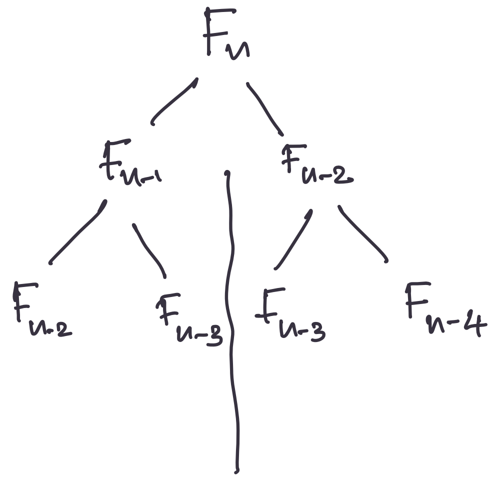
Dynamic Programming.

→ $F_n = F_{n-1} + F_{n-2}$ $F_0 = F_1 = 1.$

```
Fibonacci(n):
    <Handle base cases>.
    return Fibonacci(n-2) + Fibonacci(n-1)
```



- No. of sub problems is limited – F_i
- If we can store solutions of sub problems, we can compute faster.

Init: F[0]=1, F[1]=1

```
Fib(n):
    i=2
    while i≤n:
        F[i] = F[i-1]+F[i-2]
        i = i+1
    return F[n]
```

F[5]

1	1	2	3	5	8	13	21	34	55
0	1	2	3	4	5	6	7	8	9

- Divide the problem into "small" number of subproblems
- We should be able to efficiently put together the solution of the bigger problem from solutions to subproblems
- "structured" subproblems.

→ Memoization.

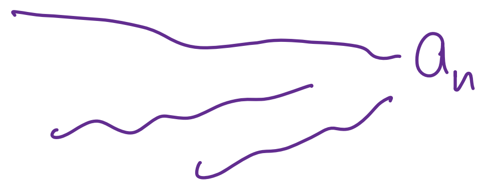
Longest Increasing Subsequence.

Sequence: $a_1, \dots, a_n$

↳ Want to find longest subseq

$a_{i_1} < a_{i_2} < \dots < a_{i_k}$ s.t $i_1 < i_2 < \dots < i_k$

Want to find the length of the longest subseq.



$A[1,\dots,n] = (a_1, \dots, a_n)$

1 - Longest subseq may not contain a_n.

↳ Look for LIS in $A[1,\dots,n-1]$

2 - Longest subseq may contain a_n:

↳ a_n is the maximal elem in that seq.

then look for LIS_smaller$(A[1,\dots,n-1], a_n)$

$A[1,\dots,n] = a_1, \dots, a_n$

← Function that outputs ~~a~~ len of longest IS in $A[1,\dots,i]$ s.t all elems in that IS have values $< x$.

LIS_smaller$(A[1,\dots,i], x)$:

$A[i] = a_i$

```
if i==0:
    return 0

m = LIS_smaller(A[1,...,i-1], x) ✓
if a_i < x:
    m = max{m, 1 + LIS_smaller(A[1,...,i-1], a_i)} ✓
return m.
```

LIS(A[1,...,n])

return LIS_smaller(A[1,...,n], ∞)

$A = 3, 6, 10, 15, 14, 4, 23$

$\widehat{LIS}((3,6,10,15,14,4,23), \infty)$

$m = \widehat{LIS}((3,6,10,15,14,4), \infty)$

$m' = \widehat{LIS}((3,6,10,15,14,4), 23)$

$m = \max\{m, 1+m'\}$.

$\widehat{LIS}(A[1,\dots,i], x)$

$\widehat{LIS}(A[1,\dots,i-1], x)$ $\quad$ $\widehat{LIS}(A[1,\dots,i-1], a_i)$

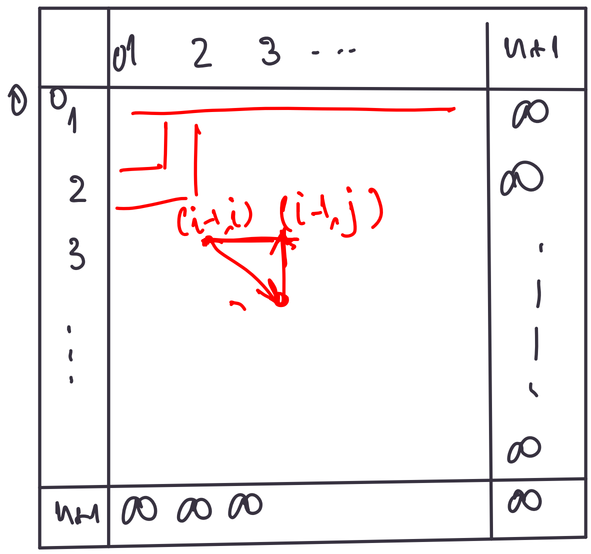
Practice recursion:

- Enumerate all k-sized subsets of $\{1,\dots,n\}$.
- Enumerate all +ve / non-ve integral solutions to the equation

$$x_1 + x_2 + \dots + x_n = k.$$

Rephrase LIS_smaller(A[1,...,i], a_j)

↳ LIS[i,j] ← length of longest incr. subseq. in A[1,...,i], with values smaller than a_j



$LIS[i,j]$ $\quad i<j$

$LIS[0,j] = 0 \quad \forall j$

$$LIS[i,j] = \begin{cases} LIS[i-1,j] & a_i > a_j \\ \max\{LIS[i-1,j],\ 1+LIS[i-1,i]\} & \text{otherwise} \end{cases}$$

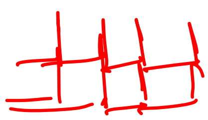
✓ a_i

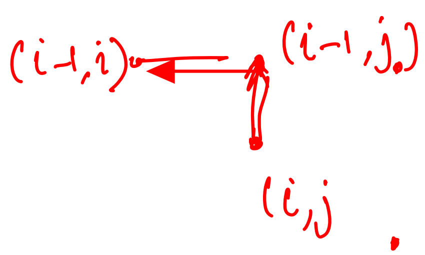
if $a_i < a_j$: $\quad$ LIS_smaller$(A[1,\dots,i-1], a_j)$

  return max{ m, 1 + $\underline{\text{LIS\_smaller}(A[1,\dots,i-1], a_i)}$ }

else:

  return m = LIS_smaller$(A[1,\dots,i-1], a_j)$.





i,j