T(n) = O(2^n)$

8. A greedy way to solve this problem would be to stop at gas at the gas stations closest to, but still less than K miles from each previous stop made. The optimal choice from each stop is to go the furthest possible.

9. This is optimal because it maximizes the distance between each stop. If a certain stop happens to be in the optimal solution, the greedy solution would be able to get there in just as many or fewer stops than any other path. Each substructure is optimal.