🎯 ALGOKOM Interactive Demo

📊 Fibonacci Calculator - Compare 4 Methods

Enter n (0-45):

20

Calculate & Compare Show Sequence 📚 View Detailed Explanations

Results & Performance Comparison:

Recursive:	6765	0.3000 ms	(avg 1x)
DP Bottom-Up:	6765	0.7000 µs	(avg 1000x)
Space Optimized:	6765	0.002000 ms	(avg 1000x)
Matrix Exponentiation:	6765	0.001300 ms	(avg 1000x)

Fibonacci Sequence [0 to 20]:
0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181, 6765

> 💡 **Tip:** Try different values! Notice how recursive method becomes exponentially slower after n=30. **Warning:** For n > 40, recursive method may take several minutes to complete! Compare the execution times!

📚 Algorithm Explanations

1. Recursive (Naive)

Cara Kerja: Memanggil diri sendiri 2x untuk setiap n. F(n) = F(n-1) + F(n-2)
Kompleksitas: O(2^n) waktu, O(n) space
Kelebihan: Mudah dipahami, sesuai definisi matematika
Kelemahan: Sangat lambat! Menghitung nilai sama berkali-kali

2. Dynamic Programming (Bottom-Up)

Cara Kerja: Menyimpan semua nilai dari 0 sampai n dalam array
Kompleksitas: O(n) waktu, O(n) space
Kelebihan: Cepat, bisa akses semua nilai kapan saja
Kelemahan: Butuh memory untuk simpan semua nilai

3. Space Optimized (Iterative)

Cara Kerja: Hanya menyimpan 2 nilai terakhir (prev & curr)
Kompleksitas: O(n) waktu, O(1) space
Kelebihan: Paling hemat memory! Tetap cepat
Kelemahan: Hanya bisa dapat nilai akhir

4. Matrix Exponentiation

Cara Kerja: Menggunakan perkalian matrix [[1,1],[1,0]]^n dengan fast power
Kompleksitas: O(log n) waktu, O(log n) space
Kelebihan: TERCEPAT untuk n sangat besar! Logaritmik
Kelemahan: Kompleks, butuh pemahaman aljabar linear

🎨 Bilinear Interpolation - Visual Demo

📖 Apa itu Bilinear Interpolation?

Bilinear Interpolation adalah teknik matematika untuk mengestimasi nilai pada titik mana pun di dalam area 2D berdasarkan nilai 4 titik corner di sekitarnya. Teknik ini digunakan secara luas dalam **image processing**, terutama untuk **resizing gambar** atau **zoom**.

🎯 **Prinsip Kerja:** Interpolasi dilakukan dalam 2 arah (x dan y), memberikan hasil yang **smooth dan natural** dibandingkan metode sederhana seperti Nearest Neighbor.

Corner Values (Q11, Q12, Q21, Q22):

Q11 (Bottom-Left):
0

Q12 (Top-Left):
100

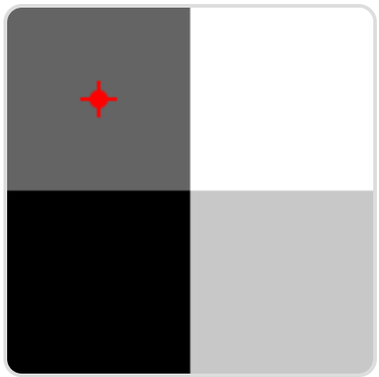
Q21 (Bottom-Right):
200

Q22 (Top-Right):
255

📚 View Detailed Explanation

Nearest Neighbor

Bilinear Interpolation



Test Point Coordinates:

X Position (0-200):

50

Y Position (0-200):

50

Test Point Calculation:

Test Point (pixel): x=50, y=50

Test Point (normalized): x=0.25, y=0.25

Rumus Bilinear Interpolation:

$$f(x,y) = (1-x)(1-y)Q_{11} + x(1-y)Q_{21} + (1-x)yQ_{12} + xyQ_{22}$$

Substitusi nilai:
f(0.25, 0.75) = (0.75)(0.25)×0 + (0.25)(0.25)×200 + (0.75)(0.75)×100 + (0.25)(0.75)×255

Step-by-step:
- Term 1 (Q_{11}): 0.1875 × 0 = **0.00**
- Term 2 (Q_{21}): 0.0625 × 200 = **12.50**

- Term 3 (Q_{12}): 0.5625 × 100 = **56.25**
- Term 4 (Q_{22}): 0.1875 × 255 = **47.81**

✅ **TOTAL = 0.00 + 12.50 + 56.25 + 47.81 = 116.56**

Interpolated Value: 116.56

💡 **Penjelasan:**

- Test point adalah titik yang kita hitung nilainya
- Nilai dihitung dari **weighted average 4 corner (Q11, Q12, Q21, Q22)**
- Semakin dekat ke corner, semakin **besar weight-nya**
- **Coba ubah Q11-Q22** dan lihat nilai test point berubah!
- **Coba geser test point** ke corner dan lihat nilainya mendekati corner tersebut!

💡 **Observation:** Move the sliders to see how bilinear interpolation creates smooth gradients compared to nearest neighbor's blocky appearance.

📚 Algorithm Explanations

Nearest Neighbor

Cara Kerja: Pilih nilai pixel terdekat tanpa interpolasi
Kompleksitas: O(1) per pixel
Kelebihan: Sangat cepat, simple
Kelemahan: Hasil kotak-kotak (pixelated), kualitas rendah
Use Case: Icon kecil, pixel art, zoom integer

Bilinear Interpolation

Cara Kerja: Interpolasi linear di 2 arah (x dan y) dari 4 pixel tetangga
Rumus: f(x,y) = (1-x)(1-y)Q11 + x(1-y)Q21 + (1-x)yQ12 + xyQ22
Kompleksitas: O(1) per pixel
Kelebihan: Smooth, natural, balance speed vs quality

Kelemahan: Sedikit blur untuk edge
Use Case: Image resizing, zoom foto, thumbnails

💡 Perbandingan

- Nearest Neighbor = Ambil 1 pixel → Hasil blocky
- Bilinear = Rata-rata weighted dari 4 pixel → Hasil smooth

⚖️ Algorithm Comparison Summary

Fibonacci

Best for Speed: Matrix Exponentiation
Best for Memory: Space Optimized
Best for Learning: DP Memoization
Range: O(1) to O(2^n)

Bilinear

Complexity: O(1) per pixel
Quality: Good balance
Use Case: Image resizing
Alternative: Bicubic (slower, smoother)