Introduction

Fibonacci Number

0, 1, 1, 2, 3, 5, 8, 13 ...

$a_0\ a_1\ a_2\ a_3\ a_4$ ...

$$\begin{cases} a_n = a_{n-1} + a_{n-2} \\ a_0 = 0 \\ a_1 = 1 \end{cases}$$

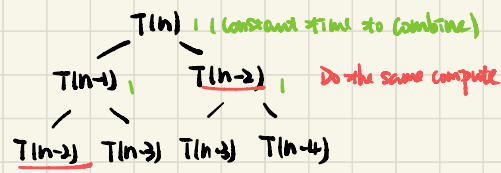
Question: Calculate a_n ?

#Solution 1 $O(2^n)$ exponential time

```
fib1(int n){
    if(n==0 || n==1)
        return n;
    return fib1(n-1) + fib1(n-2);
}
```

Recursion Tree

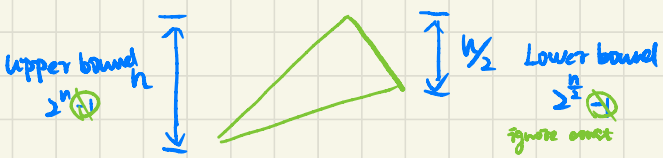
Let T(n) denotes the running time of fib1 given n



Running time is the same as the num of nodes in the recursion tree.

↳ How many nodes are there?

$2^k - 1$ (k level)



$$(1.4)^n \approx 2^{\frac{n}{2}} \le T(n) \le 2^n$$

#Solution 2 (Caching the results)

O(n) Linear time

```
int A[0, ..., n];
A[0]=0, A[1]=1;
for(i=2, ..., n){
    A[i] = A[i-1] + A[i-2];
}
return A[n];
```

Reduce Space

```
A[2];
A[0]=0, A[1]=1;
for(i=2, ..., n){
    A[i%2] = A[(i-1)%2] + A[(i-2)%2];
}
return A[n%2]
```

```
prev=0, cur=1; next=0
for(i=2, ..., n)
    next = prev+cur;
    prev=cur;
    cur=next;
return next
```

Solution 3

$$A = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \sim \begin{bmatrix} a_2 & a_1 \\ a_1 & a_0 \end{bmatrix}$$

$$A^2 = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix} \sim \begin{bmatrix} a_3 & a_2 \\ a_2 & a_1 \end{bmatrix}$$

$$\vdots$$

$$A^k = \begin{bmatrix} a_{k+1} & a_k \\ a_k & a_{k-1} \end{bmatrix}$$ ← yes

$O(n)$ each multiplying takes constant time

↓ using recursion to get $O(\log n)$

$$A^n = \begin{cases} \left(A^{\frac{n}{2}}\right)^2 & n \text{ is even} \\ \left(A^{\frac{n-1}{2}}\right)^2 \cdot A & n \text{ is odd} \end{cases}$$

eg. $A^{64} = \left(A^{32}\right)^2 = \left(\left(A^{16}\right)^2\right)^2 = \left(\left(\left(A^8\right)^2\right)^2\right)^2 = \cdots = \left(A^2\right)^{\overbrace{2\cdots2}^{2^6=64}}$ $\Rightarrow O(\log n)$

Asymptotic Analysis

O	$\leq$
o	$<$
Θ	$=$
Ω	$\geq$
ω	$>$

Compare function growth

$f(n) = 5n \quad g(n) = n^2$

$f(n) = O(g(n))$

or $f(n) = o(g(n))$

$f(n) = 100n \quad g(n) = n + 100$

$f(n) = O(g(n)),\ \Theta(g(n)),\ \Omega(g(n))$

Big-O : $f(n) = O(g(n))$ iff $\exists$ const c and n_0 st when $n > n_0$, $f(n) \leq c \cdot g(n)$ Def

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \begin{cases} 0 & \Rightarrow f(n) = o(g(n)) \\ \text{constant} & \Rightarrow f(n) = \Theta(g(n)),\ O(g(n)),\ \Omega(g(n)) \\ \infty & \Rightarrow f(n) = \omega(g(n)) \end{cases}$$

eg 1. $f(n) = n\log n$, $g(n) = n^2$

$f(n) = O(g(n))$, $o(g(n))$

2. $f(n) = 2^n$, $g(n) = 3^n$

$\lim_{n\to\infty} \frac{2^n}{3^n} = \lim_{n\to\infty} (\frac{2}{3})^n = 0$ case①

$f(n) = O(g(n)) = o(g(n))$

3. $f(n) = n+3$, $g(n) = 2\sqrt{n}$

$f(n) = \Omega(g(n)) = \omega(g(n))$

4. $f(n) = \log_2 n$, $g(n) = \log_3 n$

$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \frac{\lg 3}{\lg 2} = \log_2 3$ case②

$f(n) = \Omega(g(n)) = \Theta(g(n))$

⇒ changing base

$\lim_{n\to\infty} \frac{\log_2 n}{\log_3 n} = \frac{\lg n}{\lg 2} \cdot \frac{\lg 3}{\lg n} = \frac{\lg 3}{\lg 2}$

$$\log_a b = \frac{\log_c b}{\log_c a}$$

5. $f(n) = (\log n)^2$, $g(n) = \log n^2 = 2\log n$

$f(n) = \Omega(g(n)) = \omega(g(n))$

$$S_n = \frac{(a_1 + a_n)n}{2}$$

$$S_n = \frac{a_1(1-q^n)}{1-q} \quad (q \neq 1)$$

$1 + \underline{1+2+4+8+\cdots+1024} - 1$

$2+2$

4×2

8×2 ...

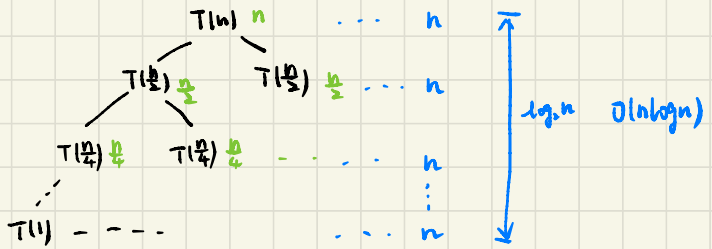
$1024 \times 2 - 1 = 2047$

Solving Recursion

- Recursion Tree
- Master Theorem

- Recursion Tree

$T(n) = 2T(\frac{n}{2}) + n$ (Merge Sort)



- Master Theorem

$T(n) = a \cdot T(\frac{n}{b}) + n^c$ (a, b, c are const) LIMITED IN USE

a: num of subproblem we need to conquer ; b: num of subproblem

① $\log_b a > c$ $T(n) = \theta(n^{\log_b a})$

② $\log_b a = c$ $T(n) = \theta(n^c \cdot \log n)$

③ $\log_b a < c$ $T(n) = \theta(n^c)$

eg 1. $T(n) = T(\frac{n}{2}) + 1$ Binary Search

$a=1,\ b=2,\ c=0$

$\log_b a = \log_2 1 = 0 = c$ case②

$T(n) = \theta(\log n)$

2. $T(n) = 8T(\frac{n}{2}) + n^2$

$a=8,\ b=2,\ c=2$

$\log_b a = \log_2 8 = 3 > c=2$ case①

$T(n) = \theta(n^3)$

3. $T(n) = 7T(\frac{n}{2}) + n^2$

$a=7,\ b=2,\ c=2$

$\log_b a = \log_2 7 > c=2$ case①

$T(n) = \theta(n^{\log_2 7})$

4. $T(n) = 4T(\frac{n}{2}) + n^2$

$T(n) = \theta(n^2 \log n)$

5. $T(n) = T(\frac{n}{2}) + n$

$T(n) = \theta(n)$

6. $T(n) = T(n-1) + n$

$T(n) = \theta(n^2)$

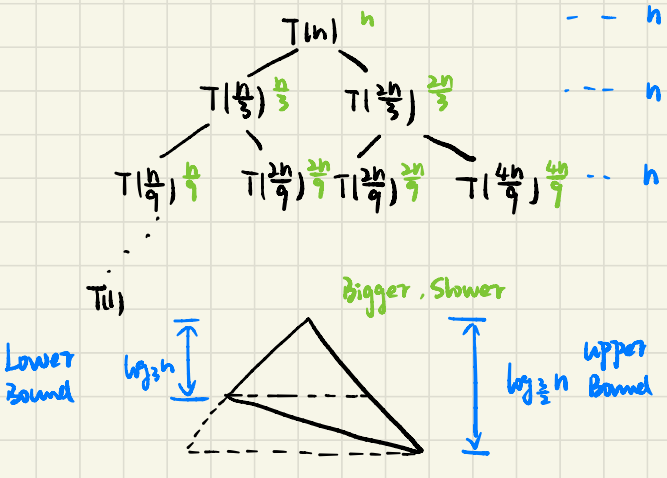
$T(n)$ n

$T(n-1)$ $n-1$

$\vdots$ $+$

$T(1)$ 1

7. $T(n) = T(\frac{n}{3}) + T(\frac{2n}{3}) + n$



$n \log_3 n \le T(n) \le n \log_{\frac{3}{2}} n \Rightarrow T(n) = \Theta(n \log n)$

8. $T(n) = T(\frac{n}{4}) + T(\frac{3n}{4}) + n$

$T(n) = \Theta(n \log n)$

9. $T(n) = T(n-1) + T(1) + n$

$T(n) = \Theta(n^2)$