• Fibonacci numbers (Basic Idea and Memorization)
• Shortest Path on Directed Acyclic Graphs (Ordering)
• Longest Common Subsequence (2-d tables)

- Fibonacci number
  \[ F(0) = 1 \quad F(1) = 1 \quad \forall n \geq 2 \quad F(n) = F(n-1) + F(n-2) \]
- how to compute \( F(n) \)
  
  - recursive solution
  
  \[
  \text{Fib}(n) =
  \begin{cases}
  1 & \text{if } n \leq 1 \\
  \text{Fib}(n-1) + \text{Fib}(n-2) & \text{return } \text{Fib}(n-1) + \text{Fib}(n-2)
  \end{cases}
  \]

- running time
  
  \[
  T(n) = T(n-1) + T(n-2) + O(1)
  \]

  \[
  T(n) = O \left( \left(\frac{\sqrt{5} + 1}{2}\right)^n \right)
  \]

- memorized search
  
  \[
  \text{Fib}(n) =
  \begin{cases}
  1 & \text{if } n \leq 1 \\
  \text{if } n \text{ is "solved", return solution}[n] \\
  R = \text{Fib}(n-1) + \text{Fib}(n-2) \\
  \text{mark } n \text{ as solved, solution}[n] = R \\
  \text{return } R
  \end{cases}
  \]

  \[
  T(n) = O(\# \text{ of elements in table} \times \text{amount of time per entry})
  \]

  \[
  = O(n \times 1) = O(n)
  \]
- **Iterative Solution**
  
  \[
  \text{Fib}(n) \\
  \text{if } n \leq 1 \text{ return } 1 \\
  F(0) = F(1) = 1 \\
  \text{for } i = 2 \to n \\
  F(i) = F(i-1) + F(i-2) \\
  \text{return } F(n)
  \]

- General idea of dynamic Programming (DP)
  - save intermediate results to avoid repeated computation
  - Design a DP algorithm
    - list, matrix, ...
    - (1) identify important subproblems (make a table)
    - (2) fill in the entries of the table in a "good" order

- **Shortest path in directed acyclic graphs**

  - directed acyclic graph (DAG)

  ![DAG Diagram]

  - Problem: Given a DAG, edge \( (i,j) \) has length \( \text{DAG}_{ij} \)
  - want to find shortest path from \( S \) to \( T \)
  - (the length)

  - **Recursive solution** (think: what is the last step of the solution)

  \[
  \text{shortest}(u) : \text{length of shortest path from } S \text{ to } u \\
  \text{if } u = S \text{ return } 0 \\
  \text{return } \min \left\{ \text{shortest}(v) + \text{DAG}_{uv} \middle| u \leadsto v \right\} \\
  \]

  ![Recursive Solution Diagram]
- memorized search
  
  \[
  \text{shortest}(u) = \begin{cases} 
    \text{if } u = s & \text{return } 0 \\
    \text{if } u \text{ is "solved" } & \text{return } \text{distance}[v] \\
    r = \infty \\
    \text{for } u = 1 \text{ to } n \\
    \text{if } (u,v) \text{ is an edge, } \text{shortest}(u) + w_{uv} < r \\
    r = \text{shortest}(u) + w_{uv} \\
    \text{mark } v \text{ as solved, distance } [v] = r \\
    \text{return } r
  \end{cases}
  \]

- Example: Longest Common Subsequence (LCS)

- Input: two sequences \( a = 1, 2, 3, 2, 1 \) \( b = 2, 3, 1, 4, 1 \)

- Subsequence: subset of elements in the same order (not necessarily continuous)

  \[
  \begin{aligned}
  1, 2, 3 \checkmark & \quad \text{for } a = 1, 2, 3, 2, 1 \\
  1, 3, 2 \checkmark & \quad \text{for } a = 1, 2, 3, 2, 1 \\
  1, 2, 1 \checkmark & \quad \text{for } a = 1, 2, 3, 2, 1 \\
  1, 1, 2 \times & \quad \text{for } a = 1, 2, 3, 2, 1 \\
  \end{aligned}
  \]

- Problem: find (the length of) the longest common subsequence of \( a, b \).

  (in this case \( 2, 3, 1 \))

  (recall: look at the last step of the solution)

  \[
  \begin{aligned}
  a = 1, 2, 3, 2, 1 \quad & \text{len}(a) = n \\
  b = 2, 3, 1, 4, 1 \quad & \text{len}(b) = m \\
  \end{aligned}
  \]

- Q: Do \( a_n, b_m \) belong to the LCS?

  \[
  \begin{aligned}
  \text{case 1} & \quad \text{because } a_n = b_m \text{ it is possible both of them are in LCS} \\
  & \quad \text{this case is impossible} \\
  \text{LCS} &= \begin{cases} 
    \square \text{ (if } a_n \neq b_m) \\
    \square \square \text{ (if } a_n = b_m) 
  \end{cases} \\
  \text{LCS} &= \begin{cases} 
    \square \text{ (if } a_n \neq b_m) \\
    \square \square \text{ (if } a_n = b_m) 
  \end{cases} \\
  \end{aligned}
  \]

  \[
  \begin{aligned}
  \text{case 2} & \quad a_n \text{ is not in LCS} \\
  \text{LCS}(a, b) &= \text{LCS}(a[1\ldots n-1], b[1\ldots m]) \\
  \end{aligned}
  \]

  \[
  \begin{aligned}
  \text{case 3} & \quad b_m \text{ is not in LCS} \\
  \text{LCS}(a, b) &= \text{LCS}(a[1\ldots n-1], b[1\ldots m]) \\
  \end{aligned}
  \]
\[ \text{LCS}(a, b) = \text{LCS}(a[1..n], b[1..m-1]) \]