Divide and conquer $\quad T(n) = T(\frac{n}{a}) + T(\frac{n}{b}) + T(\frac{n}{c}) + \cdots + O(n^2)$

polynomial time $O(n^{1.5})$

Backtracking $\quad T(n) = T(n-a) + T(n-b) + \cdots + O(n^2)$

Exponential time $O(1.6^n)$

Dynamic Programming $\quad$ Polynomial time

Pingala $\quad$ 600–200 BCE?

mātrāvṛtta $\quad$ short $\square$ $\quad$ long $\boxed{\quad\quad}$

Virahanka c.800 CE

$M(n)$ = # meters lasting n beats

$$M(0) = 1$$
$$M(1) = 1$$
$$M(n) = M(n-1) + M(n-2)$$

Fibonacci $\quad$ Liber Abaci 1202

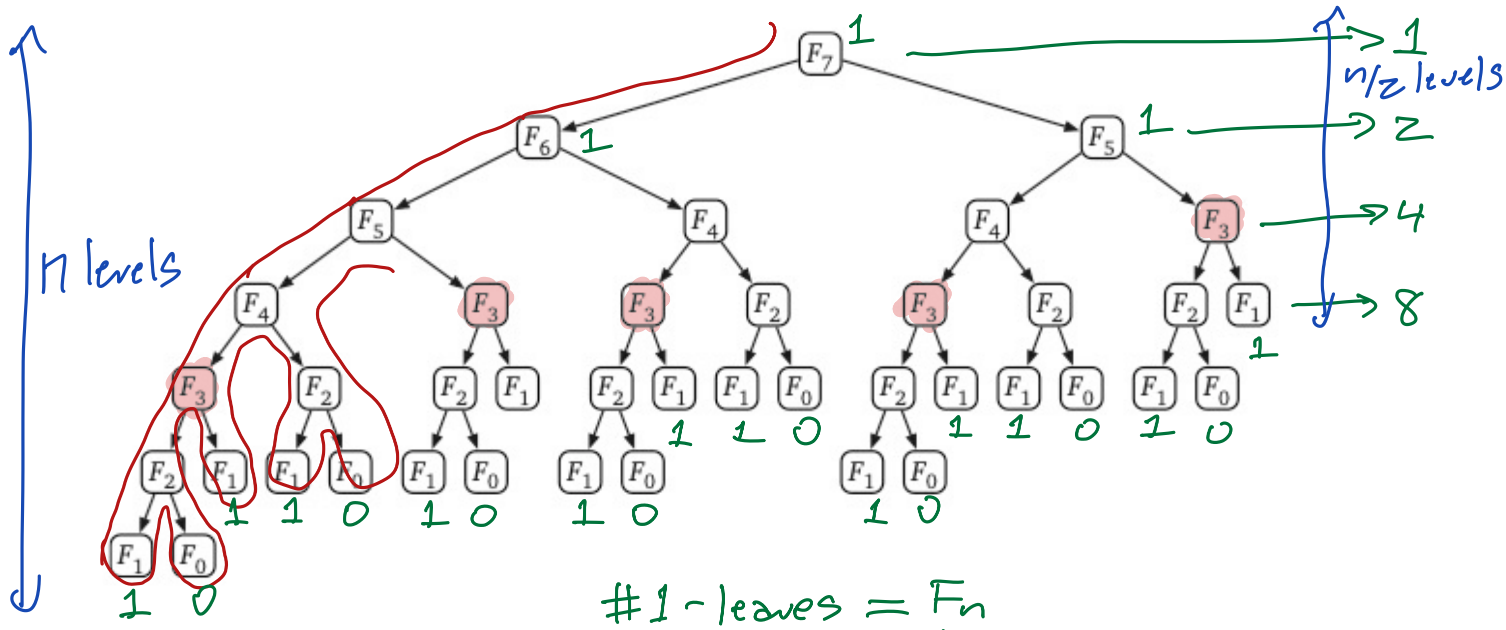
$$\boxed{\begin{aligned} F_0 &= 0 \\ F_1 &= 1 \\ F_n &= F_{n-1} + F_{n-2} \end{aligned}}$$

$$F_n = M(n-1)$$

```
RecFibo(n):
  if n = 0
      return 0
  else if n = 1
      return 1
  else
      return RecFibo(n − 1) + RecFibo(n − 2)
```

Backtracking

$T(n) = T(n-1)$
$+ T(n-2)$
$+ O(1)$



$\#1\text{-leaves} = F_n$
$\#0\text{-leaves} = F_{n-1}$

$\#\text{leaves} = F_{n+1} = \Theta(\phi^n)$

$\phi = \frac{1+\sqrt{5}}{2}$

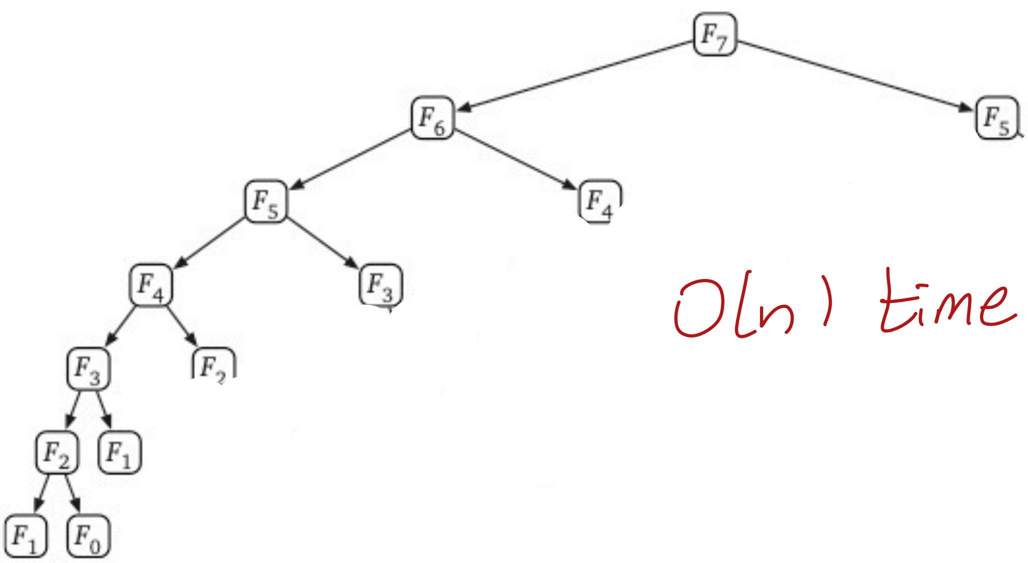
Donald Mitchie 1967

memorization

```
MemFibo(n):
  if n = 0
      return 0
  else if n = 1
      return 1
  else
      if F[n] is undefined
          F[n] ← MemFibo(n−1) + MemFibo(n−2)
      return F[n]
```

F:

0	1	1	2	3	5	8	13			
0	1	2	3	4	...					



$O(n)$ time

Dynamic Programming (Richard Bellman)

```
ITERFIBO(n):
    F[0] ← 0
    F[1] ← 1
    for i ← 2 to n
        F[i] ← F[i−1] + F[i−2]
    return F[n]
```

```
ITERFIBO2(n):
    prev ← 1
    curr ← 0
    for i ← 1 to n
        next ← curr + prev
        prev ← curr
        curr ← next
    return curr
```

Virahanka 800
Fibonacci 1202

$O(n)$ time
$O(1)$ space

$$\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} prev \\ curr \end{bmatrix} = \begin{bmatrix} curr \\ prev+curr \end{bmatrix} = \begin{bmatrix} curr \\ next \end{bmatrix}$$

$$\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^n \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} F_{n-1} \\ F_n \end{bmatrix}$$

```
⟨⟨Compute the pair F_{n−1}, F_n⟩⟩
FASTRECFIBO(n):
    if n = 1
        return 0, 1
    m ← ⌊n/2⌋
    hprv, hcur ← FASTRECFIBO(m)      ⟨⟨F_{m−1}, F_m⟩⟩
    prev ← hprv^2 + hcur^2           ⟨⟨F_{2m−1}⟩⟩
    curr ← hcur · (2 · hprv + hcur)  ⟨⟨F_{2m}⟩⟩
    next ← prev + curr               ⟨⟨F_{2m+1}⟩⟩
    if n is even
        return prev, curr
    else
        return curr, next
```

$O(\log n)$ arithmetic ops

```
SPLITTABLE(A[1..n]):
  if n = 0
      return TRUE
  for i ← 1 to n
      if ISWORD(A[1..i])
          if SPLITTABLE(A[i+1..n])
              return TRUE
  return FALSE
```

```
⟨⟨Is the suffix A[i..n] Splittable?⟩⟩
SPLITTABLE(i):
  if i > n
      return TRUE
  for j ← i to n
      if ISWORD(i, j)
          if SPLITTABLE(j+1)
              return TRUE
  return FALSE
```

$O(2^n)$ time

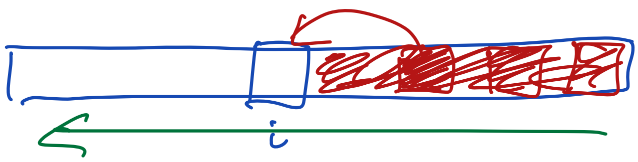
$$Splittable(i) = \begin{cases} \text{TRUE} & \text{if } i > n \\ \bigvee_{j=i}^{n} \big(IsWord(i,j) \wedge Splittable(j+1)\big) & \text{otherwise} \end{cases}$$

Splittable(i) = TRUE iff the suffix A[i..n] is splittable

Mnemonic
NOT "DP" or "OPT"

Memoize? Data structure = array SPLITTABLE[1..n]

Order?



Time? $O(n)$ subproblems
$\times\ O(n)$ calls to ISWORD for each
$= O(n^2)$ calls to ISWORD

```
FASTSPLITTABLE(A[1..n]):
  SplitTable[n+1] ← TRUE
  for i ← n down to 1
      SplitTable[i] ← FALSE
      for j ← i to n
          if ISWORD(i, j) and SplitTable[j+1]
              SplitTable[i] ← TRUE
  return SplitTable[1]
```

$O(n^2)$ calls to ISWORD