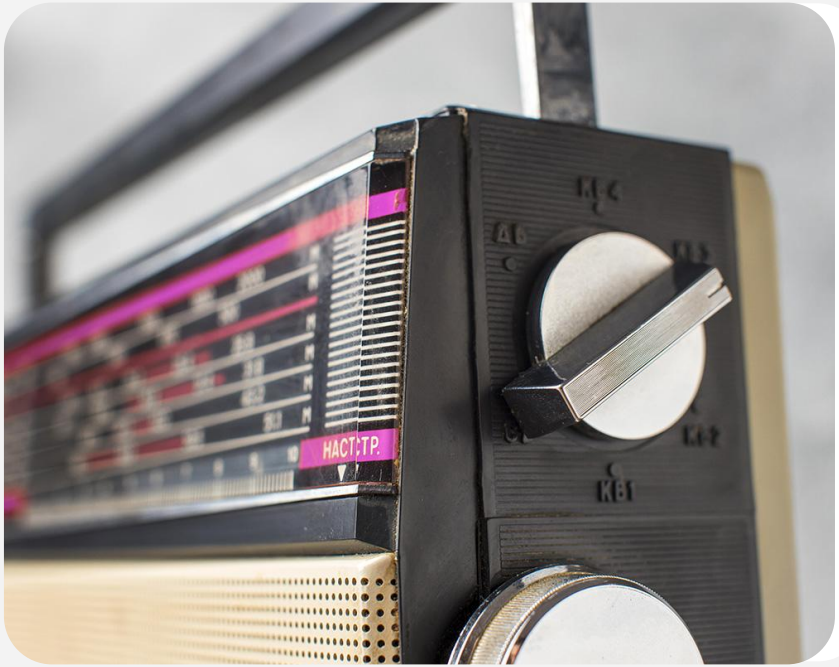
Case Study Berpikir Komputasional

https://youtu.be/HsiOJ4hSA-k

Edgrant H. S. (2206025016)



CASE STUDY 6

CASE STUDY

Background:

With the rise of e-commerce, companies are exploring autonomous drone delivery systems to improve delivery speed and reduce costs. However, operating drones in urban or rural environments presents challenges such as avoiding obstacles, adapting to weather conditions, and managing battery life.

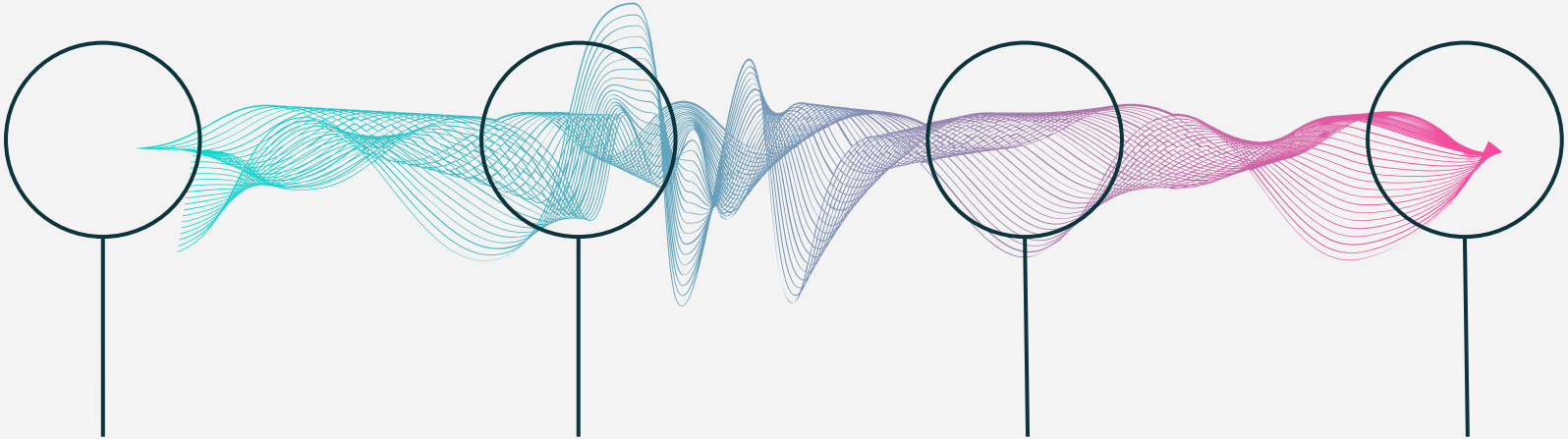
Problem Statement:

Design a computational solution to optimize drone delivery routes. The solution should consider:

1. Real-time data (e.g., weather conditions, air traffic, and battery levels).
2. Obstacle avoidance and safe navigation.
3. Delivery priorities (e.g., urgent packages).
4. Energy efficiency and battery life optimization.

Your goal is to create a system that dynamically plans and adjusts delivery routes for autonomous drones based on real-time data while balancing delivery speed, energy efficiency, and safety. Additionally, your solution must incorporate at least one classic algorithm we have learned, such as linear search, finding extreme values, greedy algorithm, or brute force.

4 PILLAR OF COMP. THINKING



Description

Pattern Recognition

Data Representation and Abstraction

Algorithm

CASE DESCRIPTION

Problem
Masalah utama dalam kasus ini adalah optimasi rute pengiriman drone yang harus menyeimbangkan kecepatan pengiriman, efisiensi energi, dan keselamatan dalam berbagai kondisi lingkungan.

Key Variables
- Data real-time (kondisi cuac, level baterai)
- Hindaran rintangan (bangunan, pohon, dan objek lain)
- Prioritas pengiriman (paket yang mendesak vs. reguler)
- Efisiensi energi (pengelolaan baterai dan jarak tempuh optimal)

Objectives
- Penyesuaian rute drone berdasarkan data real-time, prioritas pengiriman, dan hindaran rintangan.
- Mengoptimalkan pengiriman agar memiliki pemilihan yang optimal antara kecepatan dan efisiensi energi.

Challenges
- Perubahan kondisi cuaca yang dapat memengaruhi rute dan efisiensi baterai.
- Menghindari rintangan dalam lingkungan yang dinamis dan kompleks.
- Menentukan strategi optimal untuk menyeimbangkan pengiriman rute dan konsumsi daya.

PATTERN RECOGNITION

1. Pola Cuaca vs. Energi

- Kondisi cuaca (angin, hujan, suhu) mempengaruhi kecepatan dan efisiensi baterai, contohnya cuaca buruk memaksa drone mengambil rute lebih panjang atau mengurangi kecepatan untuk hemat energi.

2. Pola Rintangan

- Rintangan statis (gedung, pohon) membentuk jalur tetap, sedangkan dinamis (burung, drone lain) membutuhkan respons real-time.

3. Pola Prioritas Pengiriman

- Paket prioritas memiliki permintaan yang perlu dikirim secara mendesak dan biasanya banyak diperlukan pada waktu tertentu (contoh pagi/sore hari), memerlukan perencanaan alokasi pada waktu tersebut.
- Pengiriman reguler mengikuti distribusi merata, tetapi memiliki slot prioritas rendah.

4. Pola Konsumsi Baterai

- Keterbatasan baterai membatasi jarak tempuh dan memerlukan integrasi stasiun pengisian ke rute.
- Penambahan/pembesaran baterai menambah berat dari drone yang mengakibatkan baterai lebih tidak efisien

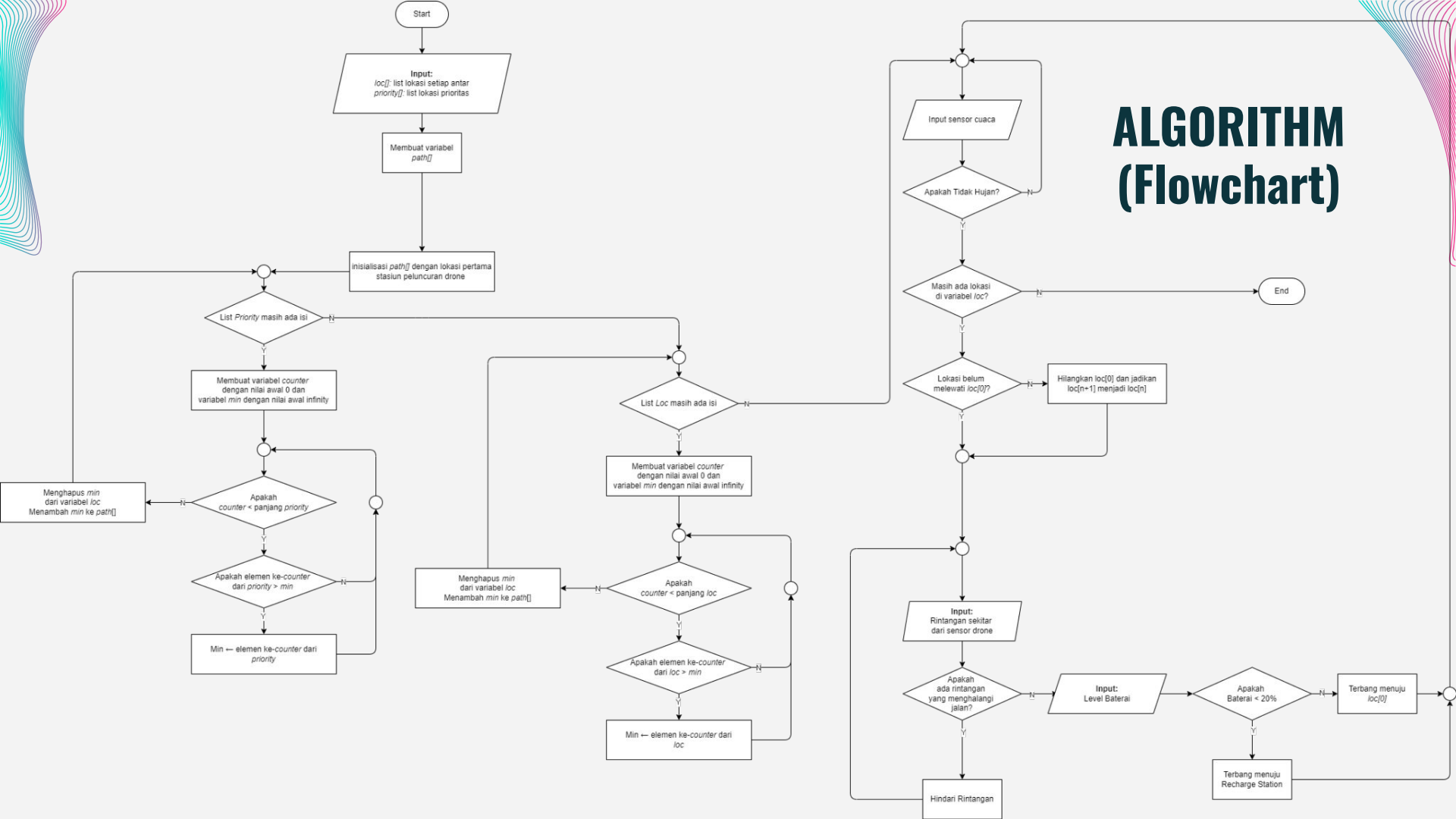
DATA REPRESENTATION

- **Data Pengiriman:** Lokasi pengiriman direpresentasi dengan koordinat x dan y.
- **Data Prioritas Pengiriman:** Lokasi pengiriman prioritas direpresentasi dengan koordinat x dan y yang disimpan berbeda dengan lokasi pengiriman non prioritas
- **Data Cuaca:** Cuaca direpresentasi secara real-time, seperti kecepatan angin, arah angin, curah hujan, dan visibilitas.
- **Data Baterai:** Baterai drone direpresentasi secara real-time dan berbentuk persentase dari seluruh kapasitas baterai.
- **Representasi Rintangan:** Rintangan direpresentasi secara real-time dan berupa boolean (yes/no) terkait apakah terdapat rintangan yang menghalangi drone.

DATA ABSTRACTION

- **Abstraksi Rute** (titik kirim, titik prioritas, baterai). Rute berupa list dengan elemen selanjutnya merupakan node pengiriman selanjutnya.
- **Model cuaca** (kategori: cerah, hujan). Cuaca merupakan boolean dengan true berupa hujan dan false berupa tidak hujan.
- **Model Rintangan** (sensor rintangan). Rintangan merupakan boolean dengan true berupa terdapat rintangan dan false berupa tidak ada rintangan.

ALGORITHM (Flowchart)



ALGORITHM (Pseudocode)

```
// Input data
INPUT locationList, priorityList

// Initialize variables
CREATE path WITH dengan elemen pertama sebagai drone launch location

// Membuat priority di awal list
WHILE priorityList IS NOT EMPTY DO
    CREATE counter = 0
    CREATE minElement = INFINITY

    WHILE counter < LENGTH(priorityList) THEN
        IF priorityList[counter] < minElement THEN
            minElement = priorityList[counter]
        END IF
    END WHILE

    APPEND minElement TO path
    REMOVE minElement FROM priorityList
END WHILE
```

ALGORITHM (Pseudocode)

```
// Menambah location setelah priority list
WHILE locationList IS NOT EMPTY DO
    CREATE counter = 0
    CREATE minElement = INFINITY

    WHILE counter < LENGTH(locationList) THEN
        IF locationList[counter] < minElement THEN
            minElement = locationList[counter]
        END IF
    END WHILE

    APPEND minElement TO path
    REMOVE minElement FROM locationList
END WHILE
```

ALGORITHM (Pseudocode)

```
// Loop penerbangan drone
WHILE locationList IS NOT EMPTY DO
    // Check for location
    IF lokasi > path[0]
        Delete path[0] // dan juga shift seluruh path[n+1] ke path[n]
     END IF

    // Check for rain
    INPUT isRaining
    WHILE isRaining THEN
        WAIT untuk tidak hujan
    END WHILE

    // Check for obstacles
    INPUT obstaclesPresent
    WHILE obstaclesPresent THEN
        AVOID obstacle
    END WHILE
                                    // Terbang Menuju lokasi
                                    INPUT battery
                                    IF battery < 20%
                                        RECHARGE at charging station
                                     ELSE
                                        MOVE toward path[0]
                                     END IF
                                END WHILE
```

ALGORITHM (Calculate Big O)

TSP: $3 + (3+n)n = n^2 + 3n + 3$

Drone: $(1 + 1 + 1 + k)n/v = 3n/v + kn/v$

- n: jumlah node
- v: kecepatan drone
- k: koefisien obstacle

Total: $n^2 + 3n + kn/v + 3n/v + 3$

Simplifikasi mengambil komponen paling signifikan: $O(n^2)$

Polinomial kuadratik