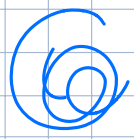
Backtracking → Exponential



2nd c. BCE Sanskrit ~~poetry~~ Pingali

0 - short
1 - long

- Fix # syllables
  vary short + long
- Fix # beats

short = 1 beat
long = 2 beats

1. 0 beats |
2. 1 beat S
3. 2 beats S S
   L
4. 3 beats S S S
   S L
   L S
5. 4 beats S S S S
   S S L
   S L S
   L S S
   L L

7th century India

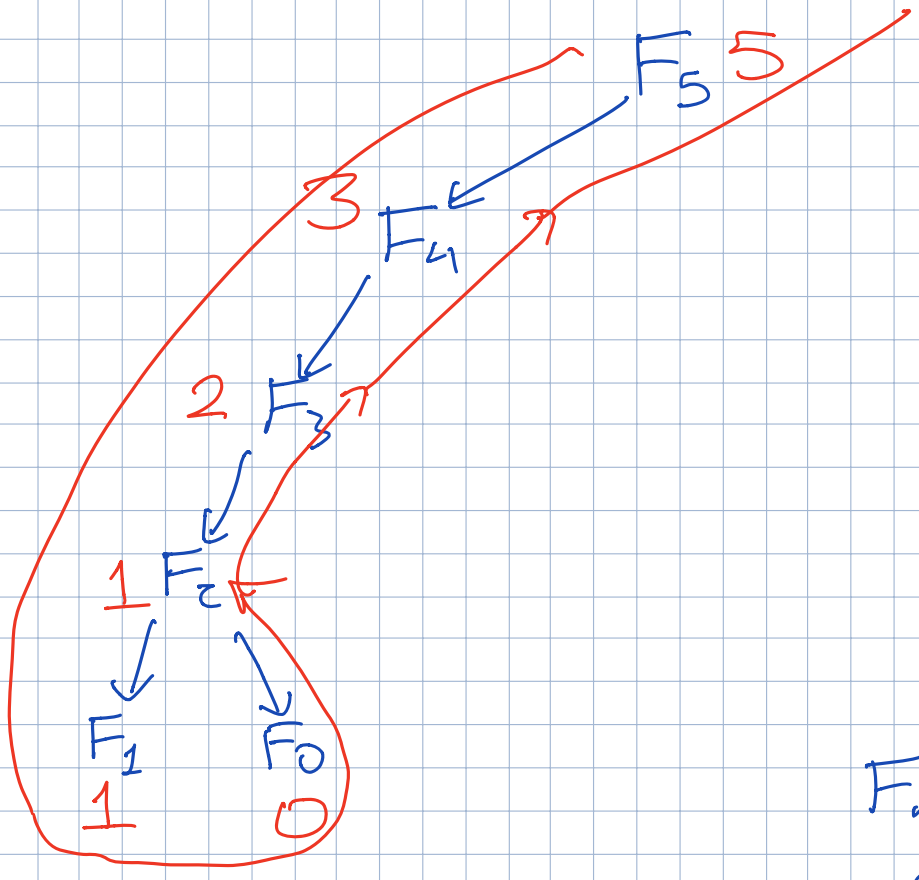
$M(n)$ = # meters n beats long

$$M(n) = \begin{cases} 1 & \text{if } n=0 \text{ or } n=1 \\ M(n-1)+M(n-2) & \text{o/w} \end{cases}$$

Fibonacci #s

$$F_n = \begin{cases} 0 & \text{if } n=0 \\ 1 & \text{if } n=1 \\ F_{n-1}+F_{n-2} & n>1 \end{cases} = M_{(n-1)}$$

```
RecFibo(n):
    if n ≤ 1
        return n  // assume n ≥ 0
    else
        return RecFibo(n-1) + RecFibo(n-2)
```



$$F_n = \Theta(\phi^n)$$

$$\phi = \frac{1+\sqrt{5}}{2} \approx 1.615$$

Memoization → O(n) time

F: [0] [1] [1] [2] [3] [5] | | | | |

F[k] ?

Von Neumann — Two-player games 1940s

Zobrist 1960s

IterFibo(n):

F[0] ← 0
F[1] ← 1
for i ← 2 to n
  F[i] ← F[i-1] + F[i-2]
return F[n]

F(n) =

O(n) "time"

```
FastFibo(n):
    prev ← 1
    cur ← 0
    for i ← 1 to n
        next ← prev + cur
        prev ← cur
        cur ← next

    return cur
```

$$\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} y \\ x+y \end{bmatrix}$$

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

$$\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^n = \begin{cases} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & \text{if } n = 0 \\ \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} & \text{if } n = 1 \\ \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^{\lfloor n/2 \rfloor} \cdot \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^{\lfloor n/2 \rfloor} \cdot \underbrace{\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}}_{\text{if } n \text{ odd}} \end{cases}$$

F_n in $O(\log n)$ ~~2×2~~ multiplications integer

---

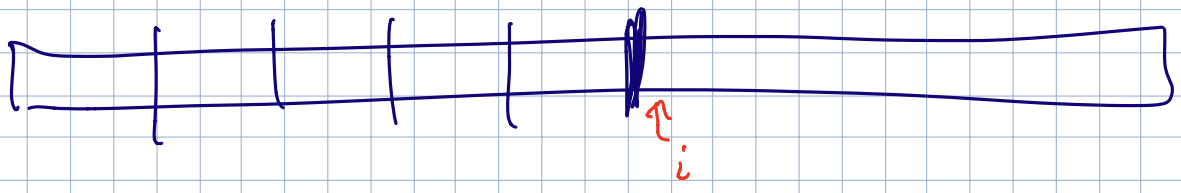
1. Develop recurrence (backtracking)
2. Find memoization structure
   → one entry per subproblem
   USE INDICES
3. Find evaluation order

Text segmentation

Given $A[1..n]$ of letters

Can it be split into English words?

Given IsWord(i,j) = TRUE iff $A[i..j]$ is an English word



Splittable(i) = TRUE iff $A[i..n]$ can be split into words

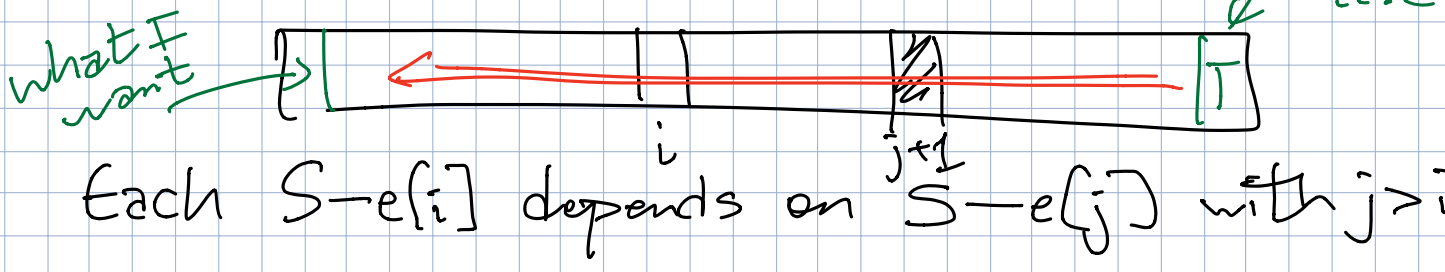
Splittable(i) = {
- TRUE if $i > n$
- TRUE if IsWord(i,j) and Splittable(j+1) for some $i \le j \le n$
- FALSE otherwise

}

(one param ⇒ $O(n)$ space; two params ⇒ $O(n^2)$ time)

We need Splittable(1).

Array Splittable[1..n+1]



Each S-e[i] depends on S-e[j] with $j > i$

So fill in decreasing index order.

$O(n^2)$ time

```
Splittable?(A[1..n]):
    Sp[n+1] ← True
    for i ← n down to 1
        Sp[i] ← FALSE
        for j ← i to n
            if IsWord(i,j) and Sp[j+1]
                Sp[i] ← True
    return Sp[1]
```