CSC4202-G6:
DESIGN AND ANALYSIS OF ALGORITHMS
(SAFE EVACUATION ROUTE IN POST-LANDSLIDE)

1. ALISYA ATHIRAH BINTI MOHD HUZZAINNY 211175
2. ADRIANA HUMAYRA BINTI SHALIZAN 213056
3. NURHAZWANI BINTI MUHAMMAD RADHI 213515

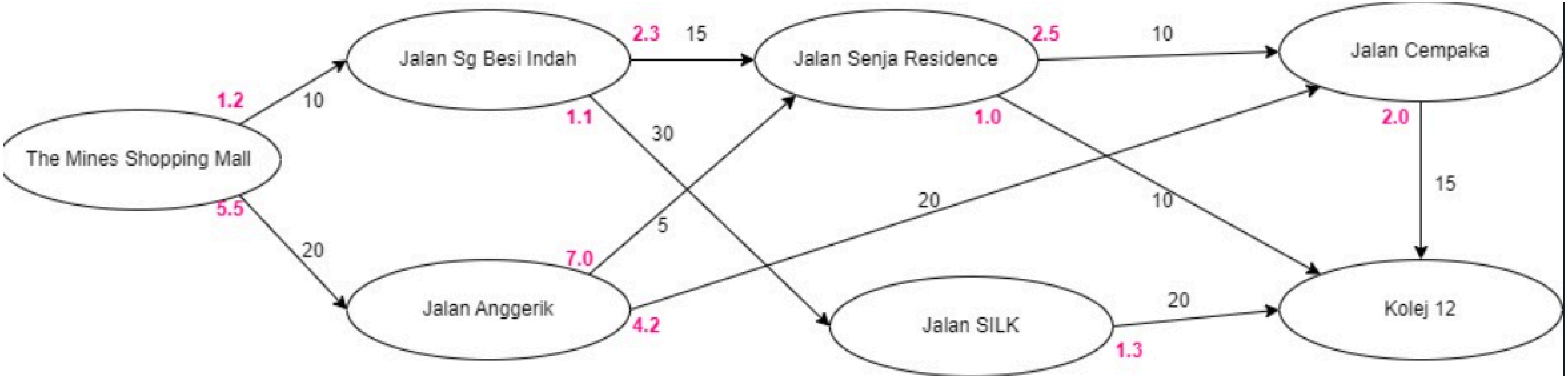
Scenario & Challenge

A catastrophic **landslide** struck around **The Mines Shopping Mall** in Seri Kembangan, resulting in significant **structural damage and blocking several pathways**. This unexpected disaster has trapped numerous individuals inside, including ten Universiti Putra Malaysia (UPM) students. These students must find **safe routes to return** to their college dormitory, **Kolej 12,** on the university campus. The **usual fastest routes are now unsafe**, so they need to consider alternative paths that offer the shortest distance while ensuring their safety.

The **primary challenges** are:

- navigating blocked roads and paths
- dealing with the dynamically changing conditions caused by the landslide
- ensuring the safety of the individuals

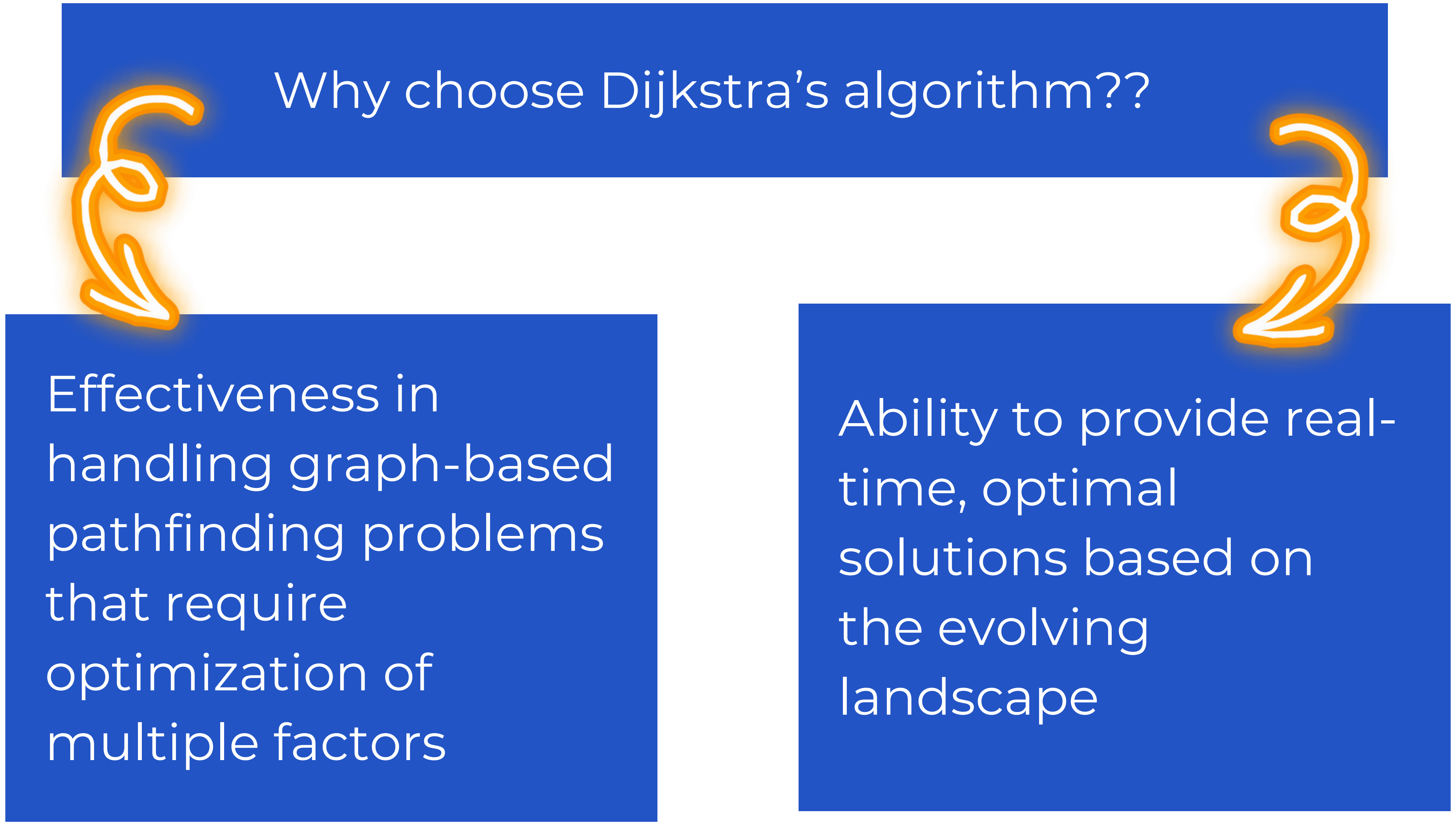
Goal



To develop an algorithm that helps the students to find the **shortest and safest routes** to reach Kolej 12. Compare each **route distance and safety score** from each Jalan stated in the illustration above. The starting location is The Mines Shopping Mall.

Algorithm Chosen

Algorithm Paradigm: The paradigm used for **Dijkstra's algorithm** is greedy algorithm.



Why choose Dijkstra’s algorithm??

Effectiveness in handling graph-based pathfinding problems that require optimization of multiple factors

Ability to provide real-time, optimal solutions based on the evolving landscape

Pseudocode

```
function findSafestPath(start: Location, destination: Location)
    start.minDistance = 0
    queue = new PriorityQueue()
    queue.add(start)

    while queue is not empty
        current = queue.poll()

        for each path in current.paths
            next = path.target
            weight = path.distance + path.safetyScore
            distanceThroughCurrent = current.minDistance + weight

            if distanceThroughCurrent < next.minDistance
                queue.remove(next)
                next.minDistance = distanceThroughCurrent
                next.previous = current
                queue.add(next)

    printPaths(destination)
```

- Uses a priority queue to always expand the least-cost (safest) node first.
- Updates the shortest known distance to each neighboring location and tracks the path taken.

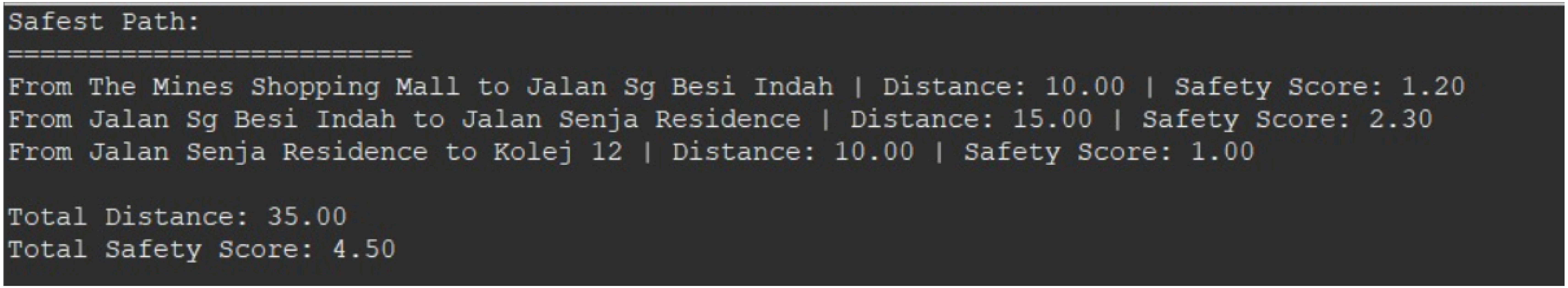
```
function getPathTo(target: Location) -> List of Location
    path = new List()
    location = target
    while location is not null
        path.add(location)
        location = location.previous
    reverse(path)
    return path
```

- Reconstructs the path from the destination back to the start by following the previous pointers.
- Provides the sequence of locations that form the safest path.

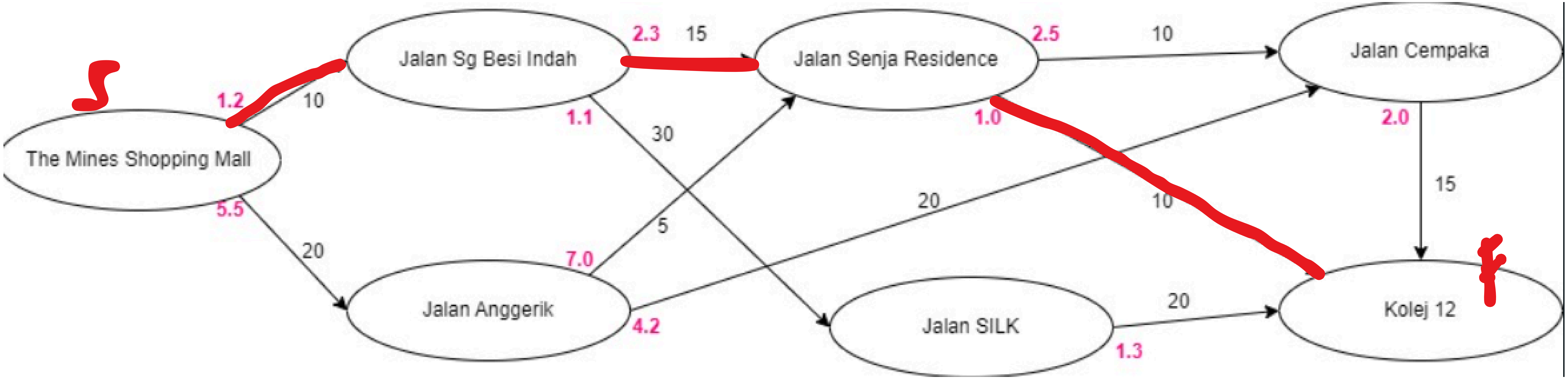
Pseudocode

```
function main()
    mall = new Location("The Mines Shopping Mall")
    kolej12 = new Location("Kolej 12")
    checkpoint1 = new Location("Jalan Sg Besi Indah")
    checkpoint2 = new Location("Jalan Anggrerik")
    checkpoint3 = new Location("Jalan Senja Residence")
    checkpoint4 = new Location("Jalan SILK")
    checkpoint5 = new Location("Jalan Cempaka")

    addPaths(mall, [
        new Path(checkpoint1, 10, 1.2),
        new Path(checkpoint2, 20, 5.5)
    ])
    addPaths(checkpoint1, [
        new Path(checkpoint3, 15, 2.3),
        new Path(checkpoint4, 30, 1.1)
    ])
    addPaths(checkpoint2, [
        new Path(checkpoint3, 5, 7.0),
        new Path(checkpoint5, 25, 4.2)
    ])
    addPaths(checkpoint3, [
        new Path(kolej12, 10, 1.0),
        new Path(checkpoint5, 10, 2.5)
    ])
    addPaths(checkpoint4, [
        new Path(kolej12, 20, 1.3)
    ])
    addPaths(checkpoint5, [
        new Path(kolej12, 15, 2.0)
    ])

    findSafestPath(mall, kolej12)
```

Output



```
Safest Path:
=========================
From The Mines Shopping Mall to Jalan Sg Besi Indah | Distance: 10.00 | Safety Score: 1.20
From Jalan Sg Besi Indah to Jalan Senja Residence | Distance: 15.00 | Safety Score: 2.30
From Jalan Senja Residence to Kolej 12 | Distance: 10.00 | Safety Score: 1.00

Total Distance: 35.00
Total Safety Score: 4.50
```



Correctness Analysis

Initialization

start.minDistance = 0

- Initialize priority queue and add the start location to it.
- Ensures the search begins from the start location with the appropriate initial distance.

Recurrence Relation

If total_distance(v) is less than minDistance(v), then: minDistance(v)=total_distance(v)

- Iterate and update the shortest path estimate for each location based on the current shortest known paths.
- Ensures that the shortest known path to each location is continually improved.

Optimization Function

minDistance(v)=min(minDistance(v),minDistance(u)+w)

- Expanding the smallest minDistance ensures correct order processing.
- This guarantees the shortest path from The Mines Shopping Mall to Kolej 12.

Algorithm Analysis cont..

- Vertices (**V**): The number of locations.
- Edges (**E**): The number of paths.
- The factor **logV** comes from the operations on the priority queue.

Best-case analysis

- Dijkstra's algorithm needs to process each vertex and edge at least once, and the priority queue operations (insertions and deletions) still take O(logV) time.
- Time Complexity: O((V+E)logV)

Average-case analysis

- Dijkstra's algorithm will typically process each vertex and edge.
- Time Complexity: O((V+E)logV)

Worst-case analysis

- Every vertex and every edge needs to be processed. Each insertion and deletion operation in the priority queue takes O(logV) time.
- Time Complexity: O((V+E)logV)

Thank you