



Algoritma dan Pemrograman

#9 Meeting

Divide, Conquer, Combine

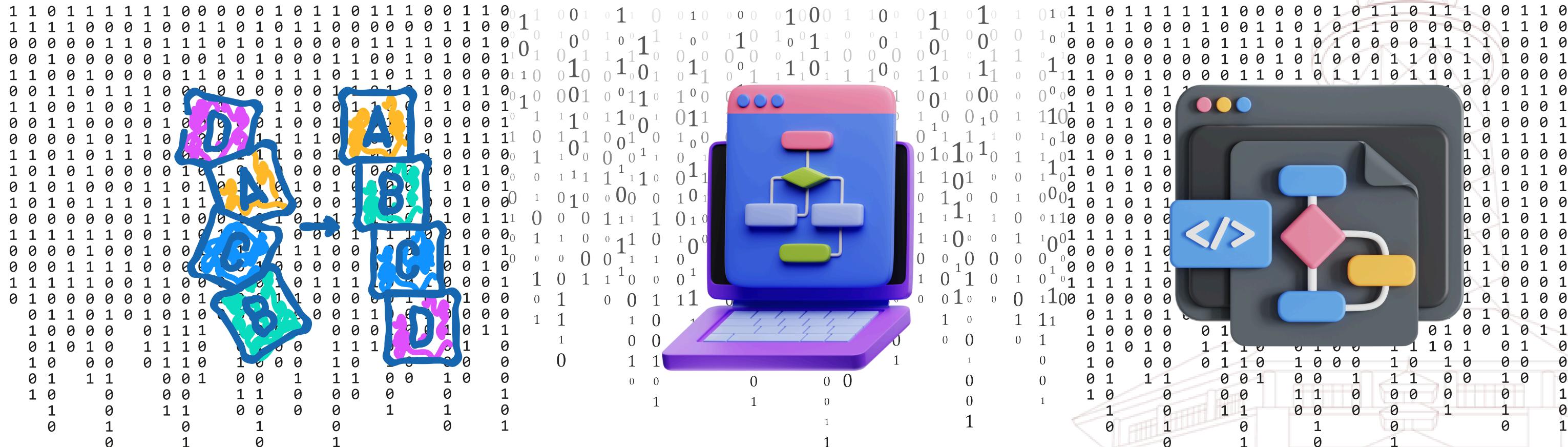
Ferdian Bangkit Wijaya, S.Stat., M.Si
NIP. 199005202024061001





#9 Meeting

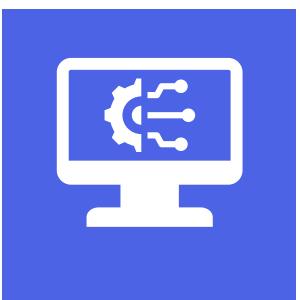
Metode Divide, Conquer, Combine





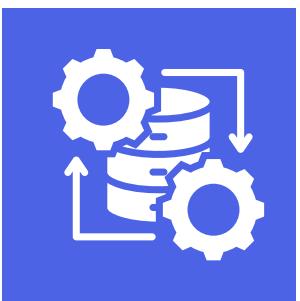
Divide, Conquer, Combine

Definisi



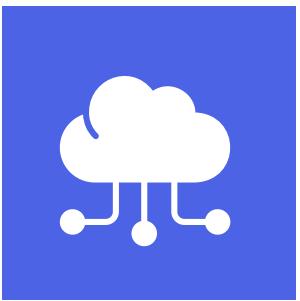
DIVIDE

Memecah masalah menjadi sub-masalah yang lebih kecil dan lebih mudah dipecahkan.



CONQUER

Menyelesaikan sub-masalah tersebut, yang biasanya dilakukan secara rekursif.



COMBINE

Menggabungkan solusi dari sub-masalah untuk mendapatkan solusi dari masalah awal.

BRANCHING FACTOR



BALANCE



Data Dependence of
Divide Function



Control Parallelism
atau Sequentially



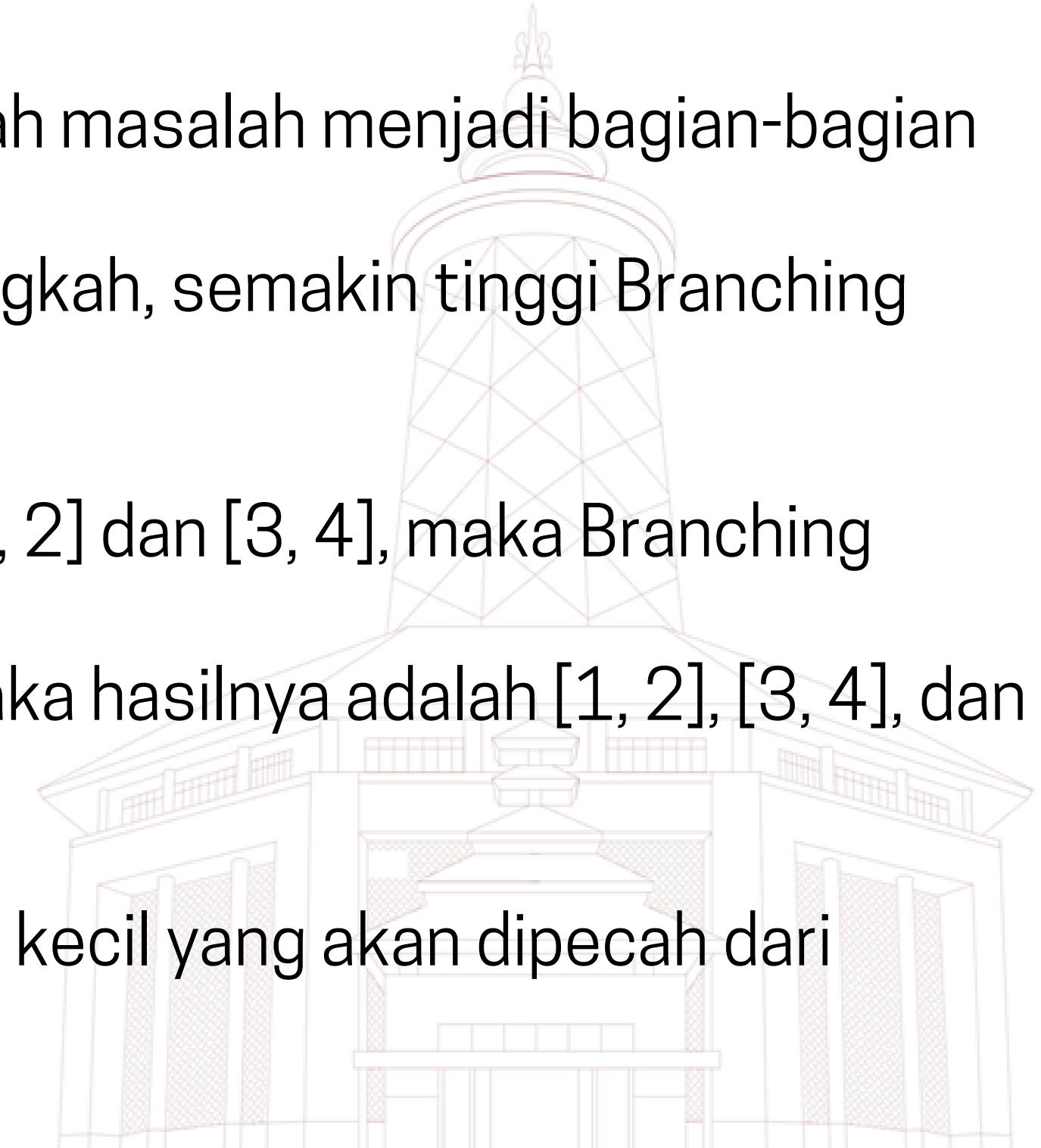
Branching Factor

Pengertian dan ilustrasi

- Pengertian:
adalah jumlah sub-bagian yang dihasilkan saat memecah masalah menjadi bagian-bagian yang lebih kecil pada langkah Divide.
Semakin banyak bagian yang dihasilkan pada setiap langkah, semakin tinggi Branching Factor-nya.

- Contoh Sederhana:
Jika array [1, 2, 3, 4] dibagi menjadi dua bagian, yaitu [1, 2] dan [3, 4], maka Branching Factor-nya adalah 2.
Jika array [1, 2, 3, 4, 5, 6] dibagi menjadi tiga bagian, maka hasilnya adalah [1, 2], [3, 4], dan [5, 6], dengan Branching Factor-nya adalah 3.

- Intinya:
Branching Factor menentukan seberapa banyak bagian kecil yang akan dipecah dari masalah besar di setiap langkah.





Balance

Pengertian dan ilustrasi

- Pengertian:

Balance berarti memecah masalah menjadi bagian-bagian yang seimbang (dalam hal jumlah elemen atau kompleksitasnya) pada langkah Divide.

Pembagian yang seimbang membantu memastikan bahwa proses di setiap sub-bagian memiliki beban yang merata, sehingga meningkatkan efisiensi.

Ketika array ganjil dibagi secara balance, perbedaan jumlah elemen antara bagian-bagian yang dihasilkan tidak boleh lebih dari 1 elemen.

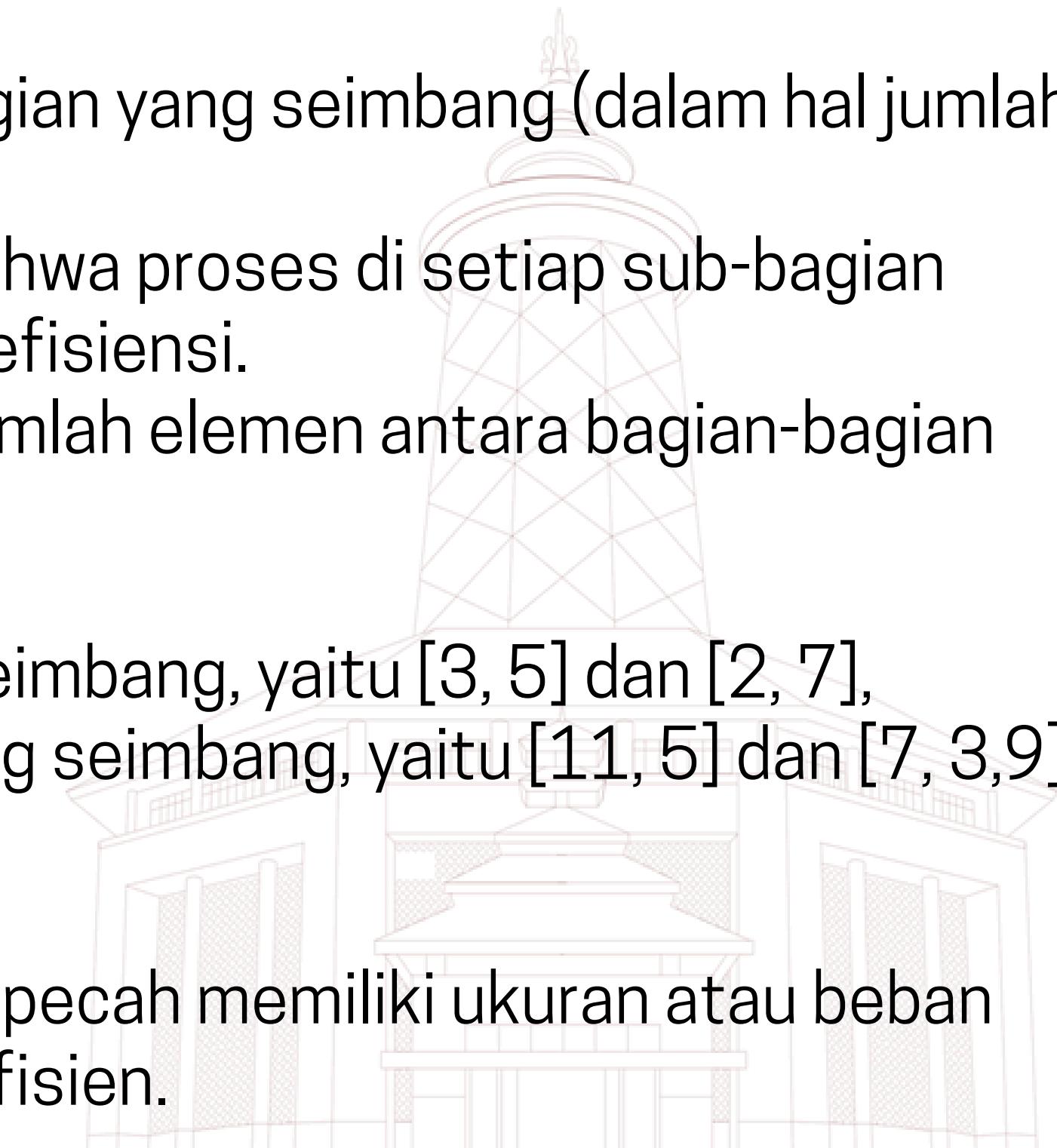
- Contoh Sederhana:

Jika array [3, 5, 2, 7] dibagi menjadi dua bagian yang seimbang, yaitu [3, 5] dan [2, 7],

Jika array [11, 5, 7, 3, 9] dibagi menjadi dua bagian yang seimbang, yaitu [11, 5] dan [7, 3, 9],

- Intinya:

Balance memastikan bahwa setiap sub-bagian yang dipecah memiliki ukuran atau beban yang sama sehingga proses pemecahannya adil dan efisien.





Data Dependence of Divide Function

Pengertian dan ilustrasi

- Pengertian:

Pada konsep ini, cara membagi masalah (pada langkah Divide) bergantung pada nilai-nilai data yang ada.

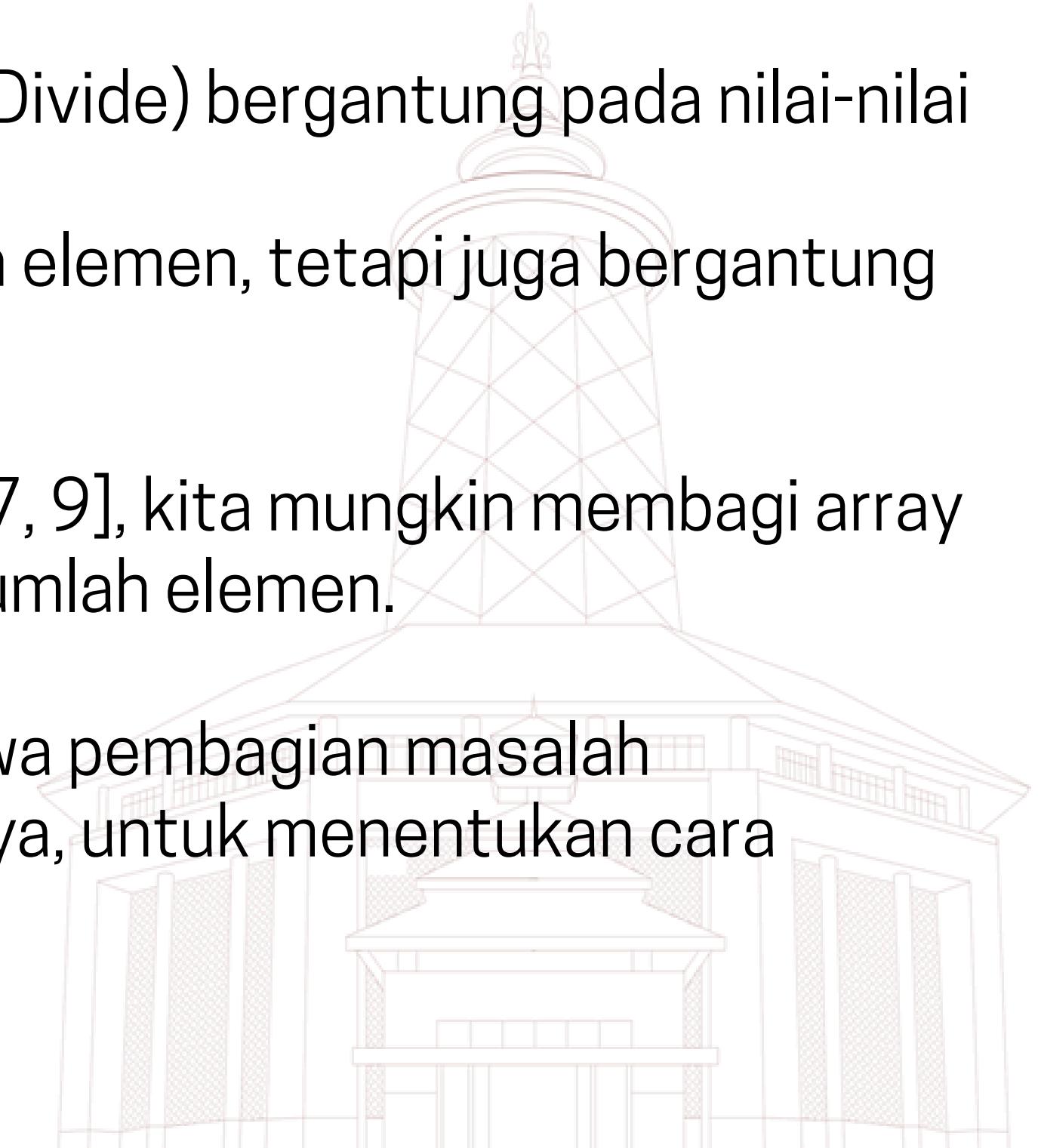
Artinya, pembagian tidak hanya didasarkan pada jumlah elemen, tetapi juga bergantung pada nilai atau sifat data tersebut.

- Contoh Sederhana:

Jika kita mencari elemen tertentu dalam array [1, 3, 5, 7, 9], kita mungkin membagi array berdasarkan nilai tengah 5, bukan hanya berdasarkan jumlah elemen.

- Intinya:

Data Dependence of Divide Function memastikan bahwa pembagian masalah mempertimbangkan nilai elemen, bukan hanya jumlahnya, untuk menentukan cara pemecahan terbaik.

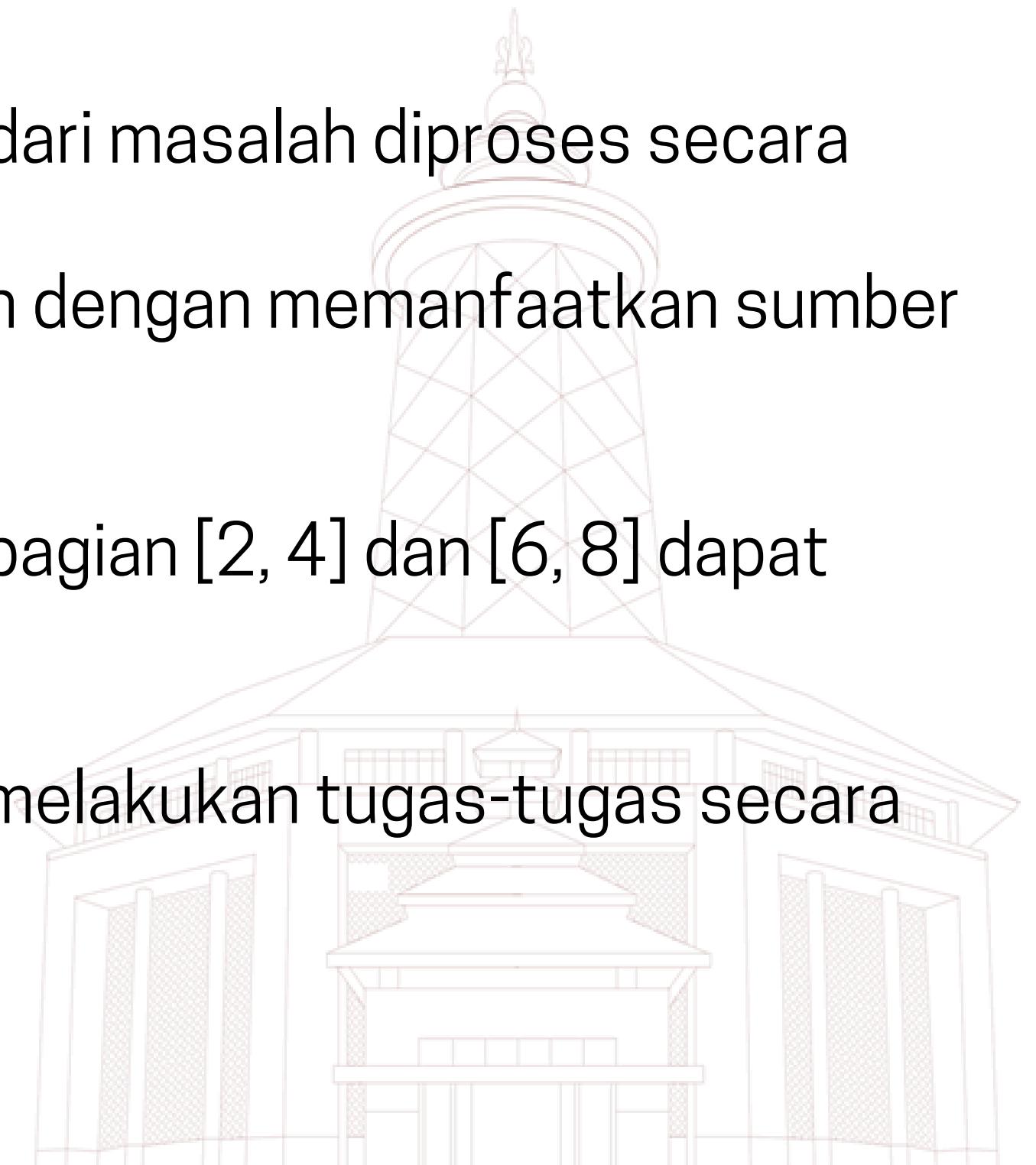




Control Parallelism atau Sequentially

Control Parallelism

- Pengertian:
adalah pendekatan yang memungkinkan bagian-bagian dari masalah diproses secara bersamaan (paralel) setelah dibagi.
Tujuannya adalah meningkatkan kecepatan pemrosesan dengan memanfaatkan sumber daya komputasi secara bersamaan.
- Contoh Sederhana:
Misalnya, saat menghitung jumlah array [2, 4, 6, 8], dua bagian [2, 4] dan [6, 8] dapat dijumlahkan secara bersamaan.
- Intinya:
Control Parallelism mempercepat pemrosesan dengan melakukan tugas-tugas secara bersamaan.





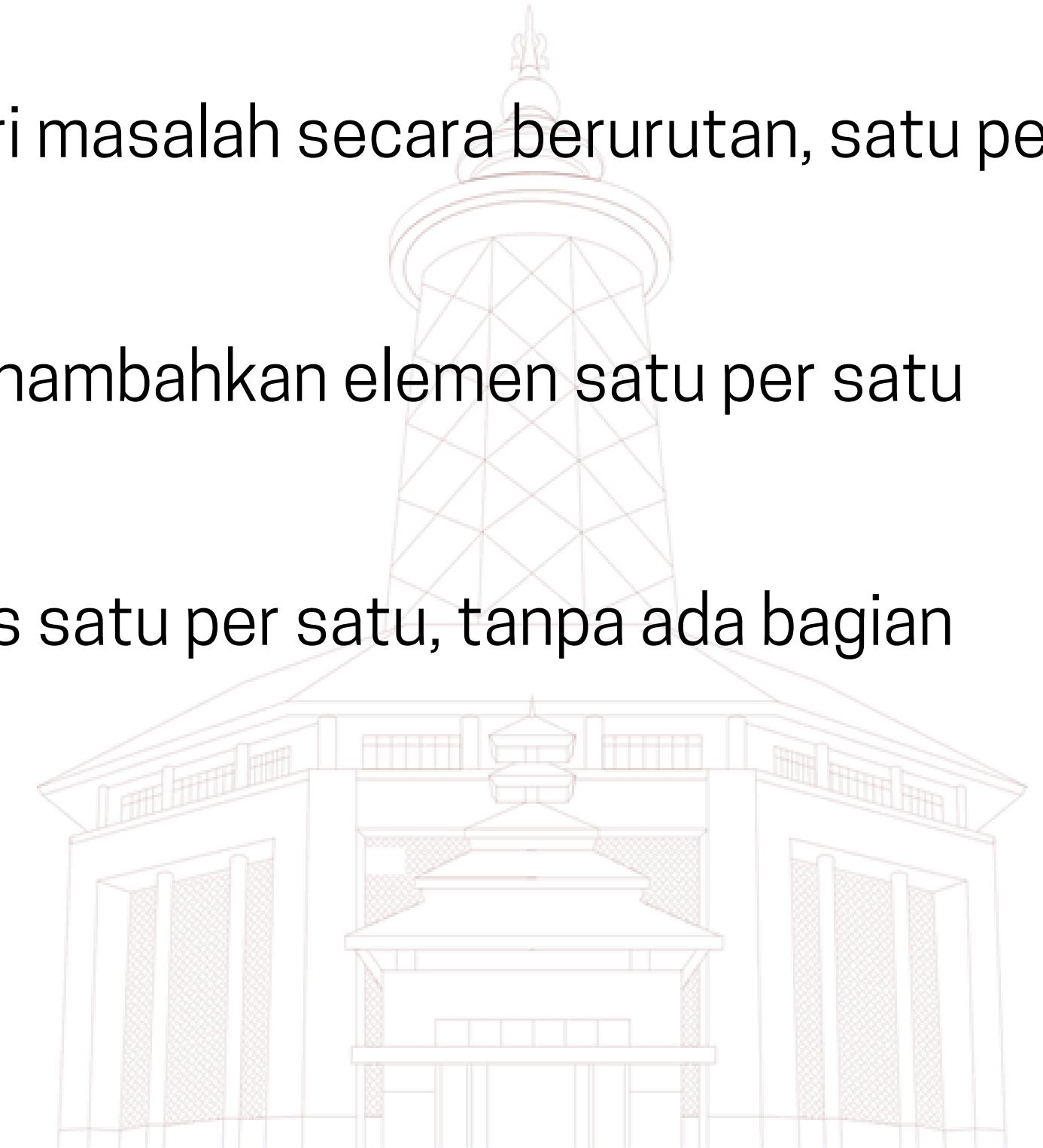
Control Parallelism atau Sequentially

Sequentially

- Pengertian:
adalah pendekatan yang memproses bagian-bagian dari masalah secara berurutan, satu per satu, setelah dibagi.

- Contoh Sederhana:
Menghitung jumlah elemen array [2, 4, 6, 8] dengan menambahkan elemen satu per satu dalam urutan.

- Intinya:
Sequentially memastikan bahwa setiap bagian diproses satu per satu, tanpa ada bagian yang diproses bersamaan.





Ilustrasi

Branching Factor

Kasus 1: Menghitung Jumlah (Sum) dengan Branching Factor

Array 1: [3, 5, 2, 7, 1]

Array 2: [4, 8, 6, 2, 10, 12]





Ilustrasi

Branching Factor

Array 1: [3, 5, 2, 7, 1]

- Langkah 1 (Branching):

Pecah array menjadi dua bagian:

Bagian 1: [3, 5]

Bagian 2: [2, 7, 1]

- Langkah 2 (Branching):

Pecah lagi bagian 2:

Bagian 1a: [3]

Bagian 1b: [5]

Bagian 2a: [2]

Bagian 2b: [7, 1]

- Langkah 3 (Conquer):

Hitung jumlah di setiap bagian:

Bagian 1a: 3

Bagian 1b: 5

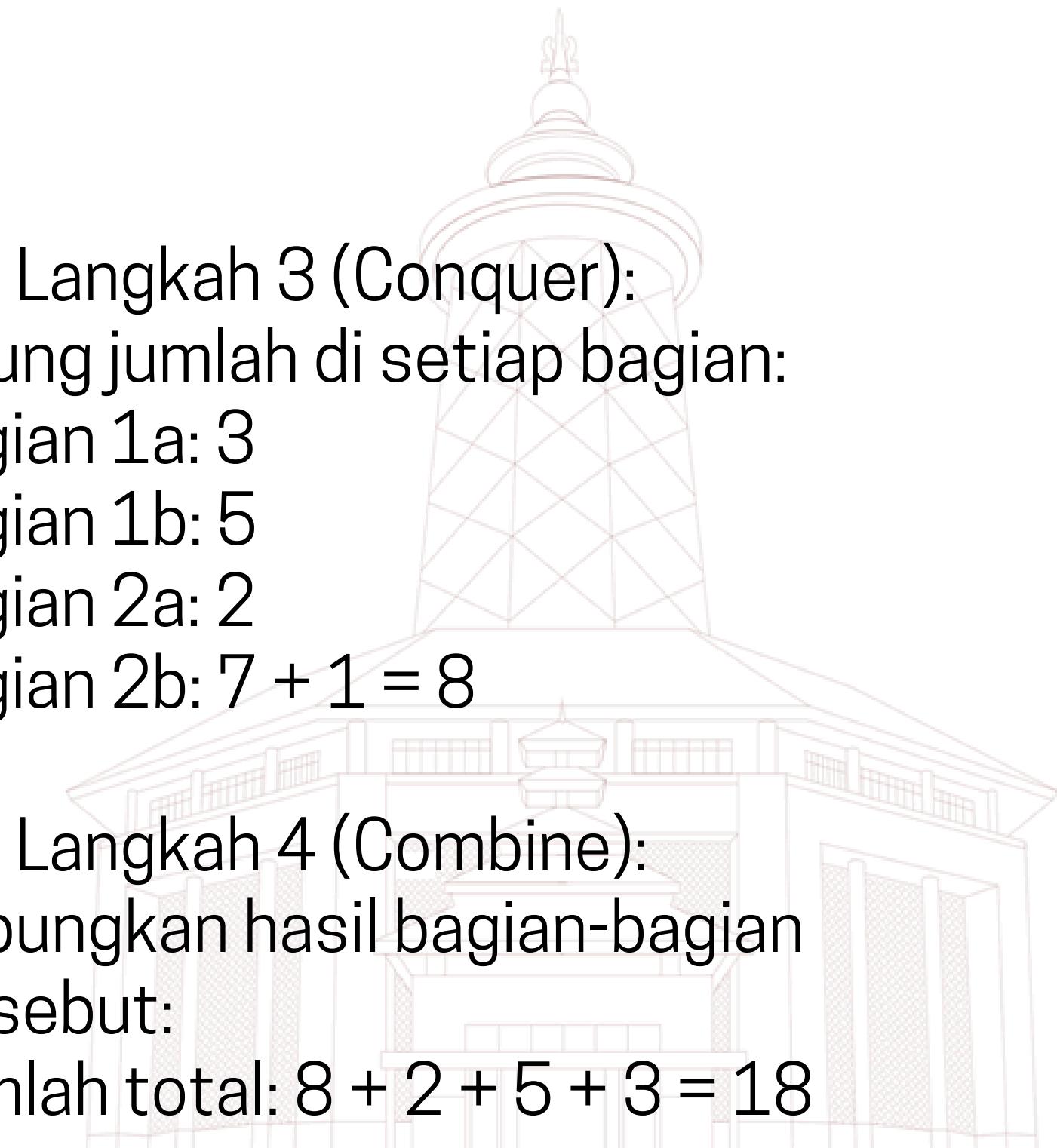
Bagian 2a: 2

Bagian 2b: $7 + 1 = 8$

- Langkah 4 (Combine):

Gabungkan hasil bagian-bagian tersebut:

Jumlah total: $8 + 2 + 5 + 3 = 18$





Ilustrasi

Branching Factor

Array 2: [4, 8, 6, 2, 10, 12]

- Langkah 1 (Branching):

Pecah array menjadi dua bagian:

Bagian 1: [4, 8, 6]

Bagian 2: [2, 10, 12]

- Langkah 2 (Branching):

Pecah lagi setiap bagian:

Bagian 1a: [4]

Bagian 1b: [8, 6]

Bagian 2a: [2]

Bagian 2b: [10, 12]

- Langkah 3 (Conquer):

Hitung jumlah di setiap bagian:

Bagian 1a: 4

Bagian 1b: $8 + 6 = 14$

Bagian 2a: 2

Bagian 2b: $10 + 12 = 22$

- Langkah 4 (Combine):

Gabungkan hasil bagian-bagian tersebut:

Jumlah total: $22 + 2 + 14 + 4 = 42$



Ilustrasi

Branching Factor

Kasus 2: Mencari Elemen Maksimal dengan Branching Factor

Array 1: [5, 3, 9, 1, 7]

Array 2: [8, 2, 10, 6, 12, 4]





Ilustrasi

Branching Factor

Array 1: [5, 3, 9, 1, 7]

- Langkah 1 (Branching):

Pecah array menjadi dua bagian:

Bagian 1: [5, 3]

Bagian 2: [9, 1, 7]

- Langkah 2 (Branching):

Pecah lagi bagian 2:

Bagian 1a: [5]

Bagian 1b: [3]

Bagian 2a: [9]

Bagian 2b: [1, 7]

- Langkah 3 (Conquer):

Temukan elemen terbesar di setiap bagian:

Bagian 1a: 5

Bagian 1b: 3

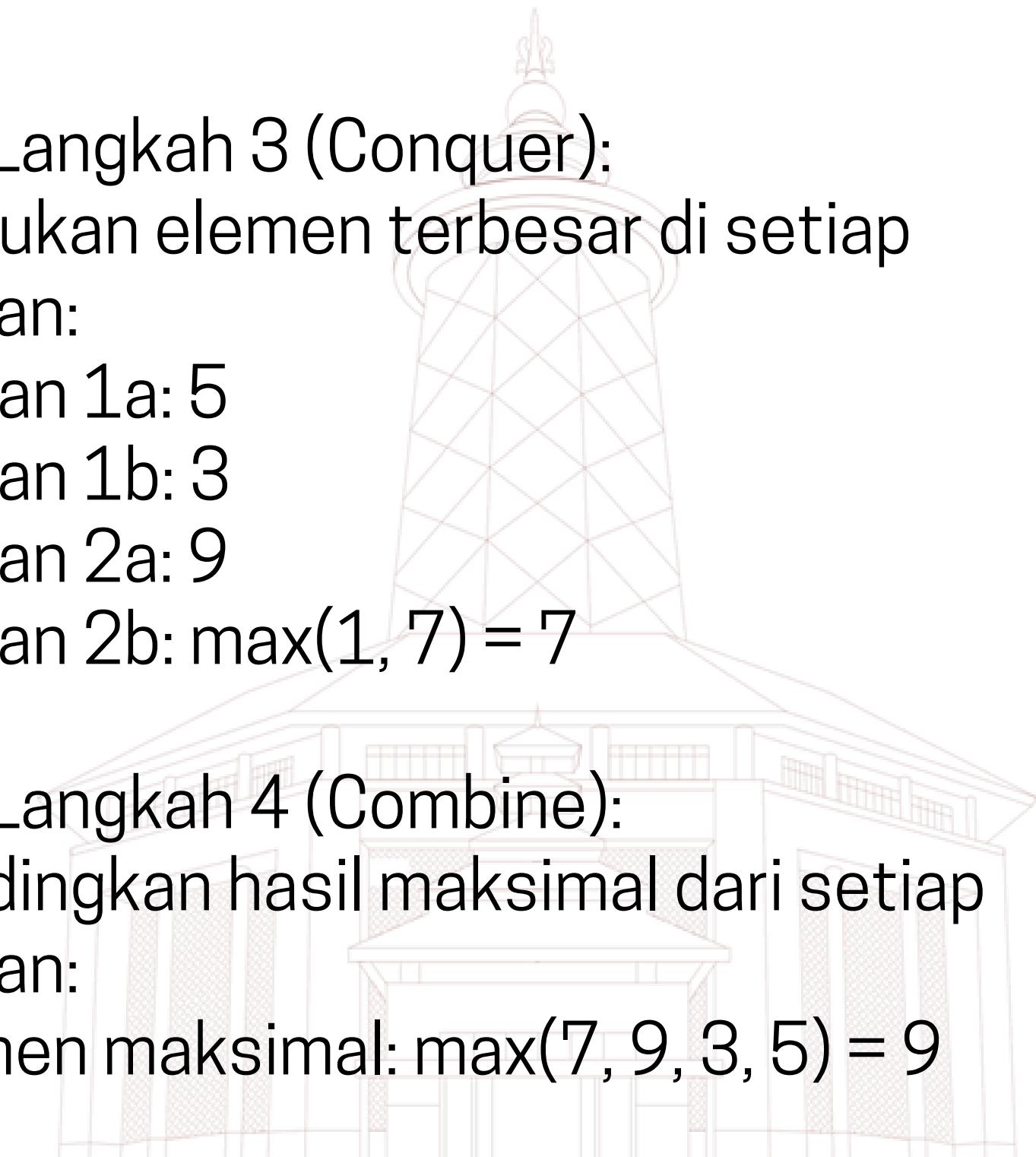
Bagian 2a: 9

Bagian 2b: $\max(1, 7) = 7$

- Langkah 4 (Combine):

Bandingkan hasil maksimal dari setiap bagian:

Elemen maksimal: $\max(7, 9, 3, 5) = 9$





Ilustrasi

Branching Factor

Array 2: [8, 2, 10, 6, 12, 4]

- Langkah 1 (Branching):

Pecah array menjadi dua bagian:

Bagian 1: [8, 2, 10]

Bagian 2: [6, 12, 4]

- Langkah 2 (Branching):

Pecah lagi setiap bagian:

Bagian 1a: [8]

Bagian 1b: [2, 10]

Bagian 2a: [6]

Bagian 2b: [12, 4]

- Langkah 3 (Conquer):

Temukan elemen terbesar di setiap bagian:

Bagian 1a: 8

Bagian 1b: $\max(2, 10) = 10$

Bagian 2a: 6

Bagian 2b: $\max(12, 4) = 12$

- Langkah 4 (Combine):

Bandingkan hasil maksimal dari setiap bagian:

Elemen maksimal: $\max(12, 6, 10, 8) = 12$



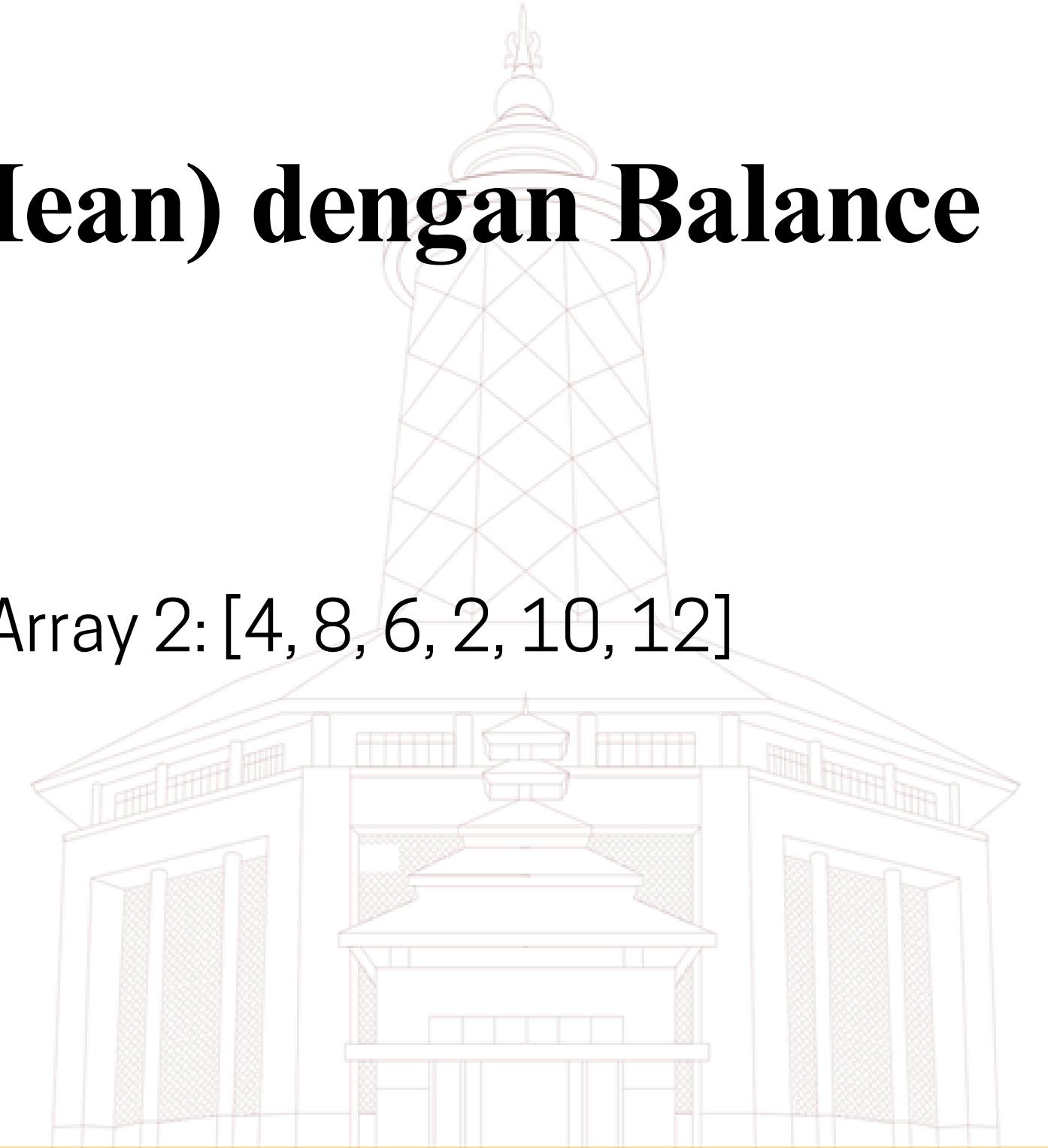
Ilustrasi

Balance

Kasus 1: Menghitung Rata-rata (Mean) dengan Balance

Array 1: [3, 5, 2, 7, 1]

Array 2: [4, 8, 6, 2, 10, 12]





Ilustrasi

Balance

Array 1: [3, 5, 2, 7, 1]

- Langkah 1 (Balance):

Pecah array menjadi dua bagian yang seimbang:

Bagian 1: [3, 5] (2 elemen)

Bagian 2: [2, 7, 1] (3 elemen)

- Langkah 2 (Conquer):

Hitung jumlah di setiap bagian:

Bagian 1: $3 + 5 = 8$

Bagian 2: $2 + 7 + 1 = 10$

- Langkah 3 (Combine):

Gabungkan hasil total dan bagi dengan jumlah elemen:

Jumlah total: $8 + 10 = 18$

Rata-rata: $18 / 5 = 3.6$



Ilustrasi

Balance

Array 2: [4, 8, 6, 2, 10, 12]

- Langkah 1 (Balance):

Pecah array menjadi dua bagian yang seimbang:

Bagian 1: [4, 8, 6] (3 elemen)

Bagian 2: [2, 10, 12] (3 elemen)

- Langkah 2 (Conquer):

Hitung jumlah di setiap bagian:

Bagian 1: $4 + 8 + 6 = 18$

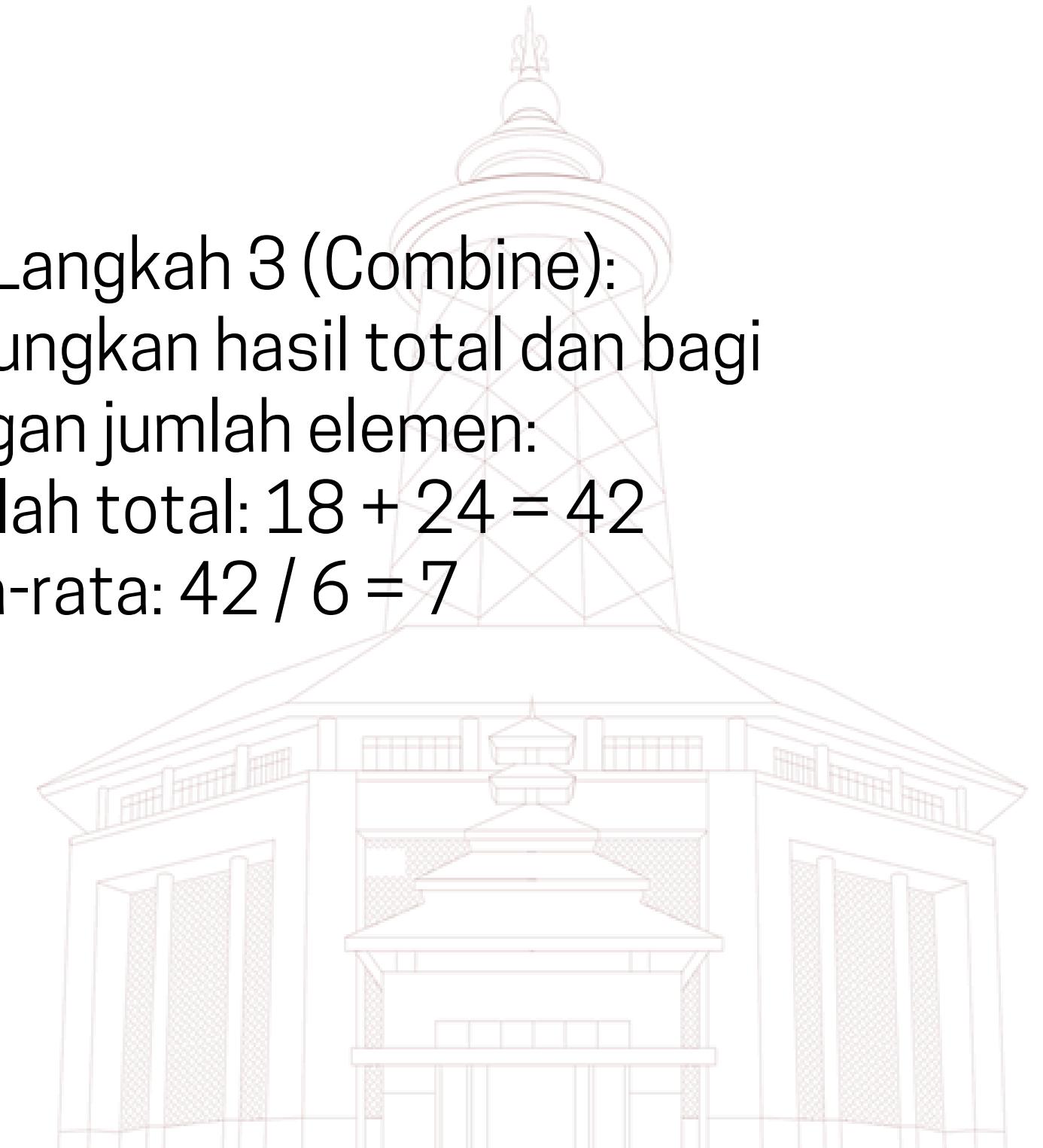
Bagian 2: $2 + 10 + 12 = 24$

- Langkah 3 (Combine):

Gabungkan hasil total dan bagi dengan jumlah elemen:

Jumlah total: $18 + 24 = 42$

Rata-rata: $42 / 6 = 7$





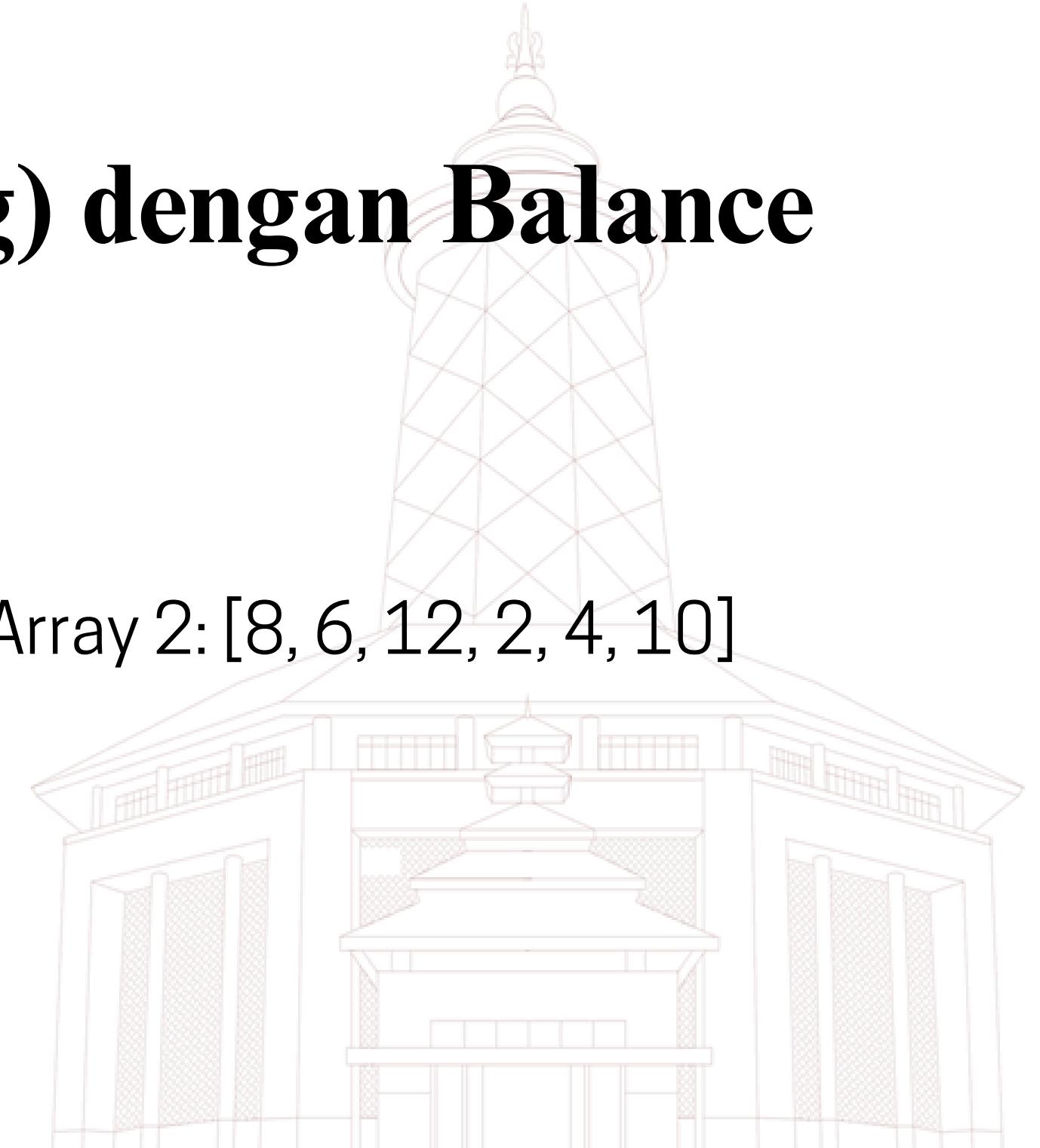
Ilustrasi

Balance-Time to participate

Kasus 2: Mengurutkan (Sorting) dengan Balance

Array 1: [3, 1, 9, 7, 5]

Array 2: [8, 6, 12, 2, 4, 10]





Ilustrasi

Balance

Array 1: [3, 1, 9, 7, 5]

- Langkah 1 (Balance):

Pecah array menjadi dua bagian yang seimbang:

Bagian 1: [3, 1] (2 elemen)

Bagian 2: [9, 7, 5] (3 elemen)

- Langkah 2 (Conquer):

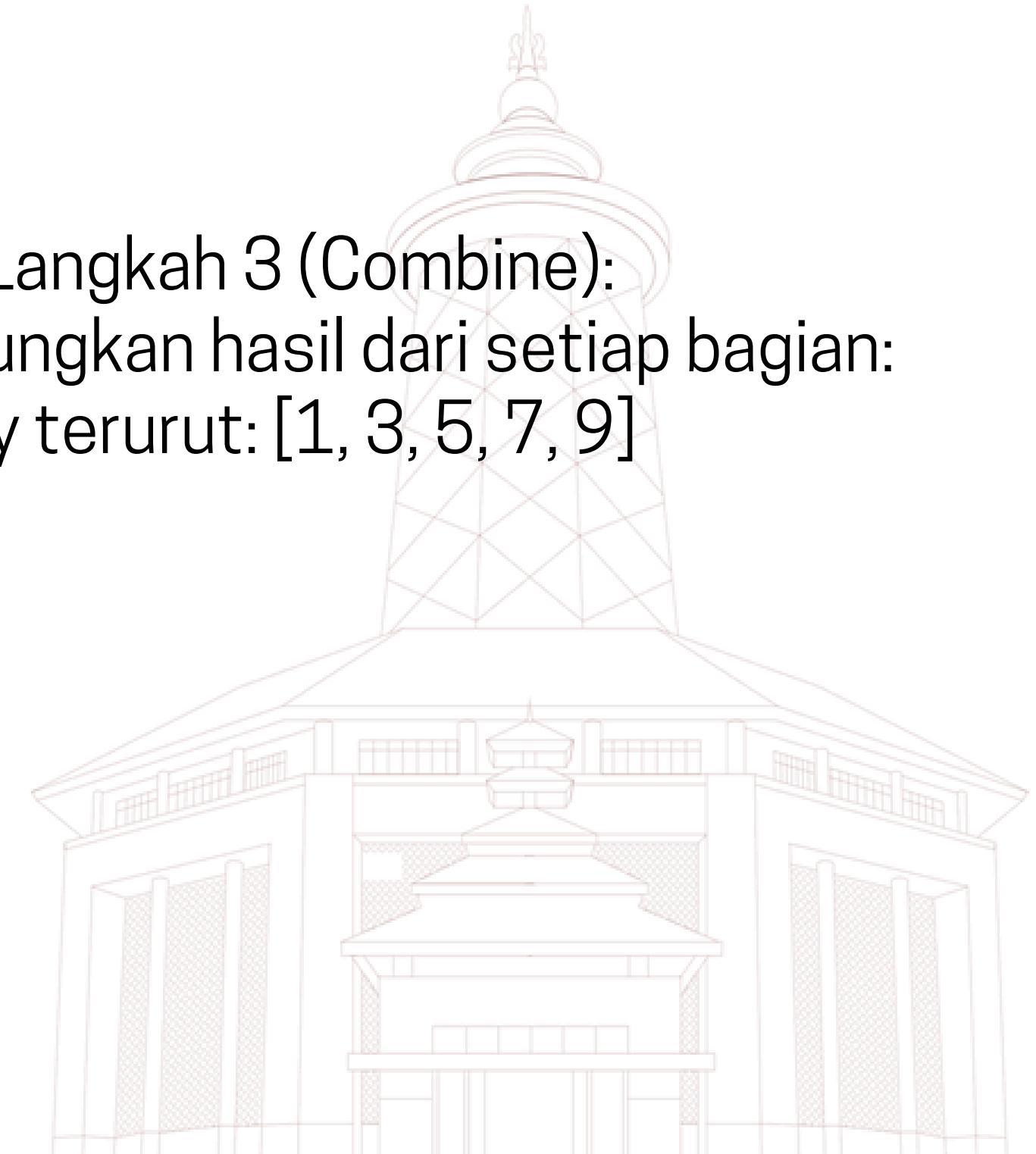
Urutkan setiap bagian:

Bagian 1: [1, 3]

Bagian 2: [5, 7, 9]

- Langkah 3 (Combine):

Gabungkan hasil dari setiap bagian:
Array terurut: [1, 3, 5, 7, 9]





Ilustrasi

Balance

Array 2: [8, 6, 12, 2, 4, 10]

- Langkah 1 (Balance):

Pecah array menjadi dua bagian yang seimbang:

Bagian 1: [8, 6, 12] (3 elemen)

Bagian 2: [2, 4, 10] (3 elemen)

- Langkah 2 (Conquer):

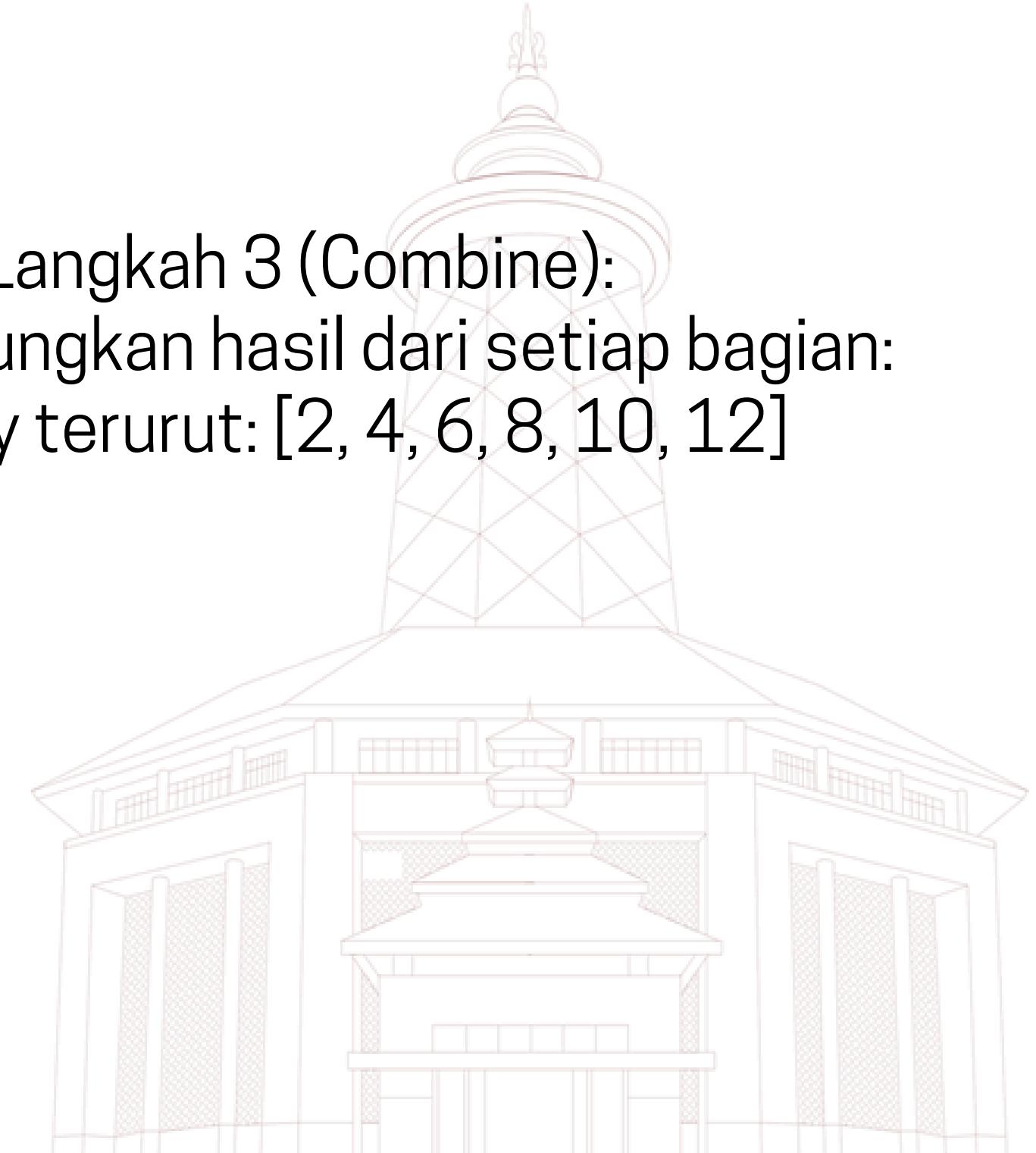
Urutkan setiap bagian:

Bagian 1: [6, 8, 12]

Bagian 2: [2, 4, 10]

- Langkah 3 (Combine):

Gabungkan hasil dari setiap bagian:
Array terurut: [2, 4, 6, 8, 10, 12]





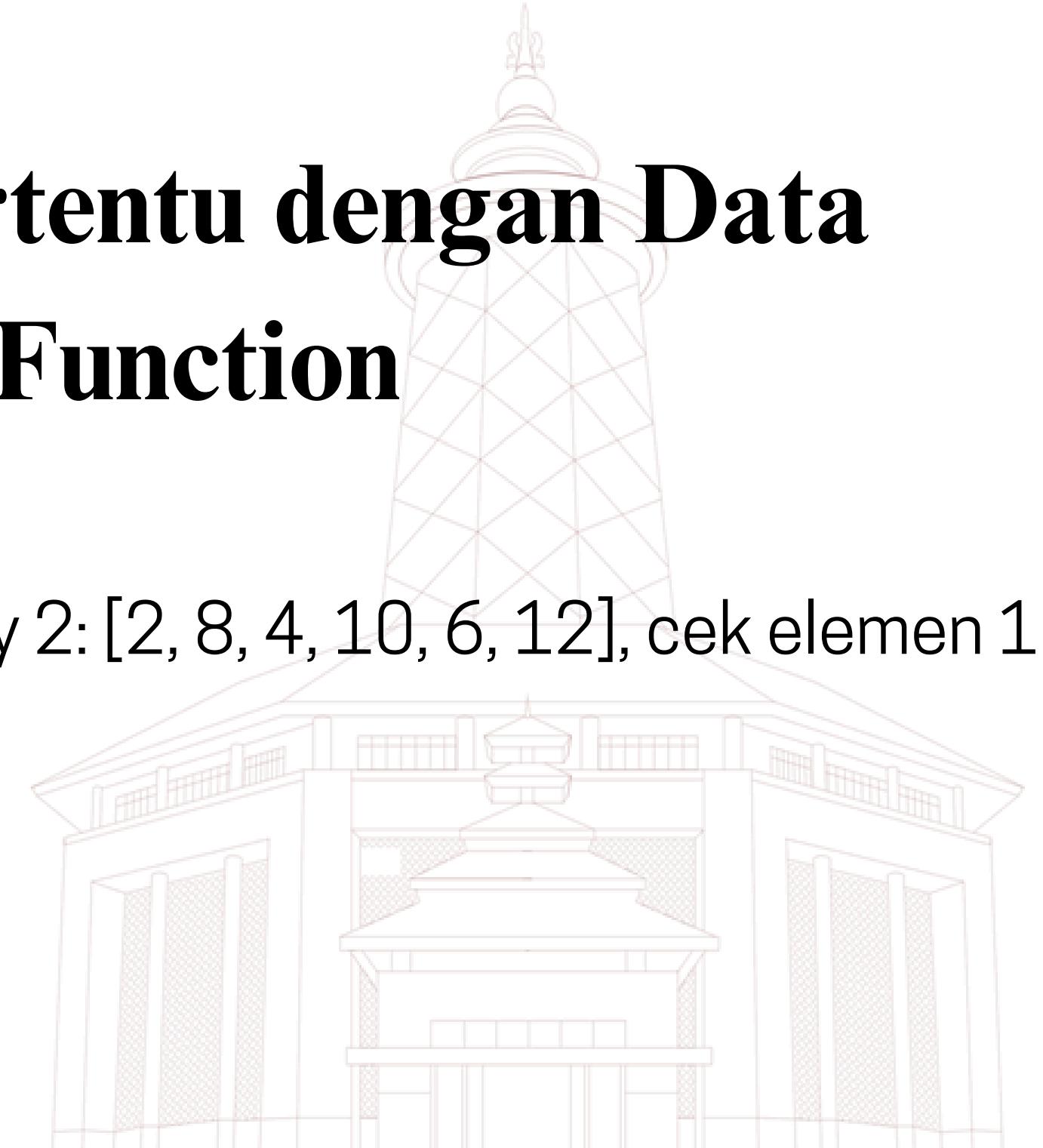
Ilustrasi

Data Dependence of Divide Function

Kasus 1: Memeriksa Elemen Tertentu dengan Data Dependence of Divide Function

Array 1: [3, 5, 1, 7, 9], cek elemen 5

Array 2: [2, 8, 4, 10, 6, 12], cek elemen 10





Ilustrasi

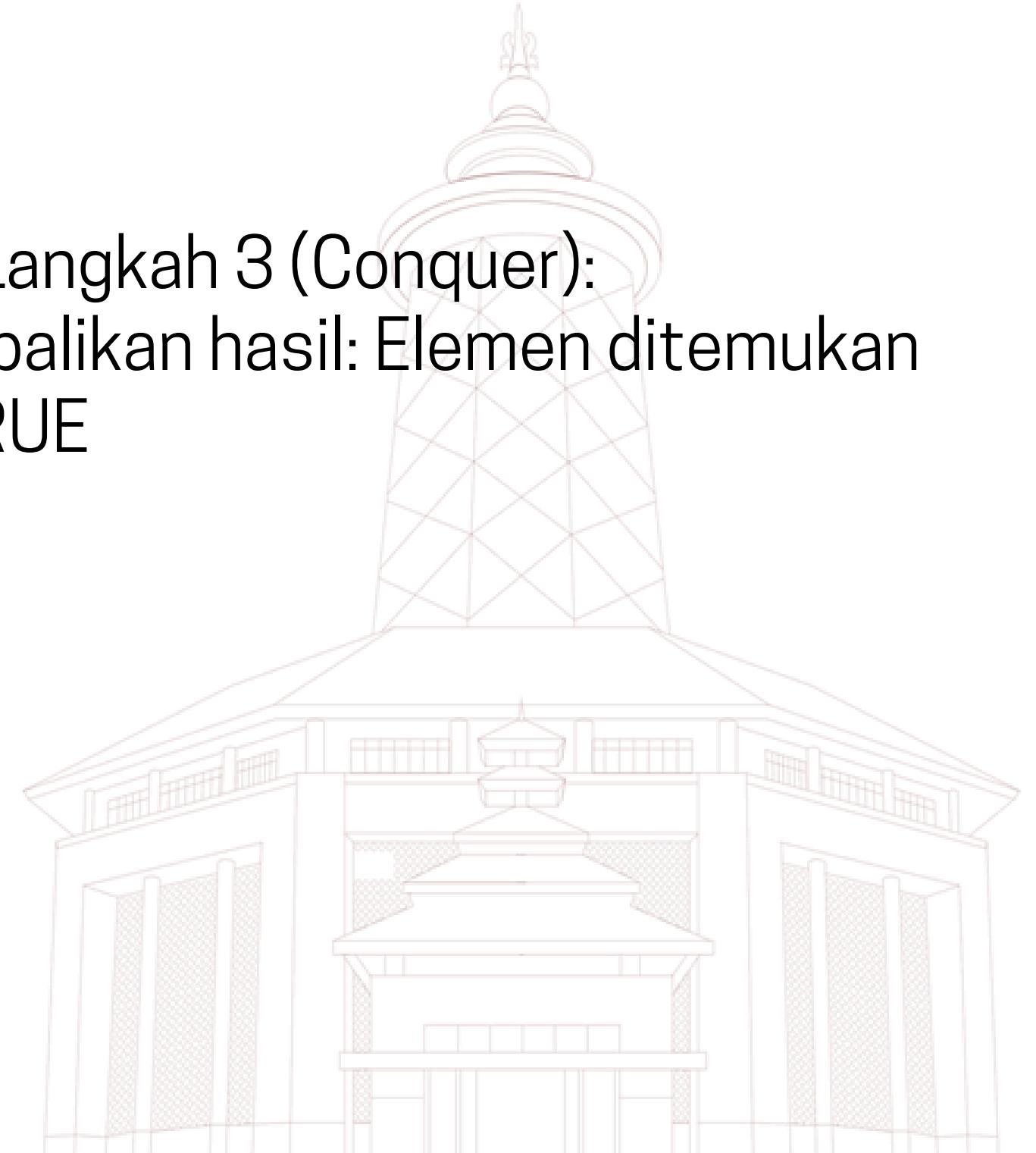
Data Dependence of Divide Function

Array 1: [3, 5, 1, 7, 9], cek elemen 5

- Langkah 1 (Prasyarat):
Urutkan array: [1, 3, 5, 7, 9]

- Langkah 2 (Divide):
Cek elemen di tengah (elemen ketiga): 5.
Karena elemen yang dicari adalah 5,
pencarian berhenti.

- Langkah 3 (Conquer):
Kembalikan hasil: Elemen ditemukan
→ TRUE





Ilustrasi

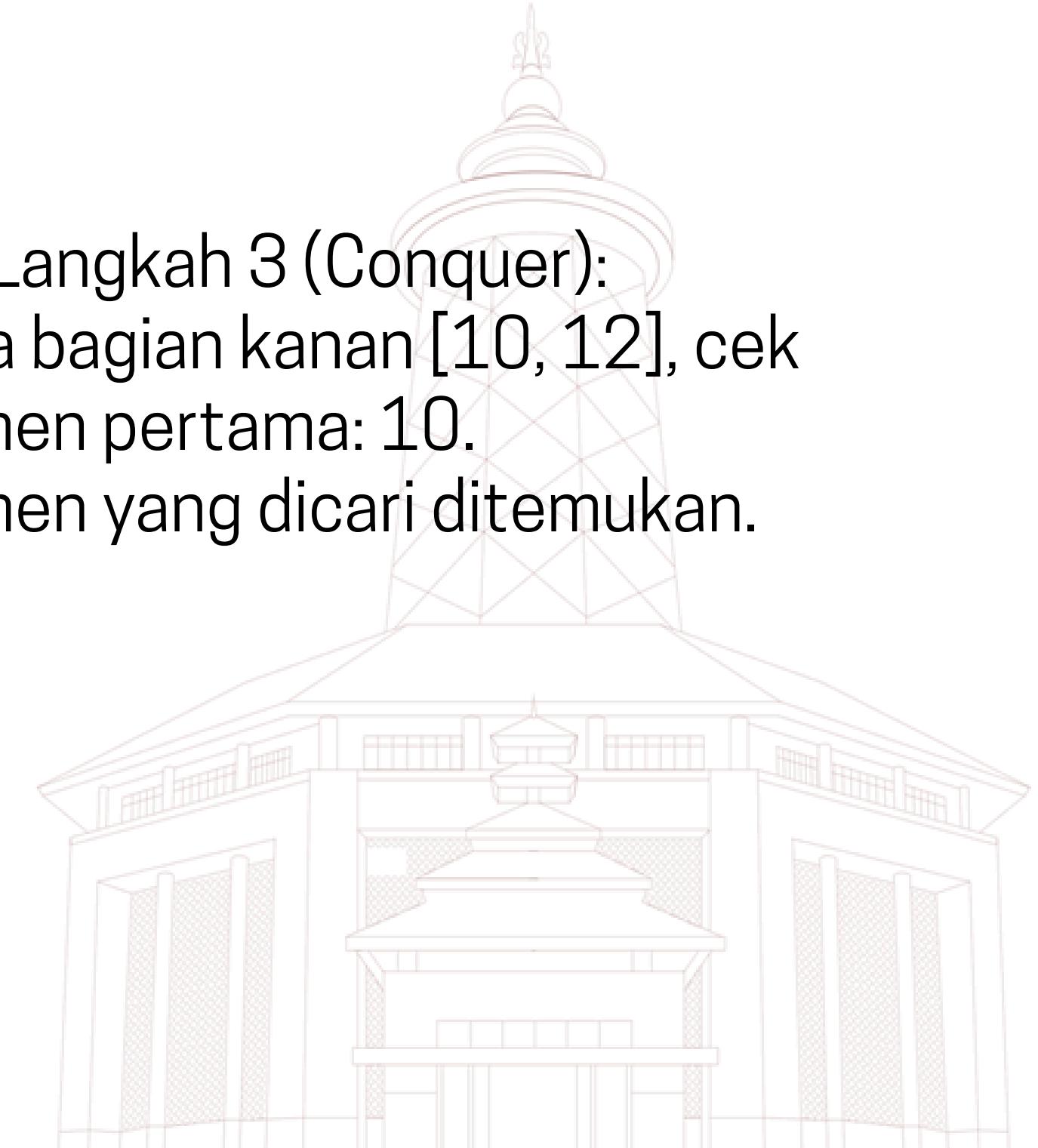
Data Dependence of Divide Function

Array 2: [2, 8, 4, 10, 6, 12], cek elemen 10

- Langkah 1 (Prasyarat):
Urutkan array: [2, 4, 6, 8, 10, 12]

- Langkah 2 (Divide):
Cek elemen tengah:
Elemen tengah adalah 6 (posisi ketiga) dan 8 (posisi keempat).
Karena 10 lebih besar dari kedua elemen tengah (6 dan 8), pencarian dilanjutkan pada bagian kanan array, yaitu [10, 12]

- Langkah 3 (Conquer):
Pada bagian kanan [10, 12], cek elemen pertama: 10.
Elemen yang dicari ditemukan.





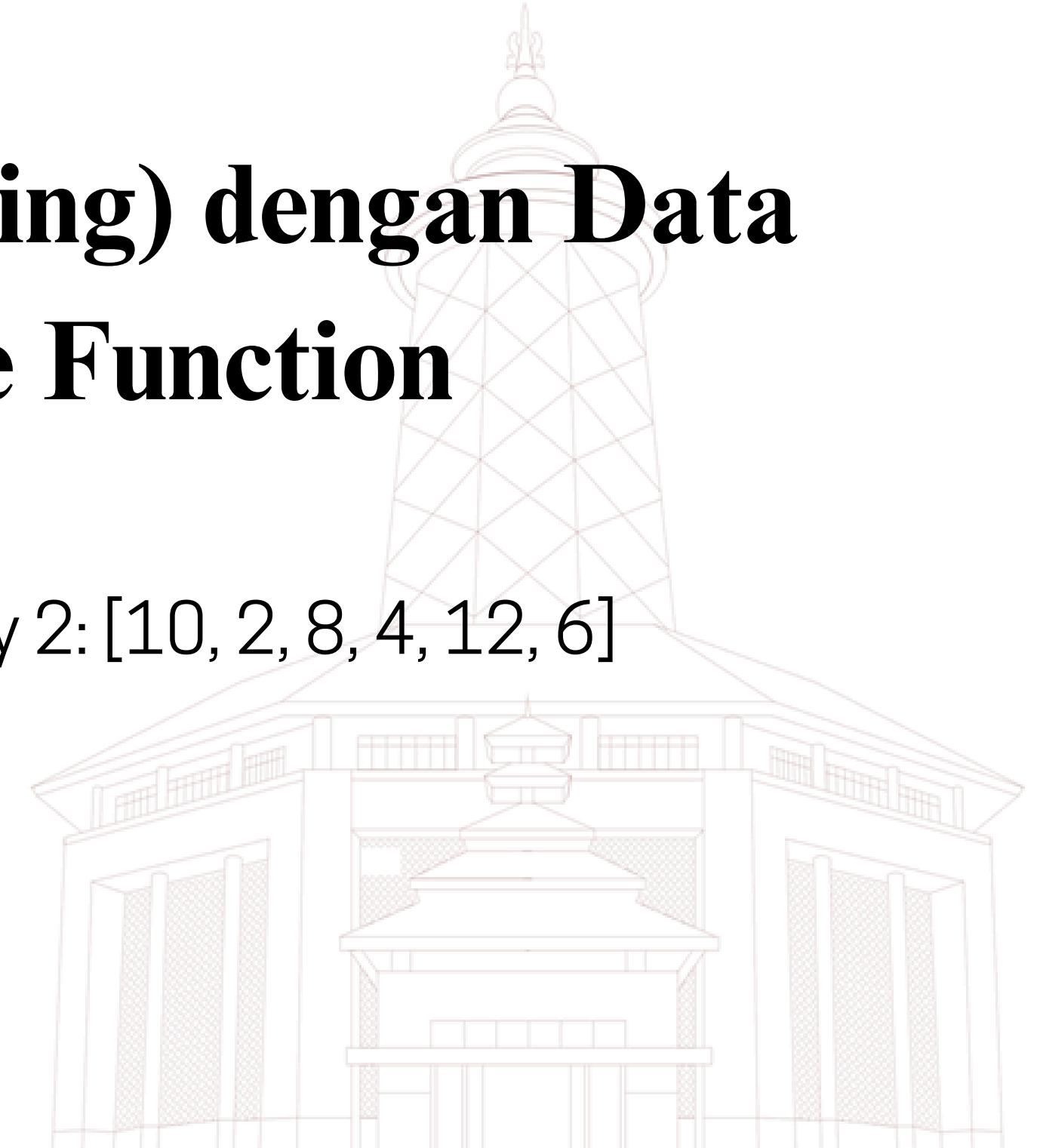
Ilustrasi

Data Dependence of Divide Function-Time to participate

Kasus 2: Mengurutkan (Sorting) dengan Data Dependence of Divide Function

Array 1: [5, 3, 9, 7, 1]

Array 2: [10, 2, 8, 4, 12, 6]



Ilustrasi

Data Dependence of Divide Function

Array 1: [5, 3, 9, 7, 1]

- Langkah 1 (Divide):

Pilih satu elemen sebagai "Pivot" (nilai acuan). Mari kita pilih 5.

Bagi array berdasarkan nilai 5:

- Bagian Kiri: Kumpulkan semua elemen yang lebih kecil dari 5 → [3, 1]
- Bagian Kanan: Kumpulkan semua elemen yang lebih besar dari 5 → [9, 7]

- Langkah 3 (Conquer):

Urutkan [3, 1] → hasilnya [1, 3]

Urutkan [9, 7] → hasilnya [7, 9]

- Langkah 4 (Combine):

Gabungkan hasil menjadi array terurut: [1, 3] + [5] + [7, 9]

Hasil akhir: [1, 3, 5, 7, 9]

Ilustrasi

Data Dependence of Divide Function

Array 2: [10, 2, 8, 4, 12, 6]

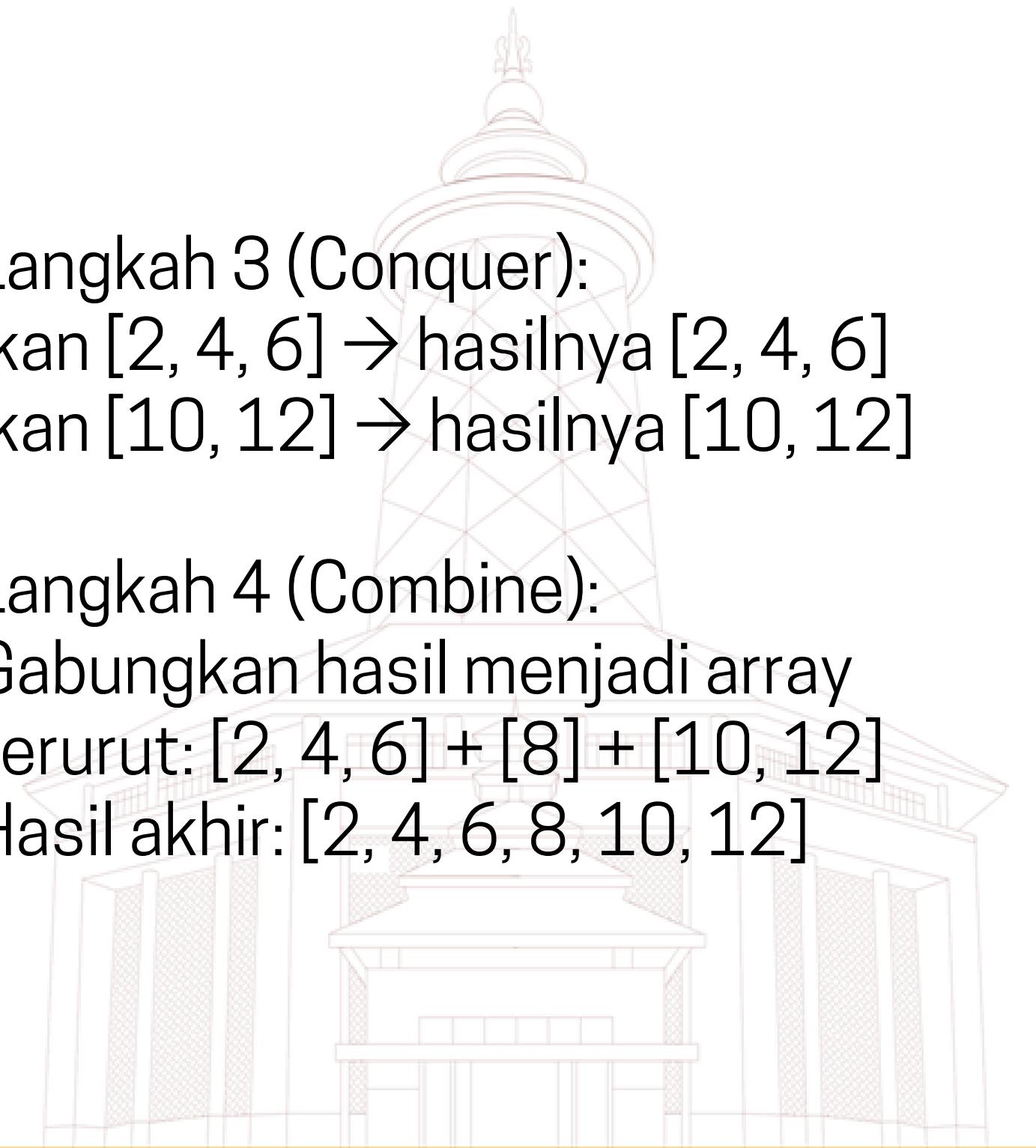
- Langkah 1 (Divide):
Pilih satu elemen sebagai "Pivot" (nilai acuan). Mari kita pilih 8.

Bagi array berdasarkan nilai 8:

- Bagian Kiri: Kumpulkan semua elemen yang lebih kecil dari 8 → [2, 4, 6]
- Bagian Kanan: Kumpulkan semua elemen yang lebih besar dari 8 → [10, 12]

- Langkah 3 (Conquer):
Urutkan [2, 4, 6] → hasilnya [2, 4, 6]
Urutkan [10, 12] → hasilnya [10, 12]

- Langkah 4 (Combine):
 - Gabungkan hasil menjadi array terurut: [2, 4, 6] + [8] + [10, 12]
 - Hasil akhir: [2, 4, 6, 8, 10, 12]





Ilustrasi

Control Parallelism

Kasus 1: Menghitung Jumlah (Sum) dengan Control Parallelism

Array 1: [3, 5, 2, 7, 1]

Array 2: [4, 8, 6, 2, 10, 12]





Ilustrasi

Control Parallelism

Array 1: [3, 5, 2, 7, 1]

- Langkah 1 (Divide):

Pecah array menjadi bagian yang lebih kecil untuk diproses secara paralel:

Bagian 1: [3, 5]

Bagian 2: [2, 7]

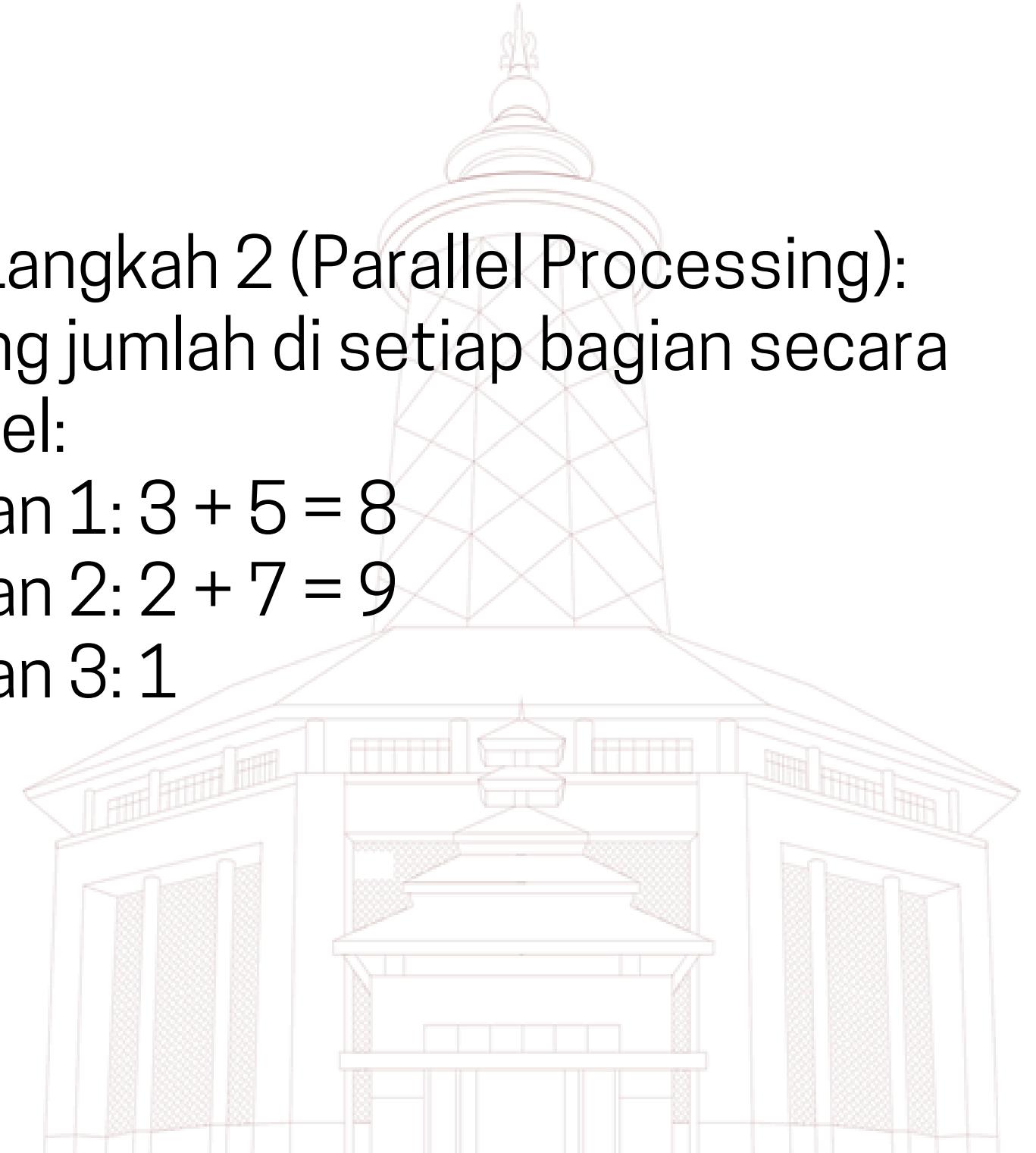
Bagian 3: [1]

- Langkah 2 (Parallel Processing):

Gabungkan hasil dari semua bagian:

Jumlah total: $8 + 9 + 1 = 18$

- Langkah 2 (Parallel Processing):
Hitung jumlah di setiap bagian secara paralel:
Bagian 1: $3 + 5 = 8$
Bagian 2: $2 + 7 = 9$
Bagian 3: 1





Ilustrasi

Control Parallelism

Array 2: [4, 8, 6, 2, 10, 12]

- Langkah 1 (Divide):

Pecah array menjadi bagian yang lebih kecil untuk diproses secara paralel:

Bagian 1: [4, 8]

Bagian 2: [6, 2]

Bagian 3: [10, 12]

- Langkah 2 (Parallel Processing):

Gabungkan hasil dari semua bagian:

Jumlah total: $12 + 8 + 22 = 42$

- Langkah 2 (Parallel Processing):

Hitung jumlah di setiap bagian secara paralel:

Bagian 1: $4 + 8 = 12$

Bagian 2: $6 + 2 = 8$

Bagian 3: $10 + 12 = 22$



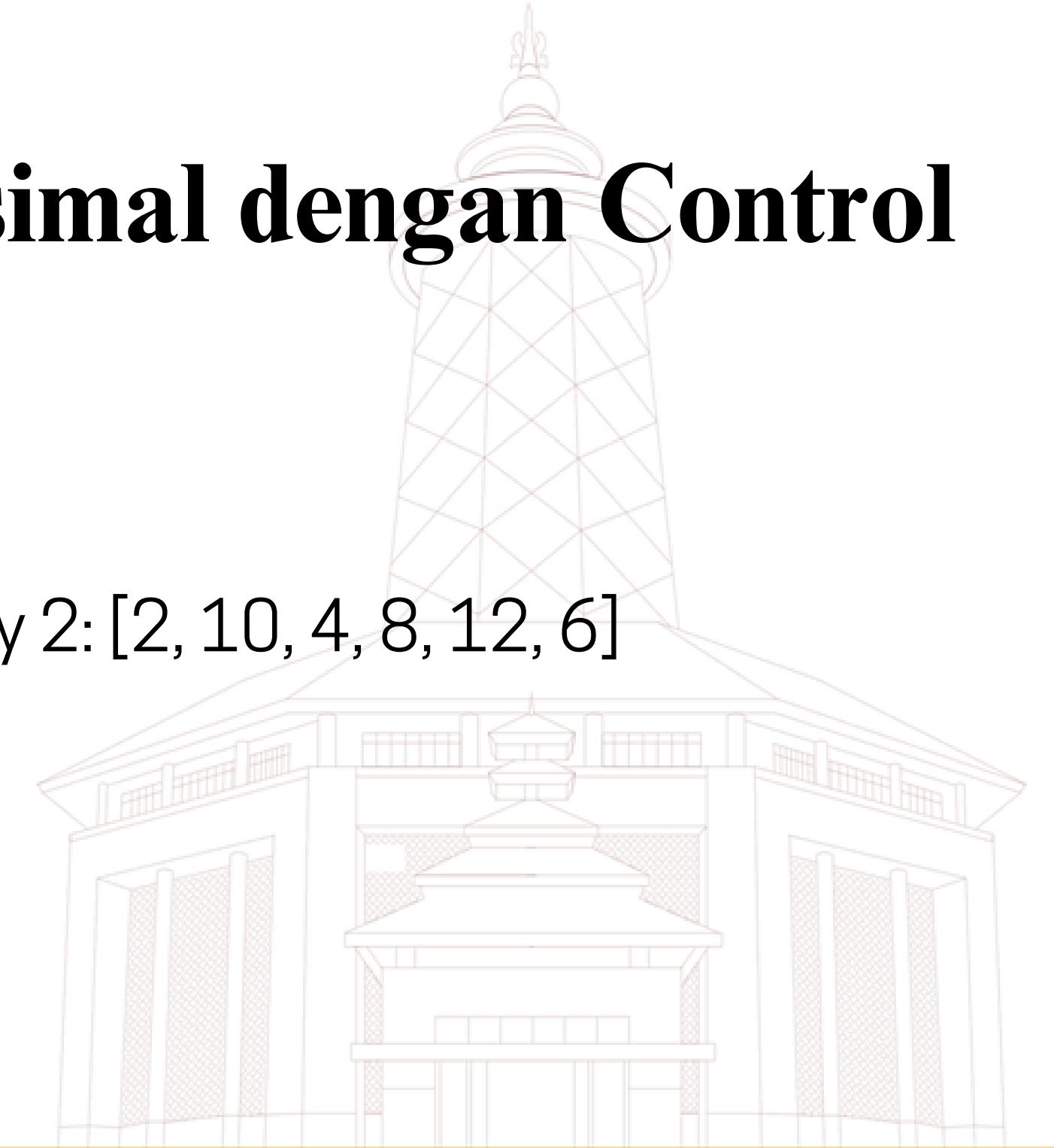
Ilustrasi

Control Parallelism

Kasus 2: Mencari Elemen Maksimal dengan Control Parallelism

Array 1: [3, 1, 9, 7, 5]

Array 2: [2, 10, 4, 8, 12, 6]





Ilustrasi

Control Parallelism

Array 1: [3, 1, 9, 7, 5]

- Langkah 1 (Divide):

Pecah array menjadi beberapa bagian yang lebih kecil untuk diproses secara paralel:

Bagian 1: [3, 1]

Bagian 2: [9, 7]

Bagian 3: [5]

- Langkah 2 (Parallel Processing):

Cari elemen maksimal di setiap bagian secara paralel:

Bagian 1: $\max(3, 1) = 3$

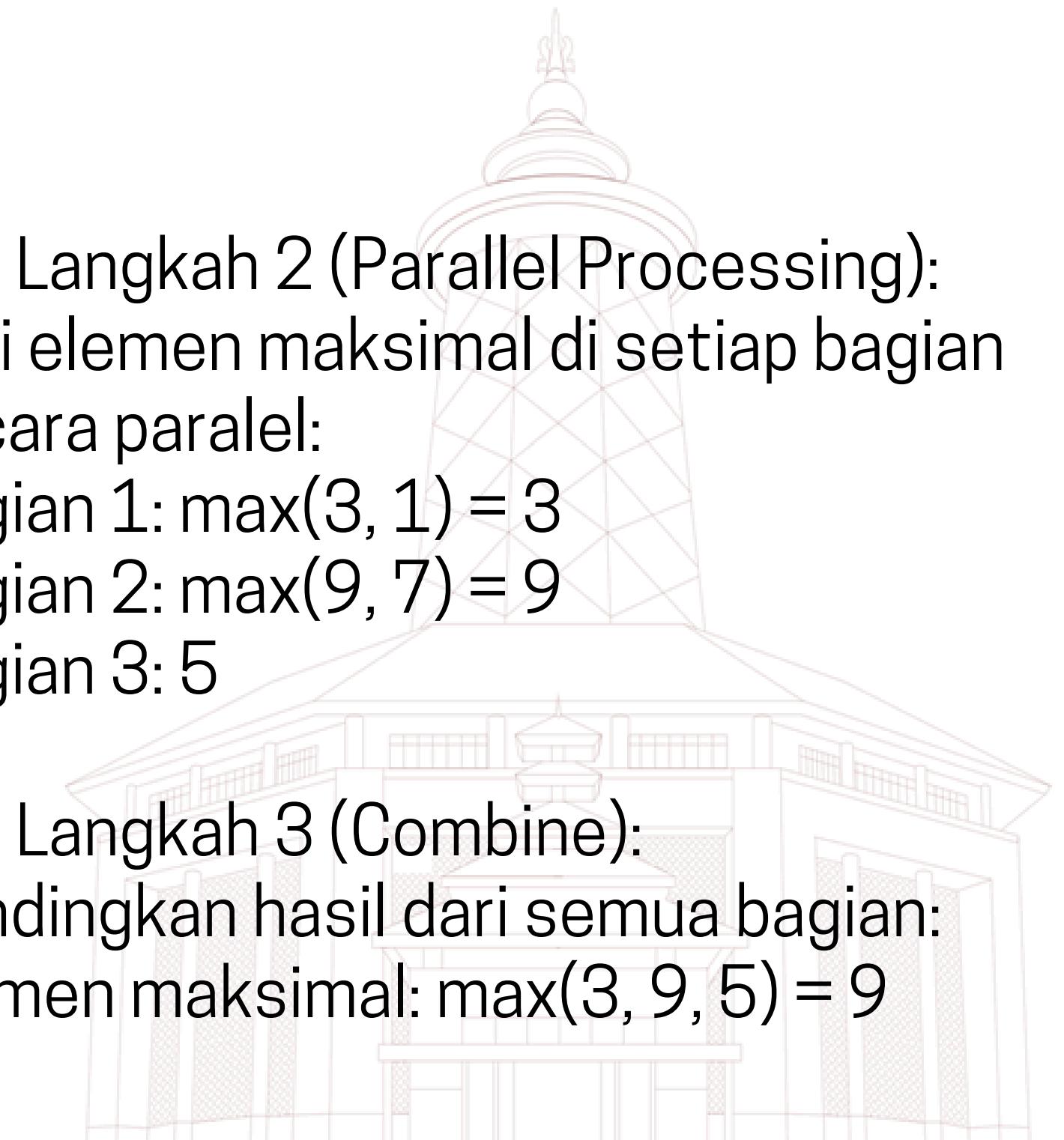
Bagian 2: $\max(9, 7) = 9$

Bagian 3: 5

- Langkah 3 (Combine):

Bandingkan hasil dari semua bagian:

Elemen maksimal: $\max(3, 9, 5) = 9$





Ilustrasi

Control Parallelism

Array 2: [2, 10, 4, 8, 12, 6]

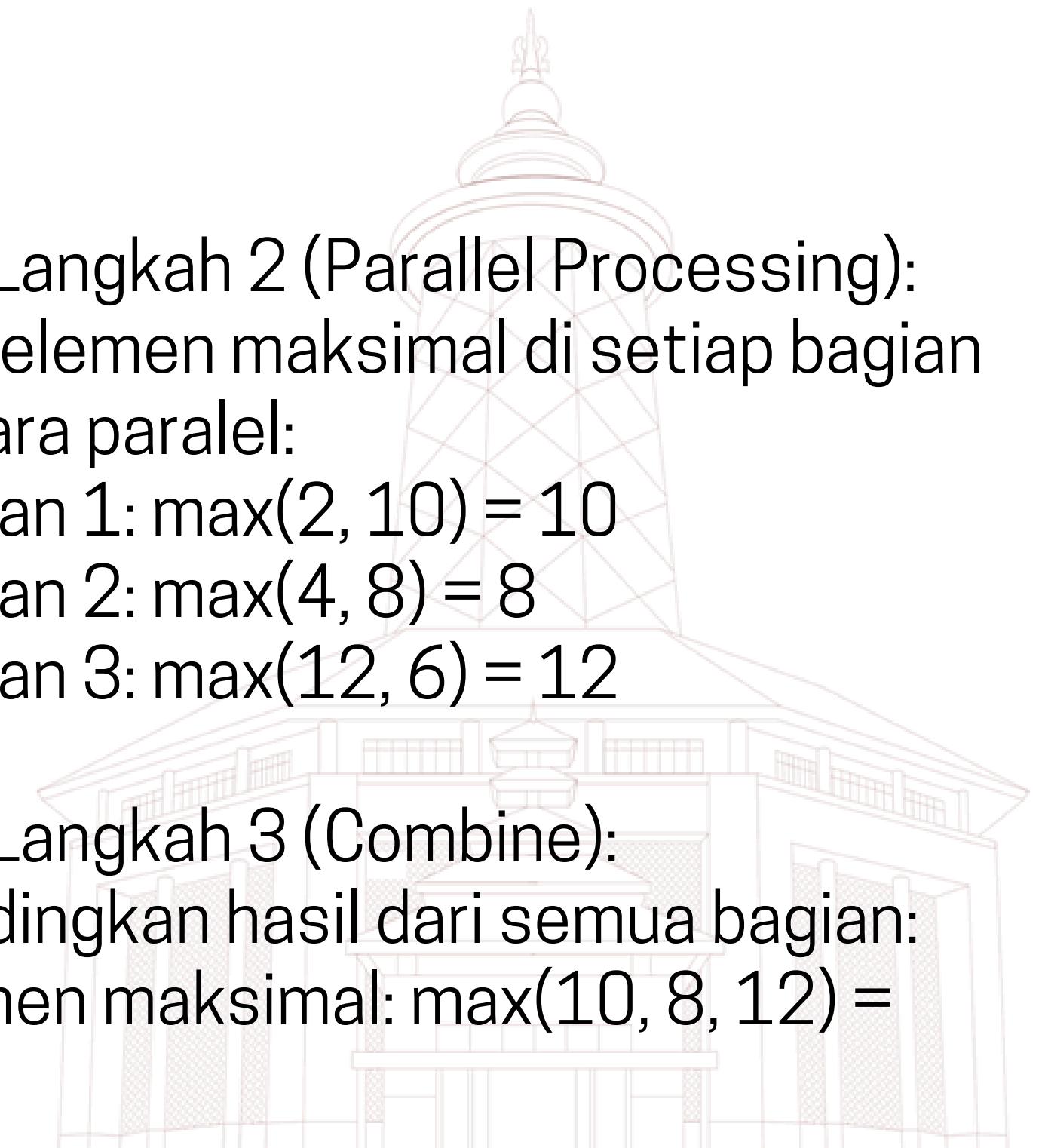
- Langkah 1 (Divide):
Pecah array menjadi beberapa bagian yang lebih kecil untuk diproses secara paralel:

Bagian 1: [2, 10]

Bagian 2: [4, 8]

Bagian 3: [12, 6]

- Langkah 2 (Parallel Processing):
Cari elemen maksimal di setiap bagian secara paralel:
Bagian 1: $\max(2, 10) = 10$
Bagian 2: $\max(4, 8) = 8$
Bagian 3: $\max(12, 6) = 12$
- Langkah 3 (Combine):
Bandingkan hasil dari semua bagian:
Elemen maksimal: $\max(10, 8, 12) = 12$





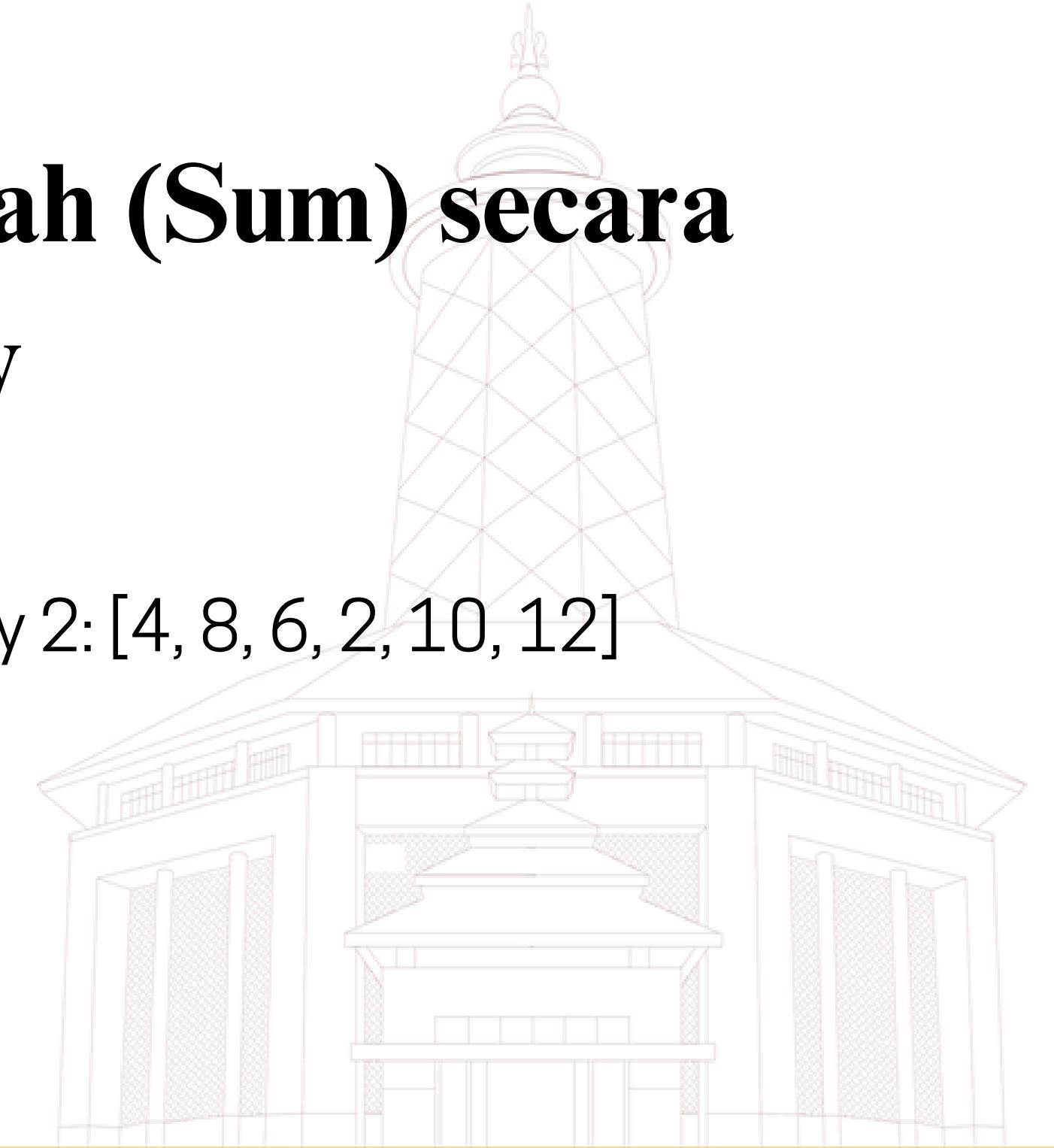
Ilustrasi

Sequentially

Kasus 1: Menghitung Jumlah (Sum) secara Sequentially

Array 1: [3, 5, 2, 7, 1]

Array 2: [4, 8, 6, 2, 10, 12]





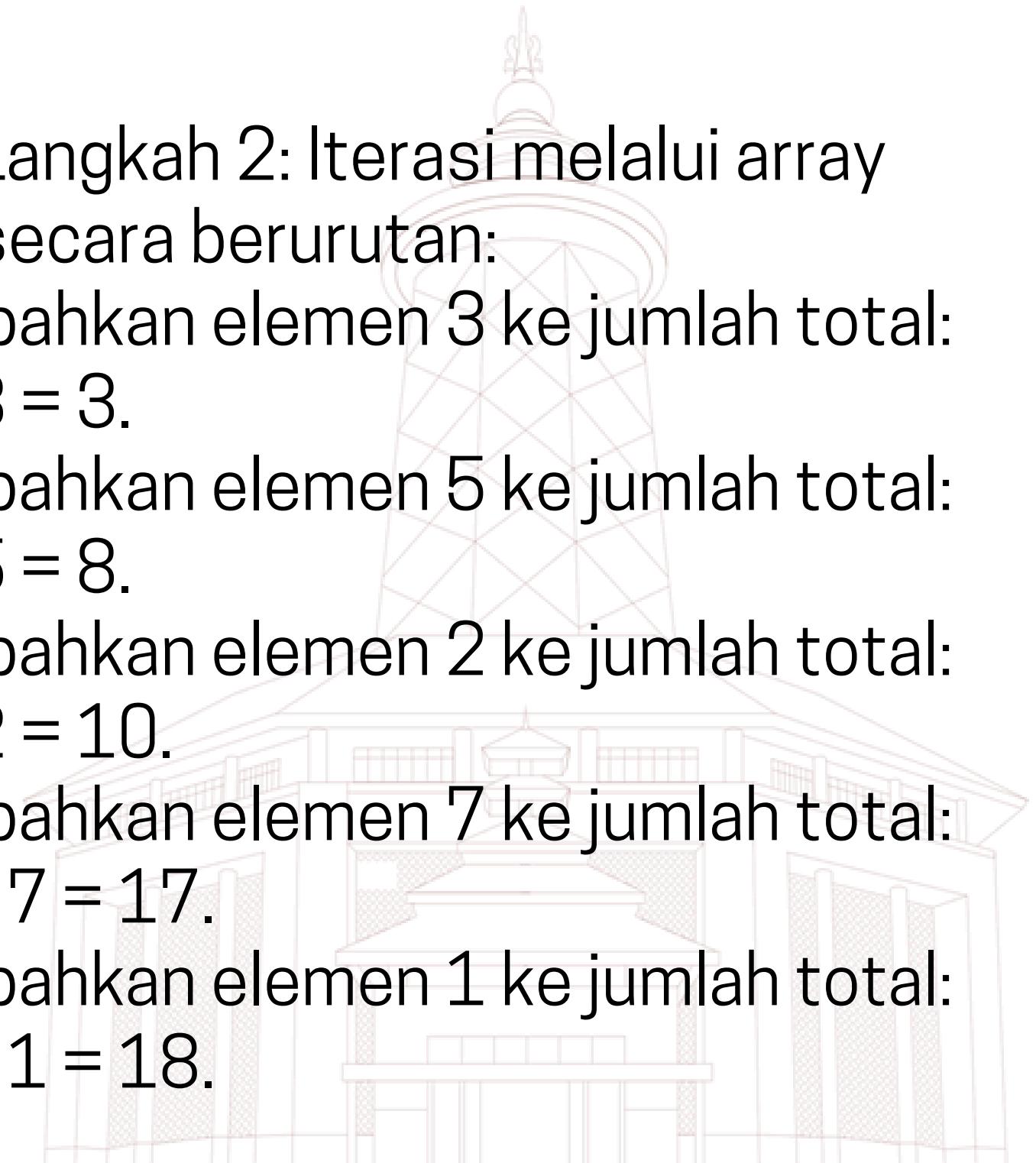
Ilustrasi

Sequentially

Array 1: [3, 5, 2, 7, 1]

- Langkah 1: Inisialisasi jumlah total dengan nilai 0
- Langkah 3: Kembalikan hasil: 18.

- Langkah 2: Iterasi melalui array secara berurutan:
Tambahkan elemen 3 ke jumlah total:
 $0 + 3 = 3$.
Tambahkan elemen 5 ke jumlah total:
 $3 + 5 = 8$.
Tambahkan elemen 2 ke jumlah total:
 $8 + 2 = 10$.
Tambahkan elemen 7 ke jumlah total:
 $10 + 7 = 17$.
Tambahkan elemen 1 ke jumlah total:
 $17 + 1 = 18$.



Ilustrasi

Sequentially

Array 2: [4, 8, 6, 2, 10, 12]

- Langkah 1: Inisialisasi jumlah total dengan nilai 0
- Langkah 3: Kembalikan hasil: 42.

- Langkah 2: Iterasi melalui array secara berurutan:
Tambahkan elemen 4 ke jumlah total:
 $0 + 4 = 4$.
Tambahkan elemen 8 ke jumlah total:
 $4 + 8 = 12$.
Tambahkan elemen 6 ke jumlah total:
 $12 + 6 = 18$.
Tambahkan elemen 2 ke jumlah total:
 $18 + 2 = 20$.
Tambahkan elemen 10 ke jumlah total:
 $20 + 10 = 30$.
Tambahkan elemen 12 ke jumlah total:
 $30 + 12 = 42$



Ilustrasi

Sequentially

Kasus 2: Mencari Elemen Maksimal secara Sequentially

Array 1: [3, 1, 9, 7, 5]

Array 2: [2, 10, 4, 8, 12, 6]



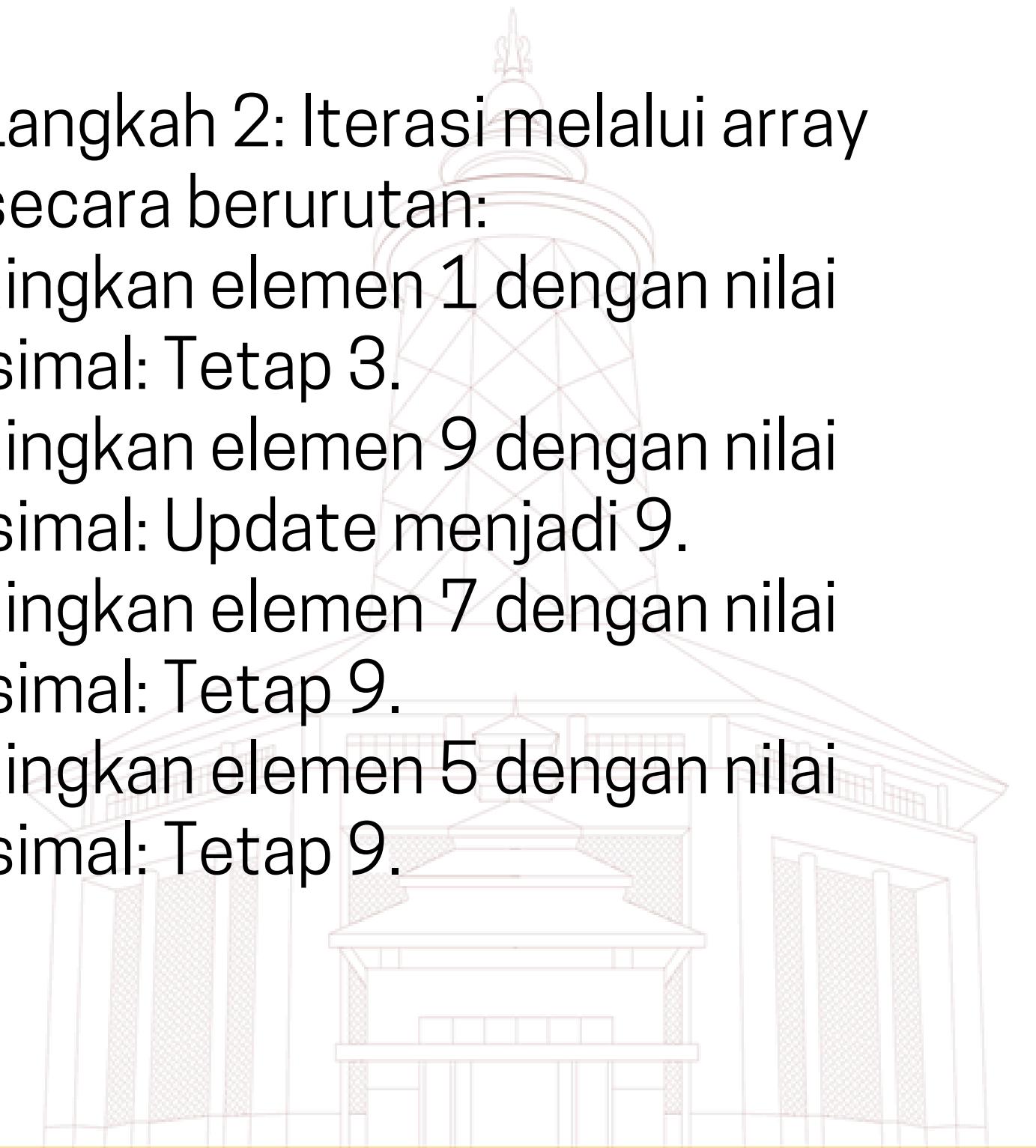


Ilustrasi

Sequentially

Array 1: [3, 1, 9, 7, 5]

- Langkah 1: Inisialisasi nilai maksimal dengan elemen pertama (3)
- Langkah 2: Iterasi melalui array secara berurutan:
Bandingkan elemen 1 dengan nilai maksimal: Tetap 3.
Bandingkan elemen 9 dengan nilai maksimal: Update menjadi 9.
Bandingkan elemen 7 dengan nilai maksimal: Tetap 9.
Bandingkan elemen 5 dengan nilai maksimal: Tetap 9.
- Langkah 3: Kembalikan hasil: 9.





Ilustrasi

Sequentially

Array 2: [2, 10, 4, 8, 12, 6]

- Langkah 1: Inisialisasi nilai maksimal dengan elemen pertama (2).
- Langkah 3: Kembalikan hasil: 12.

- Langkah 2: Iterasi melalui array secara berurutan:
Bandingkan elemen 10 dengan nilai maksimal: Update menjadi 10.
Bandingkan elemen 4 dengan nilai maksimal: Tetap 10.
Bandingkan elemen 8 dengan nilai maksimal: Tetap 10.
Bandingkan elemen 12 dengan nilai maksimal: Update menjadi 12.
Bandingkan elemen 6 dengan nilai maksimal: Tetap 12.



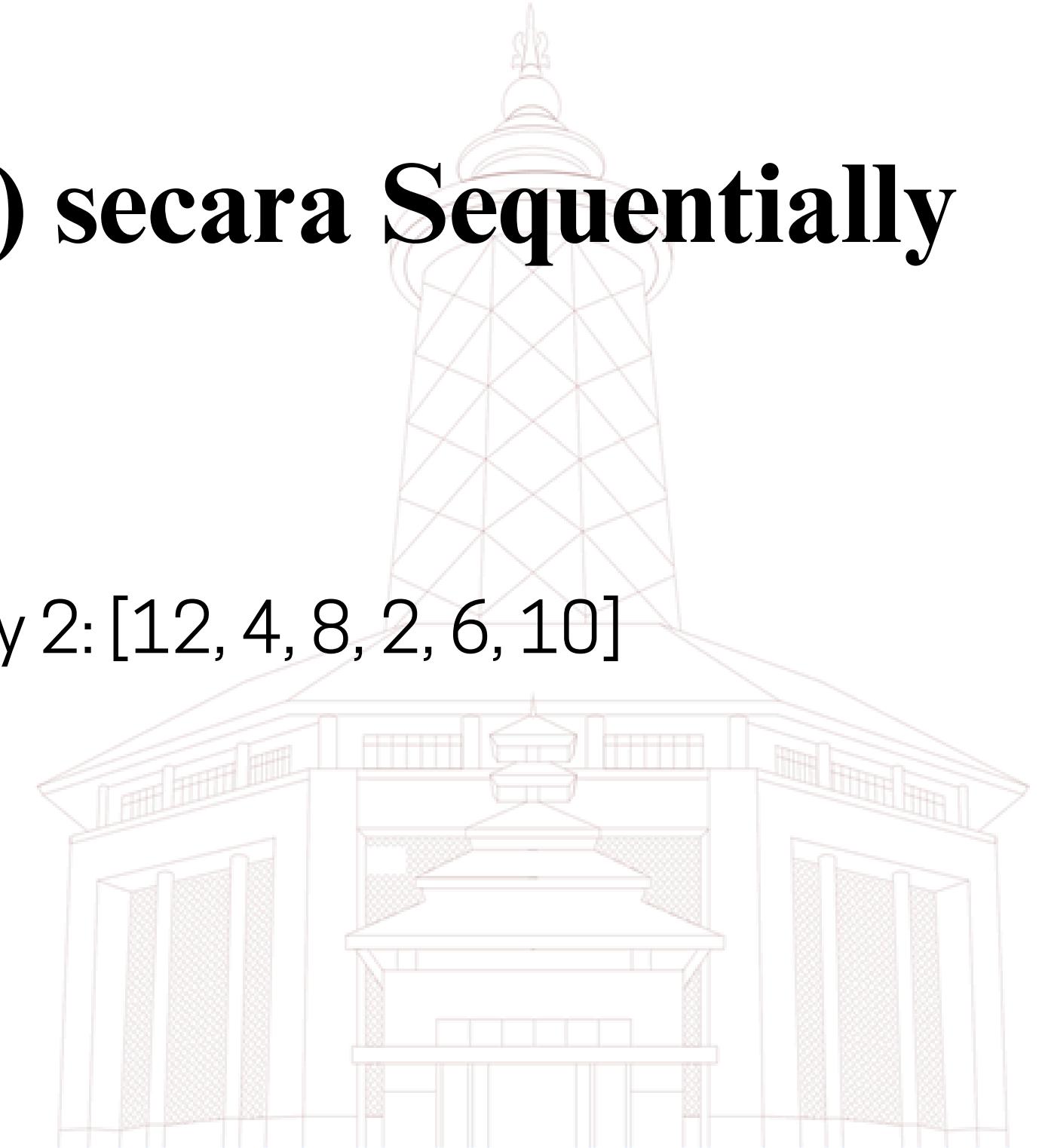
Ilustrasi

Sequentially - Time to participate

Kasus 3: Mengurutkan (Sorting) secara Sequentially

Array 1: [7, 5, 1, 9, 3]

Array 2: [12, 4, 8, 2, 6, 10]





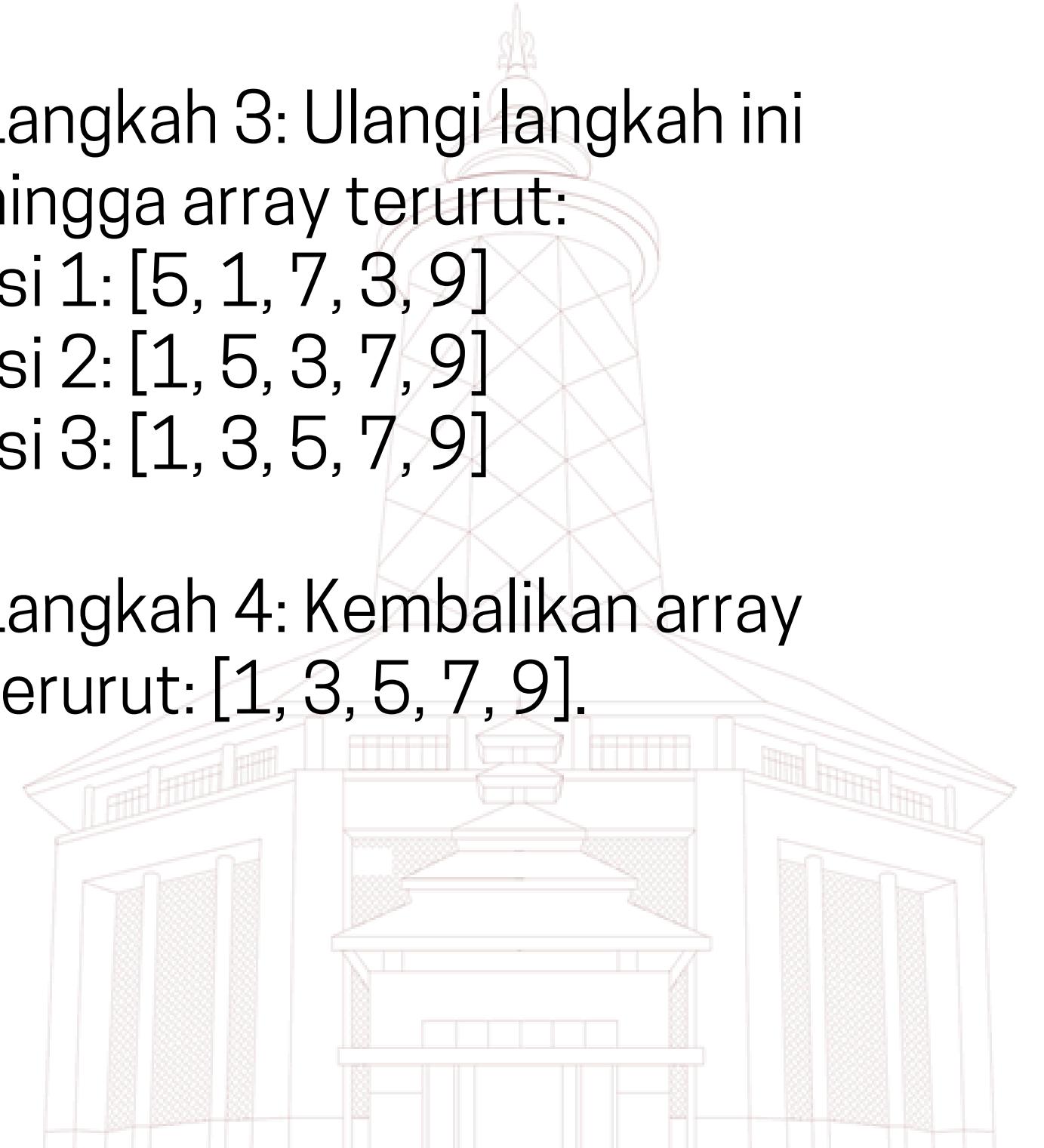
Ilustrasi

Sequentially

Array 1: [7, 5, 1, 9, 3]

- Langkah 1: Mulai iterasi dari elemen pertama dan bandingkan elemen yang berdekatan.
- Langkah 2: Jika elemen di kiri lebih besar, tukar posisi.

- Langkah 3: Ulangi langkah ini hingga array terurut:
Iterasi 1: [5, 1, 7, 3, 9]
Iterasi 2: [1, 5, 3, 7, 9]
Iterasi 3: [1, 3, 5, 7, 9]
- Langkah 4: Kembalikan array terurut: [1, 3, 5, 7, 9].



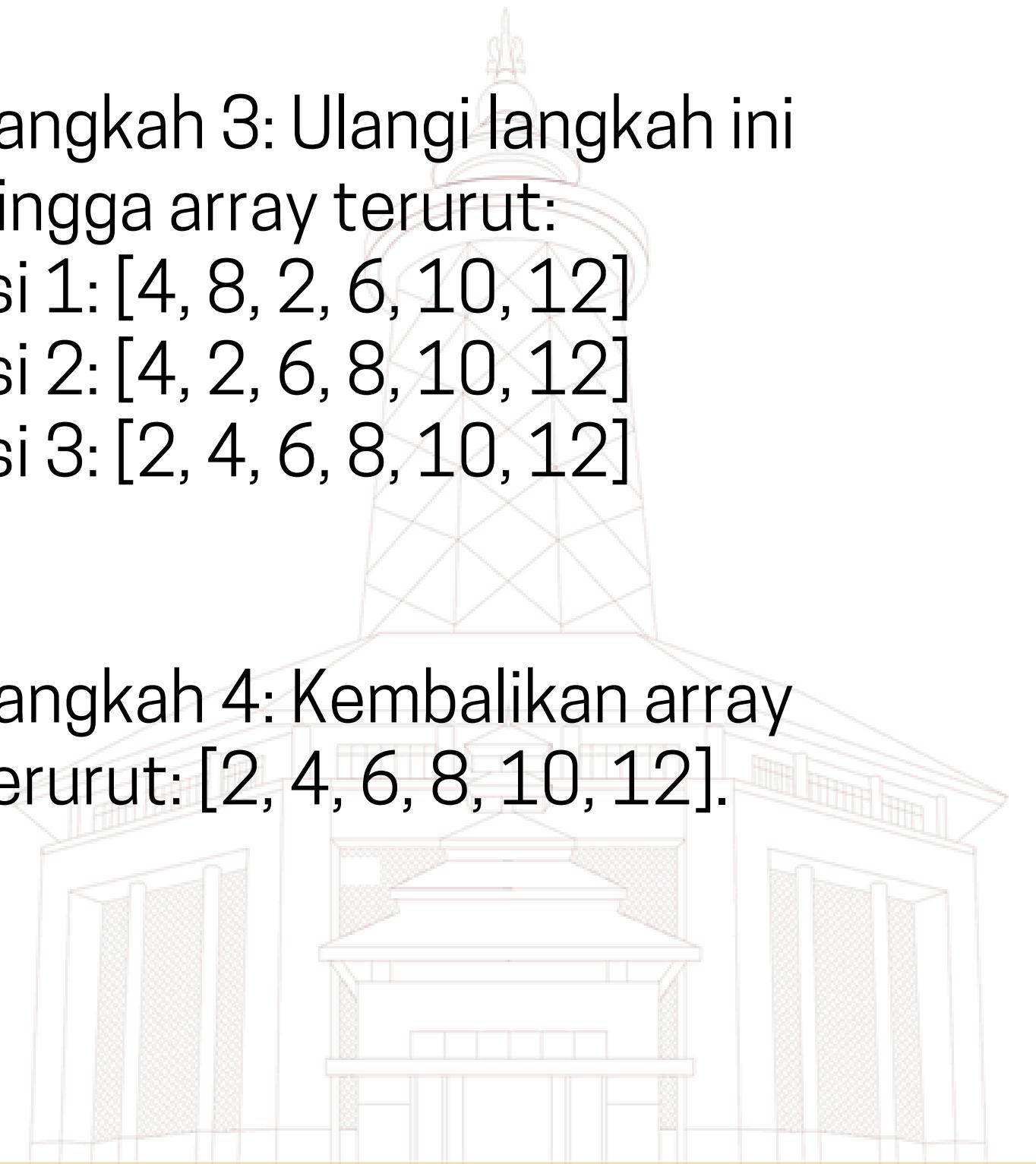
Ilustrasi

Sequentially

Array 2: [12, 4, 8, 2, 6, 10]

- Langkah 1: Mulai iterasi dari elemen pertama dan bandingkan elemen yang berdekatan.
- Langkah 2: Jika elemen di kiri lebih besar, tukar posisi.

- Langkah 3: Ulangi langkah ini hingga array terurut:
Iterasi 1: [4, 8, 2, 6, 10, 12]
Iterasi 2: [4, 2, 6, 8, 10, 12]
Iterasi 3: [2, 4, 6, 8, 10, 12]
- Langkah 4: Kembalikan array terurut: [2, 4, 6, 8, 10, 12].





SEE YOU NEXT WEEK !

Ferdian Bangkit Wijaya, S.Stat., M.Si
NIP. 199005202024061001
ferdian.bangkit@untirta.ac.id

