ASA 5

DECREASE AND CONQUER

DECREASE AND CONQUER

Decrease and Conquer adalah paradigma algoritma yang menyelesaikan suatu permasalahan dengan cara mengurangi ukuran masalah menjadi lebih kecil, kemudian menyelesaikan submasalah tersebut secara rekursif atau iteratif hingga mencapai kasus dasar.

Berbeda dengan Divide and Conquer yang membagi masalah menjadi beberapa bagian, Decrease and Conquer hanya mengurangi masalah menjadi satu bagian yang lebih kecil.

TAHAPAN

- Decrease (Mereduksi)
  - Mereduksi masalah menjadi dua atau lebih sub-masalah yang lebih kecil dengan karakteristik yang sama atau serupa.
- Conquer (Penyelesaian)
  - Menyelesaikan satu sub-masalah secara rekursif/iteratif. Jika sub-masalah cukup kecil, langsung selesaikan secara langsung (base case).

KATEGORI

1. Decrease by a Constant: Mengurangi ukuran masalah dengan nilai tetap, misalnya mengurangi ukuran masalah sebanyak constant (n-1, n-2, n-3, dst).
2. Decrease by a Constant Factor: Mengurangi ukuran masalah dengan faktor tertentu (n/2, n/3, n/4, dst).
3. Variable-Size Decrease: Mengurangi ukuran masalah dengan jumlah yang bervariasi berdasarkan kondisi tertentu.

DECREASE BY A CONSTANT

Insertion Sort adalah contoh dari Decrease by a Constant, karena setiap langkahnya memasukkan **satu** elemen ke dalam posisi yang benar di dalam bagian array yang telah diurutkan.

5	3	8	6	2	(Swap index ke-1 dan ke-5)
2	3	8	6	5	(Posisi sudah sesuai)
2	3	8	6	5	(Swap index ke-3 dan ke-5)
2	3	5	6	8	(Posisi sudah sesuai)
2	3	5	6	8	(Posisi sudah sesuai)
2	3	5	6	8	

DECREASE BY A CONSTANT

```python
def insertion_sort_iterative(arr):
    for i in range(1, len(arr)):
        key = arr[i]
        j = i - 1
        while j >= 0 and arr[j] > key:
            arr[j + 1] = arr[j]
            j -= 1
        arr[j + 1] = key
    return arr
```

```python
def insertion_sort_recursive(arr, n):
    if n <= 1:
        return arr

    # Urutkan subarray sebelumnya
    insertion_sort_recursive(arr, n - 1)

    # Masukkan elemen terakhir ke posisi yang benar
    key = arr[n - 1]
    j = n - 2
    while j >= 0 and arr[j] > key:
        arr[j + 1] = arr[j]
        j -= 1
    arr[j + 1] = key
    return arr
```

DECREASE BY A CONSTANT FACTOR

Binary Search adalah contoh dari Decrease by a Constant Factor, karena setiap langkahnya mengurangi ukuran masalah menjadi **setengah**nya.

0	1	2	3	4	5	6	7	8	9	10			
2	3	5	6	8	10	15	17	18	20	23	L = 0	R = 10	mid = (0+8)/2 = 5
2	3	5	6	8							L = 0	R = 4	mid = (0+4)/2 = 2
			6	8							L = 3	R = 4	mid = (3+4)/2 = 3

DECREASE BY A CONSTANT FACTOR

```python
def binary_search_iterative(arr, target):
    low, high = 0, len(arr) - 1
    while low <= high:
        mid = (low + high) // 2

        # Jika target ditemukan
        if arr[mid] == target:
            return mid
        # Jika target lebih kecil dari mid
        elif arr[mid] > target:
            high = mid - 1
        # Jika target lebih besar dari mid
        else:
            low = mid + 1
    return -1  # Target tidak ditemukan
```

```python
def binary_search_recursive(arr, target, low, high):
    if low <= high:
        mid = (low + high) // 2

        # Jika target ditemukan
        if arr[mid] == target:
            return mid
        # Jika target lebih kecil dari mid
        elif arr[mid] > target:
            return binary_search_recursive(arr, target, low, mid - 1)
        # Jika target lebih besar dari mid
        else:
            return binary_search_recursive(arr, target, mid + 1, high)
    return -1  # Target tidak ditemukan
```

VARIABLE-SIZE DECREASE

Algoritma Euclidean GCD termasuk dalam kategori Variable Size Decrease, karena algoritma ini menerapkan GCD(a, b) = GCD(b, a % b), dimana besaran pengurangan a %b tidak selalu tetap atau merupakan fraksi konstan dari b; melainkan, nilainya bergantung pada nilai a dan b itu sendiri.

GCD(138, 78)	138 mod 78 = 60
GCD(78, 60)	78 mod 60 = 18
GCD(60, 18)	60 mod 18 = 6
GCD(18, 6)	18 mod 6 = 0
6	

VARIABLE-SIZE DECREASE

```python
def gcd_iterative(a, b):
    while b != 0:
        a, b = b, a % b
    return a
```

```python
def gcd_recursive(a, b):
    if b == 0:
        return a
    return gcd_recursive(b, a % b)
```

SPY X FAMILIY
CODE: BLUE

Menggunakan prinsip “Binary Search the Answer” (BSTA), dimana kita sudah mengetahui bahwa solusi bersifat monotomik (terus meningkat atau terus menurun).

Untuk sebuah agen dengan peringkat r, jika agen tersebut mengumpulkan n petunjuk, maka waktu yang diperlukan adalah:

$$t = r \cdot n^2$$

Misalkan pada waktu T, agen dengan peringkat r dapat mengumpulkan petunjuk sebanyak n dengan syarat:

$$r \cdot n^2 \leq T \quad \Longrightarrow \quad n \leq \sqrt{\frac{T}{r}}$$

SPY X FAMILIY
CODE: BLUE

Karena n harus bilangan bulat, maka jumlah maksimal petunjuk yang dapat dikumpulkan oleh agen tersebut adalah:

$$n = \left\lfloor \sqrt{\frac{T}{r}} \right\rfloor$$

Jika terdapat k agen dengan peringkat $r_1, r_2, \ldots, r_k$, maka total petunjuk yang dikumpulkan pada waktu T adalah:

$$\text{Total} = \sum_{i=1}^{k} \left\lfloor \sqrt{\frac{T}{r_i}} \right\rfloor$$

Kita mencari nilai minimum T sehingga:

$$\sum_{i=1}^{k} \left\lfloor \sqrt{\frac{T}{r_i}} \right\rfloor \geq \text{clue}$$

SPY X FAMILIY
CODE: BLUE

Misalkan:

$$\textbf{ranks} = [4,\ 2,\ 3,\ 1] \quad \text{dan} \quad \textbf{clue} = 10$$

Kita coba dengan $T = 16$ menit:

$$\text{Agen dengan } r = 4: \quad \left\lfloor \sqrt{\frac{16}{4}} \right\rfloor = \lfloor \sqrt{4} \rfloor = 2 \quad \text{petunjuk}$$

$$\text{Agen dengan } r = 2: \quad \left\lfloor \sqrt{\frac{16}{2}} \right\rfloor = \lfloor \sqrt{8} \rfloor = 2 \quad \text{petunjuk}$$

$$\text{Agen dengan } r = 3: \quad \left\lfloor \sqrt{\frac{16}{3}} \right\rfloor = \lfloor \sqrt{5.33} \rfloor = 2 \quad \text{petunjuk}$$

$$\text{Agen dengan } r = 1: \quad \left\lfloor \sqrt{\frac{16}{1}} \right\rfloor = \lfloor 4 \rfloor = 4 \quad \text{petunjuk}$$

Total petunjuk:

$$2 + 2 + 2 + 4 = 10 \quad \text{petunjuk}$$

Karena jumlahnya sama dengan **clue**, maka $T = 16$ adalah waktu minimum.

SPY X FAMILIY
CODE: BLUE

Misalkan:

$$\textbf{ranks} = [4,\ 2,\ 3,\ 1] \quad \text{dan} \quad \textbf{clue} = 10$$

Kita coba dengan $T = 16$ menit:

$$\text{Agen dengan } r = 4: \quad \left\lfloor \sqrt{\frac{16}{4}} \right\rfloor = \lfloor \sqrt{4} \rfloor = 2 \quad \text{petunjuk}$$

$$\text{Agen dengan } r = 2: \quad \left\lfloor \sqrt{\frac{16}{2}} \right\rfloor = \lfloor \sqrt{8} \rfloor = 2 \quad \text{petunjuk}$$

$$\text{Agen dengan } r = 3: \quad \left\lfloor \sqrt{\frac{16}{3}} \right\rfloor = \lfloor \sqrt{5.33} \rfloor = 2 \quad \text{petunjuk}$$

$$\text{Agen dengan } r = 1: \quad \left\lfloor \sqrt{\frac{16}{1}} \right\rfloor = \lfloor 4 \rfloor = 4 \quad \text{petunjuk}$$

Total petunjuk:

$$2 + 2 + 2 + 4 = 10 \quad \text{petunjuk}$$

Karena jumlahnya sama dengan **clue**, maka $T = 16$ adalah waktu minimum.

SPY X FAMILIY
CODE: BLUE

Diberikan **ranks** $= [4, 2, 3, 1]$, untuk setiap agen:

$$\text{Jumlah petunjuk} = \left\lfloor \sqrt{\frac{T}{r}} \right\rfloor .$$

Berikut adalah tabel yang menunjukkan jumlah petunjuk yang dikumpulkan masing-masing agen dan totalnya untuk beberapa nilai T:

T	$\left\lfloor \sqrt{\frac{T}{4}} \right\rfloor$	$\left\lfloor \sqrt{\frac{T}{2}} \right\rfloor$	$\left\lfloor \sqrt{\frac{T}{3}} \right\rfloor$	$\left\lfloor \sqrt{\frac{T}{1}} \right\rfloor$	Total
0	0	0	0	0	0
1	0	0	0	1	1
4	1	1	1	2	5
9	1	2	1	3	7
16	2	2	2	4	10
25	2	3	2	5	12
36	3	4	3	6	16

Keterangan:

- Pada $T = 0$, tidak ada petunjuk yang dikumpulkan.
- Pada $T = 1$, hanya agen dengan peringkat 1 yang mengumpulkan 1 petunjuk.
- Pada $T = 16$, total petunjuk yang terkumpul adalah 10, yang sesuai dengan target **clue** $= 10$ pada contoh sebelumnya.

SPY X FAMILIY CODE: BLUE

```python
import math

def code_blue(ranks, clue):
    l, r = 0, max(ranks) * clue * clue
    ans = 0

    while l < r:
        mid = (l + r) // 2
        jum = 0

        for rank in ranks:
            jum += int(math.sqrt(mid / rank))

        if jum >= clue:
            r = mid
        else:
            l = mid + 1

        if jum == clue:
            ans = mid

    return ans
```

THANK YOU

FOR YOUR NICE ATTENTION

link canva: https://www.canva.com/design/DAGiIKI7GW8/wXE9kZO88PUo-iWzQUyCAQ/edit?utm_content=DAGiIKI7GW8&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton