CSE 202: Design and Analysis of Algorithms

Lecture 5

Instructor: Kamalika Chaudhuri
Announcements

• Kamalika’s Office hours today moved to tomorrow, 12:15-1:15pm
• Homework 1 due now
• Midterm on **Feb 14**
Algorithm Design Paradigms

- **Exhaustive Search**

- **Greedy Algorithms**: Build a solution incrementally piece by piece

- **Divide and Conquer**: Divide into parts, solve each part, combine results

- **Dynamic Programming**: Divide into subtasks, perform subtask by size. Combine smaller subtasks to larger ones

- **Hill-climbing**: Start with a solution, improve it
Divide and Conquer

- Integer Multiplication
- Closest pair of points on a Plane
- Strassen’s algorithm for matrix multiplication
**Integer Multiplication: The Grade School Way**

How to multiply two $n$-bit numbers?

1. Create array of $n$ intermediate sums
2. Add up the sums

Time per addition = $O(n)$
Total time = $O(n^2)$.

Can we do better?
A Simple Divide and Conquer

**Problem:** How to multiply two n-bit numbers $x$ and $y$?

$x = x_L 2^{n/2} + x_R$

$y = y_L 2^{n/2} + y_R$
A Simple Divide and Conquer

Problem: How to multiply two n-bit numbers \( x \) and \( y \)?

\[
x = x_L 2^{n/2} + x_R
\]

\[
y = y_L 2^{n/2} + y_R
\]

\[
xy = (x_L 2^{n/2} + x_R)(y_L 2^{n/2} + y_R)
\]

\[
= x_L y_L 2^n + (x_R y_L + x_L y_R) 2^{n/2} + x_R y_R
\]
A Simple Divide and Conquer

**Problem:** How to multiply two n-bit numbers \(x\) and \(y\)?

\[
x = x_L 2^{n/2} + x_R
\]

\[
y = y_L 2^{n/2} + y_R
\]

\[
xy = (x_L 2^{n/2} + x_R)(y_L 2^{n/2} + y_R)
\]

\[
= x_L y_L 2^n + (x_R y_L + x_L y_R) 2^{n/2} + x_R y_R
\]

**Operations:**
1. 4 multiplications of \(n/2\) bit #s
2. Shifting by \(n\) bits
3. 3 additions
A Simple Divide and Conquer

**Problem:** How to multiply two n-bit numbers $x$ and $y$?

$x = \begin{array}{c}
\hline
x_L \\
\hline
x_R \\
\hline
\end{array}$

$y = \begin{array}{c}
\hline
y_L \\
\hline
y_R \\
\hline
\end{array}$

$x = x_L \ 2^{n/2} + x_R$

$y = y_L \ 2^{n/2} + y_R$

$xy = (x_L \ 2^{n/2} + x_R)(y_L \ 2^{n/2} + y_R)$

$= x_L y_L \ 2^n + (x_R y_L + x_L y_R) \ 2^{n/2} + x_R y_R$

**Operations:**
1. 4 multiplications of n/2 bit #s
2. Shifting by n bits
3. 3 additions

**Recurrence:**
$$T(n) = 4T(n/2) + O(n)$$
Problem: How to multiply two n-bit numbers x and y?

\[ x = x_L \cdot 2^{n/2} + x_R \]

\[ y = y_L \cdot 2^{n/2} + y_R \]

\[ xy = (x_L \cdot 2^{n/2} + x_R)(y_L \cdot 2^{n/2} + y_R) \]

\[ = x_Ly_L \cdot 2^n + (x_Ry_L + x_Ly_R) \cdot 2^{n/2} + x_Ry_R \]

Operations:
1. 4 multiplications of n/2 bit #s
2. Shifting by n bits
3. 3 additions

Recurrence:
\[ T(n) = 4T(n/2) + O(n) \]
\[ T(n) = O(n^2) \]

What is the base case?
A Better Divide and Conquer

**Problem:** How to multiply two n-bit numbers \( x \) and \( y \)?

\[
x = x_L 2^{n/2} + x_R \\
y = y_L 2^{n/2} + y_R
\]

\[
xy = (x_L 2^{n/2} + x_R)(y_L 2^{n/2} + y_R) \\
= x_L y_L 2^n + (x_R y_L + x_L y_R) 2^{n/2} + x_R y_R
\]

Need: \( x_L y_L, x_R y_R, (x_R y_L + x_L y_R) \)

Computed by 3 multiplications as:
\[
(x_R y_L + x_L y_R) = (x_L + x_R)(y_L + y_R) - x_L y_L - x_R y_R
\]
A Better Divide and Conquer

**Problem:** How to multiply two n-bit numbers \( x \) and \( y \)?

\[
x = x_L 2^{n/2} + x_R
\]

\[
y = y_L 2^{n/2} + y_R
\]

\[
xy = (x_L 2^{n/2} + x_R)(y_L 2^{n/2} + y_R)
\]

\[
= x_L y_L 2^n + (x_R y_L + x_L y_R) 2^{n/2} + x_R y_R
\]

Need: \( x_L y_L \), \( x_R y_R \), \( (x_R y_L + x_L y_R) \)

Computed by 3 multiplications as:

\[
(x_R y_L + x_L y_R) = (x_L + x_R)(y_L + y_R) - x_L y_L - x_R y_R
\]

**Operations:**
1. 3 multiplications of n/2 bit #s
2. Shifting by n bits
3. 6 additions
A Better Divide and Conquer

**Problem:** How to multiply two n-bit numbers \( x \) and \( y \)?

\[
x = x_L 2^{n/2} + x_R
\]
\[
y = y_L 2^{n/2} + y_R
\]

\[
xy = (x_L 2^{n/2} + x_R)(y_L 2^{n/2} + y_R)
\]

\[
= x_Ly_L 2^n + (x_Ry_L + x_Ly_R) 2^{n/2} + x_Ry_R
\]

Need: \( x_Ly_L, x_Ry_R, (x_Ry_L + x_Ly_R) \)

Computed by 3 multiplications as:

\[
(x_Ry_L + x_Ly_R) = (x_L + x_R)(y_L + y_R) - x_Ly_L - x_Ry_R
\]

**Operations:**
1. 3 multiplications of \( n/2 \) bit #s
2. Shifting by \( n \) bits
3. 6 additions

**Recurrence:**

\[
T(n) = 3T(n/2) + O(n)
\]
**Problem:** How to multiply two n-bit numbers x and y?

\[
x = \begin{array}{c}
x_L \\
\hline \\
x_R \\
\end{array} \quad y = \begin{array}{c}
y_L \\
\hline \\
y_R \\
\end{array}
\]

\[
x = x_L 2^{n/2} + x_R
\]

\[
y = y_L 2^{n/2} + y_R
\]

\[
xy = (x_L 2^{n/2} + x_R)(y_L 2^{n/2} + y_R) = x_L y_L 2^n + (x_R y_L + x_L y_R) 2^{n/2} + x_R y_R
\]

Need: \(x_L y_L, x_R y_R, (x_R y_L + x_L y_R)\)

Computed by 3 multiplications as:

\[
(x_R y_L + x_L y_R) = (x_L + x_R)(y_L + y_R) - x_L y_L - x_R y_R
\]

**Operations:**
1. 3 multiplications of n/2 bit #s
2. Shifting by n bits
3. 6 additions

**Recurrence:**

\[
T(n) = 3T(n/2) + O(n)
\]

\[
T(n) = O(n^{1.59})
\]
A Better Divide and Conquer

**Problem:** How to multiply two n-bit numbers \( x \) and \( y \)?

\[
\begin{align*}
  x &= x_L 2^{n/2} + x_R \\
  y &= y_L 2^{n/2} + y_R
\end{align*}
\]

\[
xy = (x_L 2^{n/2} + x_R)(y_L 2^{n/2} + y_R)
= x_L y_L 2^n + (x_R y_L + x_L y_R) 2^{n/2} + x_R y_R
\]

Need: \( x_L y_L, x_R y_R, (x_R y_L + x_L y_R) \)

Computed by 3 multiplications as:
\[
(x_R y_L + x_L y_R) = (x_L + x_R)(y_L + y_R) - x_L y_L - x_R y_R
\]

Best: \( O(n \log n \ 2^{\Theta(\log^* n)}) \) [Furer07]

**Operations:**
1. 3 multiplications of \( n/2 \) bit #s
2. Shifting by \( n \) bits
3. 6 additions

**Recurrence:**
\[
T(n) = 3T(n/2) + O(n)
\]
\[
T(n) = O(n^{1.59})
\]
Divide and Conquer

• Integer Multiplication
• Closest pair of points on a Plane
• Strassen’s algorithm for matrix multiplication
Closest Pair of Points on a Plane

Given a set $P$ of $n$ points on the plane, find the two points $p$ and $q$ in $P$ s.t $d(p, q)$ is minimum.
Closest Pair of Points on a Plane

Given a set $P$ of $n$ points on the plane, find the two points $p$ and $q$ in $P$ s.t. $d(p, q)$ is minimum.

Naive solution $= O(n^2)$
What about one dimension?

Given a set $P$ of $n$ points on the line, find the two points $p$ and $q$ in $P$ s.t $d(p, q)$ is minimum.
What about one dimension?

Given a set $P$ of $n$ points on the line, find the two points $p$ and $q$ in $P$ s.t $d(p, q)$ is minimum.

Property: The closest points are adjacent in sorted order.
What about one dimension?

Given a set P of n points on the line, find the two points p and q in P s.t. d(p, q) is minimum.

Property: The closest points are adjacent in sorted order.

Algorithm 1
1. Sort the points
2. Find a point $p_i$ in the sorted set s.t. $d(p_i, p_{i+1})$ is minimum.
What about one dimension?

Given a set $P$ of $n$ points on the line, find the two points $p$ and $q$ in $P$ s.t. $d(p, q)$ is minimum.

Property: The closest points are adjacent in sorted order.

Algorithm I
1. Sort the points
2. Find a point $p_i$ in the sorted set s.t. $d(p_i, p_{i+1})$ is minimum

Running Time = $O(n \log n)$
Does this work in 2D?

(a, b) : closest in x-coordinate
(a, q) : closest in y-coordinate
(p, q) : closest

Sorting the points by x or y coordinate, and looking at adjacent pairs does not work!
A Divide and Conquer Approach

Find-Closest-Pair(Q):

1. Find the median x coordinate \( l_x \)
2. \( L = \) Points to the left of \( l_x \)
3. \( R = \) Rest of the points
4. Find-closest-pair(L)
5. Find-closest-pair(R)
6. Combine the results

Problem Case:
\( p \) in \( L \), \( q \) in \( R \)
(or vice versa)

How to combine?
A Divide and Conquer Approach

**Find-Closest-Pair(Q):**

1. Find the median x coordinate $l_x$
2. L = Points to the left of $l_x$
3. R = Rest of the points
4. Find-closest-pair(L)
5. Find-closest-pair(R)
6. Combine the results

**How to combine?**

1. Naive combination
   Time = $O(n^2)$
2. Can we do better?
A Property of Closest Points

Let:
\[ t_1 = \min_{x, y \in L} d(x, y) \]
\[ t_2 = \min_{x, y \in R} d(x, y) \]
\[ t = \min (t_1, t_2) \]

If \( d(a, b) < t \), where \( a \) is in \( L \), \( b \) in \( R \), then, \( a \) and \( b \) lie within distance \( t \) of \( l_x \)
A Property of Closest Points

Let: \[ t_1 = \min_{x, y \in L} d(x, y) \]
\[ t_2 = \min_{x, y \in R} d(x, y) \]
\[ t = \min (t_1, t_2) \]

If \( d(a, b) < t \), where \( a \) is in \( L \), \( b \) in \( R \), then, \( a \) and \( b \) lie within distance \( t \) of \( l_x \).
A Property of Closest Points

Let: \[ t_1 = \min_{x, y \in L} d(x, y) \]
\[ t_2 = \min_{x, y \in R} d(x, y) \]
\[ t = \min (t_1, t_2) \]

If \( d(a, b) < t \), where \( a \) is in \( L \), \( b \) in \( R \), then, \( a \) and \( b \) lie within distance \( t \) of \( l_x \)
A Property of Closest Points

Let:
\[ t_1 = \min_{x, y \in L} d(x, y) \]
\[ t_2 = \min_{x, y \in R} d(x, y) \]
\[ t = \min (t_1, t_2) \]

If \( d(a, b) < t \), where \( a \) is in \( L \), \( b \) in \( R \), then, \( a \) and \( b \) lie within distance \( t \) of \( l_x \).

We can safely rule out all points in the grey regions!
Another Property

Let: \[ t_1 = \min_{x, y \in L} d(x, y) \]
\[ t_2 = \min_{x, y \in R} d(x, y) \]
\[ t = \min (t_1, t_2) \]
\[ S_y = \text{all points within distance } t \text{ of } l_x \text{ in sorted order of } y \text{ coords} \]

If \( d(a, b) < t \), \( a \) in \( L \), \( b \) in \( R \), then \( a \) and \( b \) occur within 15 positions of each other in \( S_y \)

**Fact:** Each box in figure can contain at most one point.

If \( d(a,b) < t \), \( a \) in \( L \), \( b \) in \( R \), \( a, b \) are > 15 positions apart then, there are \( \geq 3 \) rows between \( a \) and \( b \)

Thus, \( d(a, b) \geq 3t/2 \). **Contradiction!**
A Divide and Conquer Approach

**Find-Closest-Pair(Q):**

1. Find the median x coordinate \( l_x \)
2. \( L = \) Points to the left of \( l_x \)
3. \( R = \) Rest of the points
4. \((a_1, b_1, t_1) = \text{Find-closest-pair}(L)\)
5. \((a_2, b_2, t_2) = \text{Find-closest-pair}(R)\)
6. \( t = \min(t_1, t_2) \)
7. Combine the results

**Property:** If \( a \) in \( L \), \( b \) in \( R \), and if \( d(a, b) < t \), then \( a \) and \( b \) occur within 15 positions of each other in \( S_y \)

**How to combine?**

- \( S = \) points in \( Q \) within distance \( t \) of \( l_x \)
- \( S_y = \) sort \( S \) by \( y \) coords
- For \( p \) in \( S_y \)
  - Find distance from \( p \) to next 15 points
  - Let \((a, b)\) be the pair with min such distance
  - If \( d(a, b) < t \):
    - Return \((a, b, d(a, b))\)
  - Else if \( t_1 < t_2 \):
    - Return \((a_1, b_1, t_1)\)
  - Else:
    - Return \((a_2, b_2, t_2)\)
A Divide and Conquer Approach

Property: If a in L, b in R, and if \( d(a, b) < t \), then a and b occur within 15 positions of each other in \( S_y \)

Find-Closest-Pair\((Q_x, Q_y)\):
1. Find the median x coordinate \( l_x \)
2. \( L = \) Points to the left of \( l_x \)
3. \( R = \) Rest of the points
4. \((a_1, b_1, t_1) = \) Find-closest-pair\((L_x, L_y)\)
5. \((a_2, b_2, t_2) = \) Find-closest-pair\((R_x, R_y)\)
6. \( t = \min(t_1, t_2) \)
7. Combine the results

Implementing Divide + Combine:
1. Maintain sorted lists of the input points:
   \( P_x = \) sorted by x, \( P_y = \) sorted by y
2. Computing \( L_x, L_y, R_x, R_y \) from \( Q_x, Q_y \): \( O(|Q|) \) time
3. Computing \( S_y \) from \( Q_y \): \( O(|S|) \) time
4. Combination procedure: \( O(|S_y|) \) time

How to combine?

\( S = \) points in \( Q \) within distance \( t \) of \( l_x \)
\( S_y = \) sort \( S \) by y coords
For \( p \) in \( S_y \)
   Find distance from \( p \) to next 15 points
   Let \( (a, b) \) be the pair with min such distance
   If \( d(a, b) < t \):
      Return \( (a, b, d(a, b)) \)
   Else if \( t_1 < t_2 \):
      Return \( (a_1, b_1, t_1) \)
   Else:
      Return \( (a_2, b_2, t_2) \)
A Divide and Conquer Approach

**Property:** If \( a \) in \( L \), \( b \) in \( R \), and if \( d(a, b) < t \), then \( a \) and \( b \) occur within 15 positions of each other in \( S_y \)

**Find-Closest-Pair**(\( Q_x, Q_y \)):

1. Find the median \( x \) coordinate \( l_x \)
2. \( L \) = Points to the left of \( l_x \)
3. \( R \) = Rest of the points
4. \((a_1, b_1, t_1) = \text{Find-closest-pair}(L_x, L_y)\)
5. \((a_2, b_2, t_2) = \text{Find-closest-pair}(R_x, R_y)\)
6. \( t = \min(t_1, t_2) \)
7. Combine the results

**How to combine?**

\( S = \) points in \( Q \)
\( S_y = \) sort \( S \) by \( y \) coords

\( S_y = \) sort \( S \) by \( y \) coords

- For \( p \) in \( S_y \)
  - Find distance from \( p \) to next 15 points
  - Let \((a, b)\) be the pair with min such distance

- If \( d(a, b) < t \):
  - Return \((a, b, d(a, b))\)
- Else if \( t_1 < t_2 \):
  - Return \((a_1, b_1, t_1)\)
- Else:
  - Return \((a_2, b_2, t_2)\)

**Implementing Divide + Combine:**

\[
T(n) = 2 T(n/2) + cn \\
T(n) = \Theta(n \log n)
\]
Summary: Closest Pair of Points

Given a set $P$ of $n$ points on the plane, find the two points $p$ and $q$ in $P$ s.t $d(p, q)$ is minimum.

**Find-Closest-Pair($Q_x, Q_y$):**

1. Find the median $x$ coordinate $l_x$
2. $L =$ Points to the left of $l_x$
3. $R =$ Rest of the points
4. $(a_1, b_1, t_1) =$ Find-closest-pair($L_x, L_y$)
5. $(a_2, b_2, t_2) =$ Find-closest-pair($R_x, R_y$)
6. $t = \min(t_1, t_2)$
7. Let $S_y =$ points within distance $t$ of $l_x$ sorted by $y$
8. For each $p$ in $S_y$
   - Find distance from $p$ to next 15 points in $S_y$
   - Let $(a, b)$ be the closest such pair
9. Report the closest pair out of:
   - $(a, b), (a_1, b_1), (a_2, b_2)$

**Recurrence:**

$T(n) = 2T(n/2) + cn$

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

What is a base case?
Divide and Conquer

• Integer Multiplication
• Closest pair of points on a Plane
• Strassen’s algorithm for matrix multiplication
Matrix Multiplication

**Problem:** Given two $n \times n$ matrices $X$ and $Y$, compute $Z = XY$

$$Z_{ij} = \sum X_{ik} Y_{kj}$$

**Naive Method:** $O(n^3)$ time
A Simple Divide and Conquer

**Problem:** Given two $n \times n$ matrices $X$ and $Y$, compute $Z = XY$

$$X = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \quad Y = \begin{bmatrix} E & F \\ G & H \end{bmatrix}$$

$$Z = \begin{bmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{bmatrix}$$

A..H: $n/2 \times n/2$

**Algorithm 1:**
1. Compute $AE$, $BG$, $CE$, $DG$, $AF$, $BH$, $CF$, $DH$
2. Compute $AE + BG$, $CE + DG$, $AF + BH$, $CF + DH$

**Operations:** 8 $n/2 \times n/2$ matrix multiplications, 4 additions

**Recurrence:** $T(n) = 8 T(n/2) + O(n^2) \quad T(n) = O(n^3)$
Strassen's Algorithm

Problem: Given two $n \times n$ matrices $X$ and $Y$, compute $Z = XY$

$$X = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \quad Y = \begin{bmatrix} E & F \\ G & H \end{bmatrix}$$

$$Z = \begin{bmatrix} P_5 + P_4 - P_2 + P_6 & P_1 + P_2 \\ P_3 + P_4 & P_1 + P_5 - P_3 - P_7 \end{bmatrix}$$

$A..H: n/2 \times n/2$

$P_1 = A(F - H) \quad P_3 = (C + D)E \quad P_5 = (A + D)(E + H) \quad P_7 = (A - C)(E + F)$

$P_2 = (A + B)H \quad P_4 = D(G - E) \quad P_6 = (B - D)(G + H)$

Operations: $7 \frac{n}{2} \times \frac{n}{2}$ matrix multiplications, $O(1)$ additions

Recurrence: $T(n) = 7T(\frac{n}{2}) + O(n^2) \quad T(n) = O(n^{2.81})$
Strassen’s Algorithm

Problem: Given two \( n \times n \) matrices \( X \) and \( Y \), compute \( Z = XY \)

\[
X = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \quad Y = \begin{bmatrix} E & F \\ G & H \end{bmatrix}
\]

\[
Z = \begin{bmatrix} P_5 + P_4 - P_2 + P_6 & P_1 + P_2 \\ P_3 + P_4 & P_1 + P_5 - P_3 - P_7 \end{bmatrix}
\]

\( A..H: n/2 \times n/2 \)

\[
P_1 = A(F - H) \quad P_3 = (C + D)E \quad P_5 = (A + D)(E + H) \quad P_7 = (A - C)(E + F)
\]

\[
P_2 = (A + B)H \quad P_4 = D(G - E) \quad P_6 = (B - D)(G + H)
\]

Operations: 7 \( n/2 \times n/2 \) matrix multiplications, \( O(1) \) additions

Recurrence: \( T(n) = 7T(n/2) + O(n^2) \) \( T(n) = O(n^{2.81}) \)
Algorithm Design Paradigms

- **Exhaustive Search**

- **Greedy Algorithms:** Build a solution incrementally piece by piece

- **Divide and Conquer:** Divide into parts, solve each part, combine results

- **Dynamic Programming:** Divide into subtasks, perform subtask by size. Combine smaller subtasks to larger ones

- **Hill-climbing:** Start with a solution, improve it
Dynamic Programming (DP): A Simple Example

**Problem:** Compute the n-th Fibonacci number

**Recursive Solution**

```python
function Fib1(n)
    if n = 1 return 1
    if n = 2 return 1
    return Fib1(n-1) + Fib1(n-2)
```

**Running Time:**

\[ T(n) = T(n-1) + T(n-2) + 1 \]
Dynamic Programming (DP): A Simple Example

**Problem:** Compute the n-th Fibonacci number

**Recursive Solution**

```python
function Fib1(n)
    if n = 1 return 1
    if n = 2 return 1
    return Fib1(n-1) + Fib1(n-2)
```

**Running time:** $O(c^n)$

**Running Time:**

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

$T(n) = O(c^n)$
Dynamic Programming (DP): A Simple Example

**Problem:** Compute the \( n \)-th Fibonacci number

**Recursive Solution**

```
function Fib1(n)
if n = 1 return 1
if n = 2 return 1
return Fib1(n-1) + Fib1(n-2)
```

**Running time:** \( O(c^n) \)

**Running Time:**

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

\[
T(n) = O(c^n)
\]

**Dynamic Programming Solution**

```
function Fib2(n)
Create an array fib[1..n]
fib[1] = 1
fib[2] = 1
for i = 3 to n:
    fib[i] = fib[i-1] + fib[i-2]
return fib[n]
```
Dynamic Programming (DP): A Simple Example

**Problem:** Compute the n-th Fibonacci number

**Recursive Solution**

```
function Fib1(n)
if n = 1 return 1
if n = 2 return 1
return Fib1(n-1) + Fib1(n-2)
```

**Running time:** \(O(c^n)\)

**Dynamic Programming Solution**

```
function Fib2(n)
Create an array fib[1..n]
fib[1] = 1
fib[2] = 1
for i = 3 to n:
    fib[i] = fib[i-1] + fib[i-2]
return fib[n]
```

**Running time:** \(O(n)\)

**Running Time:**

- Recursive Solution: \(T(n) = T(n-1) + T(n-2) + 1\)
  \(T(n) = O(c^n)\)
- Dynamic Programming Solution: \(T(n) = O(n)\)
Why does DP do better?

**Problem:** Compute the n-th Fibonacci number

**Recursive Solution**

```python
function Fib1(n)
  if n = 1 return 1
  if n = 2 return 1
  return Fib1(n-1) + Fib1(n-2)
```

**Running time:** $O(c^n)$

**Dynamic Programming Solution**

```python
function Fib2(n)
  Create an array fib[1..n]
  fib[1] = 1
  fib[2] = 1
  for i = 3 to n:
    fib[i] = fib[i-1] + fib[i-2]
  return fib[n]
```

**Running time:** $O(n)$
Dynamic Programming

Main Steps:

1. Divide the problem into **subtasks**

2. Define the subtasks **recursively** (express larger subtasks in terms of smaller ones)

3. Find the **right order** for solving the subtasks (but do not solve them recursively!)
Dynamic Programming

Main Steps:

1. Divide the problem into subtasks: compute \( \text{fib}[i] \)

2. Define the subtasks recursively (express larger subtasks in terms of smaller ones)

3. Find the right order for solving the subtasks (\( i = 1, \ldots, n \))

Dynamic Programming Solution

function \( \text{Fib2}(n) \)
Create an array \( \text{fib}[1..n] \)
\( \text{fib}[1] = 1 \)
\( \text{fib}[2] = 1 \)
for \( i = 3 \) to \( n \):
  \( \text{fib}[i] = \text{fib}[i-1] + \text{fib}[i-2] \)
return \( \text{fib}[n] \)

Running time: \( O(n) \)
Dynamic Programming

- String Reconstruction
- ...
