

**GROUP PROJECT**

**Learning outcomes:**

1. Search for and organize information in the form of an algorithm for a computer problem as the following: (LL)
   1. Analyse the problem to suggest a solution by choosing a suitable algorithm design technique.
   2. Develop skills to reason about and prove properties of algorithms such as their correctness and running time.

**Instructions:**

1. General instructions:
   1. Create an original scenario (you can be inspired by examples on the internet or used in the course, but the detailing of the story should be created fresh by your team of 3 people) that requires an optimal solution.
   2. Explain why finding an optimal solution for this scenario is important.
   3. Review the suitability of sorting, DAC, DP, greedy and graph algorithms as a solution paradigm for the chosen problem by stating their strengths and weaknesses.
   4. Design the algorithm to solve the problem and explain the idea of your algorithm paradigm by emphasising which part needs **recurrence** and the function for the **optimization**.
   5. Define the algorithm specification.
   6. Develop a program Java language.
   7. Provide an analysis of the **algorithm’s correctness** as well as **time complexity** (best, average and worst time) by using asymptotic notation.
   8. Develop an online portfolio (using google sites or github or google colab or any suitable tool) with the following steps/content:
      1. Illustrate the problem.
      2. Explain your algorithm paradigm and show the **pseudocode**. You may provide your code in the portfolio if you wish.
      3. Demonstrate your program and describe the output.
      4. Describe the algorithm analysis.
   9. Deliver a presentation in week 14.
   10. Submit the following through Putrablast before the presentation
       1. Link to your online portfolio
       2. A zip file of your codes
       3. Filled project progress (refer APPENDIX)
2. Please make sure each member’s workload is fairly distributed and good project management is exercised. The weight of the project is 20%. Peer-based evaluation (5 marks) will be utilised besides the evaluations through filled progress monitoring (10 marks), online portfolio in github (50 marks) and presentation (15 marks).
3. Follow the following algorithm specification steps:

* Problem definition based on the chosen scenario (tell your story by choosing a geographical setting, type of disaster, damage impact, highlight the importance of AAD in the scenario and provide illustrations, state the goal and expected output to support the decision making)
* Development of a model for the chosen scenario (state the data type, state the objective function and constraints, provide examples and other requirements based on the scenario such as objective, space, time or value constraints.
* Specification of an Algorithm (state which topic and algorithm have you selected and why, include comparison of several other options and discuss the suitability of your proposed solution)
* Designing an Algorithm (provide a pseudocode and/or flowchart and use illustrations to help you)
* Checking the correctness of an Algorithm (asymptomatic, recurrence)
* Analysis of an Algorithm (growth of function for worst, best, average analysis)
* Implementation of an Algorithm.
* Program testing (provide a demo based on your story of the chosen scenario)
* Documentation through online portfolio.

1. Tips for completing the project
   1. Understanding of the AAD topics including examples of problems and list of algorithms that each topic covers is a MUST to ensure you can design the solution based on the instructions given.
   2. The “How Might We (HMW) technique” launches brainstorms by asking questions that seed your idea. Eg, HMW maximize the profit from building on this location by optimising the selection from a list of property options? From this question, your group may discuss potential situations and solutions. Then, discuss the possible algorithms to be used for this problem.
   3. You need to work collaboratively and be encouraged to use various materials around you as your reference (please make sure you include a good bibliography list in your documentation). During discussion, you may use ideas reviewing techniques such as identifying the “Pluses, Potential, Concerns, Options (PPCO)”.

Pluses: What are (at least) three things you like about the idea?

Potentials: What are (at least) three good things that might result if the idea were implemented?

Concerns: What are some concerns you have about the idea (phrased as a question starting with “How to…” or “How might…”)

Options or Overcome the concerns: What are some ideas you have for how to fix the concerns you just noted?

Or, you may use the Strength, Weakness, Opportunities, Threats (SWOT) technique.

* 1. Time management and each member’s dedication is the key to the group’s success.
  2. Use your creativity and critical thinking skills to design the algorithms and communicate it well in written and verbal format.

APPENDIX

**Initial Project Plan (week 10, submission date: 31 May 2024)**

| Group Name | First Call |
| --- | --- |
| Members | | Name | Email | Phone number | | --- | --- | --- | | NAME 1 | CHNG JUN BIN | 011-73190784 | | NAME 2 | CHIN WEN PING | 014-3691032 | | NAME 3 | LIM CHIEW FUNG | 017-4094911 | |
| Problem scenario description | Disaster Response Optimization in Kelantan Each year, the monsoon rains lead to significant flooding in Kelantan, resulting in widespread damage. The heavy rainfall causes rivers to overflow, submerging homes, roads, and farmlands. |
| Why it is important | Efficient and swift disaster response to minimize loss of life and property and ensure the rapid and safe evacuation of residents is paramount to reducing loss of life and preventing injuries. |
| Problem specification | In Kelantan, Malaysia, severe flooding prompts urgent evacuation planning. Data on coordinates, demographics, and infrastructure is crucial.  **Objective:** Minimize response and evacuation times **Constraints:** Time, geography, and shelter capacities.  Greedy algorithms determine efficient evacuation routes for rapid disaster response. |
| Potential solutions | Greedy algorithms will be used for real-time evacuation planning, providing efficient routes for quick decision-making. |
| Sketch (framework, flow, interface) |  |

**Project Proposal Refinement (week 11, submission date: 7 June 2023)**

| Group Name |  |
| --- | --- |
| Members | | **Name** | **Role** | | --- | --- | | CHNG JUN BIN | Problem Analyst | | CHIN WEN PING | Algorithm Designer | | LIM CHIEW FUNG | Implementation Specialist | |
| Problem statement | Design an optimal solution for evacuation planning in Kelantan, which is prone to severe flooding during hurricane season. The goal is to minimize response time and ensure efficient evacuation to minimize loss of life and property. |
| Objectives | 1. Develop a model for evacuation planning. 2. Design and implement algorithms to solve evacuation problems. 3. Analyze the performance and correctness of the implemented algorithms |
| Expected output | Efficient evacuation routes for residents.  A functioning Java program that implements the algorithms.  Analysis of the algorithm's correctness and time complexity. |
| Problem scenario description | Kelantan, a state located in the northeastern part of Peninsular Malaysia, is known for its rich cultural heritage and agriculture-based economy. However, Kelantan frequently experiences severe flooding during the monsoon season, particularly from November to January. |
| Why it is important | Efficient disaster response is crucial to minimizing the adverse effects of flooding in Kelantan. Well-planned evacuation routes save lives. |
| Problem specification | **Data needed:** Geographic coordinates of affected areas.  Population data and demographics.  Infrastructure details (e.g., roads, shelters).  **Objective Function:**  Evacuation Planning: Minimize evacuation time and ensure the safety of residents. |
| Potential solutions | **Greedy Algorithms**  Strengths: Efficient for making real-time decisions; simple to implement.  Weaknesses: May not always provide the globally optimal solution.  Application: Evacuation planning to quickly determine the best routes. |
| Sketch (framework, flow, interface) |  |
| Methodology | | Milestone | Time | | --- | --- | | <eg: scenario refinement> | wk10 | | <eg: find example solutions and suitable algorithms. Discuss in group why that solution and the example problems relate to the problem in the project> | wk11 | | <eg: edit the coding of the chosen problem and complete the coding. Debug> | wk12 | | <eg: conduct analysis of correctness and time complexity > | wk13 | | <prepare online portfolio and presentation> | wk14 | |

**Project Progress (Week 10 – Week 14)**

| **Milestone 1** |  |
| --- | --- |
| **Date (week)** | Week 10 - Week 11 |
| **Description/**  **sketch** | **Task:** Scenario refinement and algorithm selection.  **Details:** Finalized the problem scenario based on feedback and additional research. Evaluated potential algorithms (Greedy algorithms for evacuation planning). Discussed and finalized the choice of algorithms. |
|
| **Role** | **Member 1 (Chng Jun Bin)**: Gather additional data on geographic coordinates and population demographics, conduct a literature review on similar disaster response optimization problems.  **Member 2 (Chin Wen Ping)**: Refined the objective functions and constraints for evacuation, analyzed the strengths and weaknesses of Greedy algorithms.  **Member 3 (Lim Chiew Fung)**: Updated the project plan and timeline, facilitated group discussion, and documented the decision-making process. |

| **Milestone 2** |  |
| --- | --- |
| **Date (Wk)** | Week 12 |
| **Description/**  **sketch** | **Task:** Coding and initial debugging.  **Details:** Developed initial versions of the algorithms in Java. Completed coding and began debugging the programs. |
|
| **Role** | **Member 1 (Chng Jun Bin)**: Implemented the Greedy algorithm for evacuation planning.  **Member 2 (Chin Wen Ping)**: Implemented the Greedy algorithm for evacuation planning.  **Member 3 (Lim Chiew Fung)**: Performed initial debugging and testing. |

| **Milestone 3** |  |
| --- | --- |
| **Date (Wk)** | Week 13 - Week 14 |
| **Description/**  **sketch** | **Task:** Analysis and final preparation.  **Details:** Analyzed the correctness of the implemented algorithms. Calculated the time complexity for best, average, and worst-case scenarios. Created an online portfolio showcasing the project details, code, and analysis. Prepared a presentation for the final submission. |
|
| **Role** | **Member 1 (Chng Jun Bin)**: Verified the correctness of the Greedy algorithm, developed the content for the online portfolio.  **Member 2 (Chin Wen Ping)**: Verified the correctness of the Greedy algorithm, prepared the presentation slides.  **Member 3 (Lim Chiew Fung)**: Compiled the time complexity analysis, coordinated the final review and submission. |

**Problem Scenario Description:**

Disaster Response Optimization in Kelantan

Kelantan, Malaysia, frequently experiences severe flooding during the monsoon season, resulting in significant damage and disruption. The heavy rains cause rivers to overflow, submerging homes, roads, and farmlands. Efficient and swift evacuation planning is critical to minimize loss of life and ensure the safety of residents.

**Problem Statement:**

Design an optimal solution for evacuation planning in Kelantan, which is prone to severe flooding during hurricane season. The goal is to minimize response time and ensure efficient evacuation to minimize loss of life and property.

**Objective Function:**

Evacuation Planning: Minimize evacuation time and ensure the safety of residents.

**Importance of Optimal Solution:**

Efficient disaster response and evacuation planning are vital for saving lives and minimizing property damage. Optimal evacuation routes ensure that residents can be moved quickly and safely to shelters, reducing the risk of fatalities and injuries.

**Expected Output:**

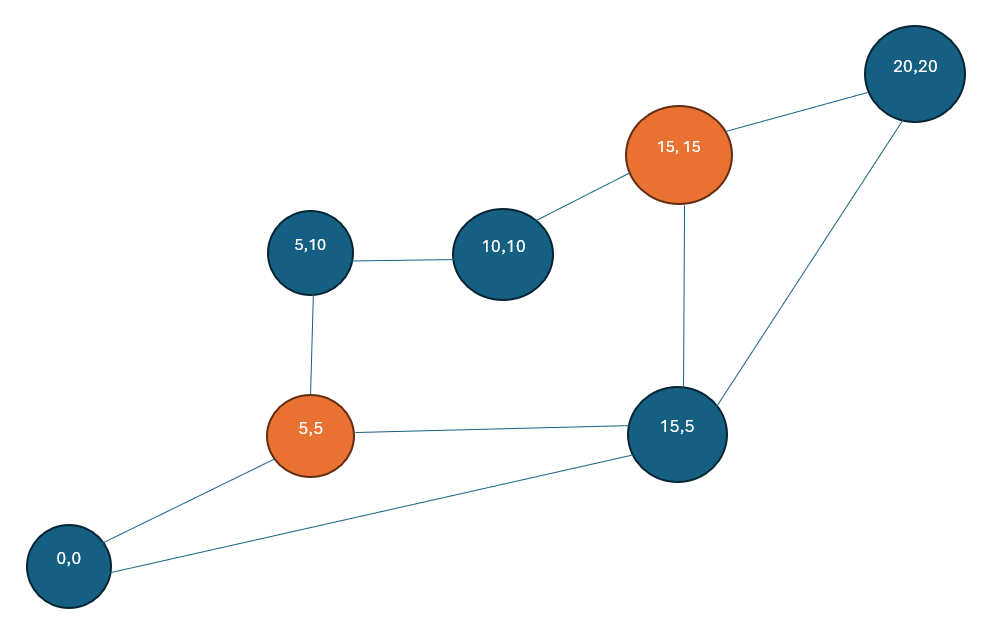
1. Efficient evacuation routes for residents.
2. A functioning Java program that implements the algorithms.
3. Analysis of the algorithm's correctness and time complexity.

**Algorithm Paradigm Suitability Review:**

| Algorithm | Strengths | Weakness |
| --- | --- | --- |
| Sorting | It is useful for ordering evacuation points based on proximity or urgency. | Insufficient for dynamic route optimization. |
| Divide and Conquer (DAC) | Breaks down large problems into smaller subproblems. | May not consider global optimality in route planning. |
| Dynamic Programming (DP) | Ideal for problems with overlapping subproblems and optimal substructure. | Computationally expensive and requires careful problem formulation. |
| Greedy Algorithms | Efficient for making real-time decisions and simple to implement. | May not always yield the globally optimal solution. |
| Graph Algorithms | Suitable for representing evacuation routes and networks. | Requires modifications to handle multiple objectives and real-time constraints. |

We choose a greedy algorithm as our algorithm design for evacuation planning in Kelantan, Malaysia. This is because it provides real-time decision making of the best routes and is simple to implement.

**Illustration of Evacuation Planning Map for Affected Area:**



**Java code for Evacuation Planning using Greedy Algorithm:**

import java.util.\*;

class Coordinate {

int x, y;

Coordinate(int x, int y) {

this.x = x;

this.y = y;

}

@Override

public String toString() {

return "(" + x + ", " + y + ")";

}

}

class Route {

Coordinate start, end;

int time;

Route(Coordinate start, Coordinate end, int time) {

this.start = start;

this.end = end;

this.time = time;

}

@Override

public String toString() {

return "Route from " + start + " to " + end + " takes " + time + " time units.";

}

}

public class EvacuationPlanning {

private List<Coordinate> shelters;

private int[][] roadConditions;

private Coordinate[] affectedAreas;

public EvacuationPlanning(List<Coordinate> shelters, int[][] roadConditions, Coordinate[] affectedAreas) {

this.shelters = shelters;

this.roadConditions = roadConditions;

this.affectedAreas = affectedAreas;

}

public void planEvacuation() {

for (Coordinate area : affectedAreas) {

List<Route> allRoutes = findAllRoutes(area);

for (Route route : allRoutes) {

System.out.println(route);

}

Route bestRoute = findShortestRoute(allRoutes);

System.out.println("Chosen " + bestRoute + "\n");

}

}

private List<Route> findAllRoutes(Coordinate area) {

List<Route> allRoutes = new ArrayList<>();

for (Coordinate shelter : shelters) {

int time = calculateRouteTime(area, shelter);

allRoutes.add(new Route(area, shelter, time));

}

return allRoutes;

}

private Route findShortestRoute(List<Route> routes) {

return routes.stream().min(Comparator.comparingInt(route -> route.time)).orElse(null);

}

private int calculateRouteTime(Coordinate start, Coordinate end) {

// Use Manhattan distance as a base time calculation

int baseTime = Math.abs(start.x - end.x) + Math.abs(start.y - end.y);

// Add an additional time factor to differentiate similar distances

Random rand = new Random();

int additionalTime = rand.nextInt(10); // Add a random factor between 0 and 9

return baseTime + additionalTime;

}

public static void main(String[] args) {

// Initialize coordinates and conditions

List<Coordinate> shelters = Arrays.asList(new Coordinate(0, 0), new Coordinate(10, 10), new Coordinate(20, 20), new Coordinate(5, 10), new Coordinate(15, 5));

int[][] roadConditions = new int[25][25]; // Placeholder, not used in current implementation

Coordinate[] affectedAreas = {new Coordinate(5, 5), new Coordinate(15, 15)};

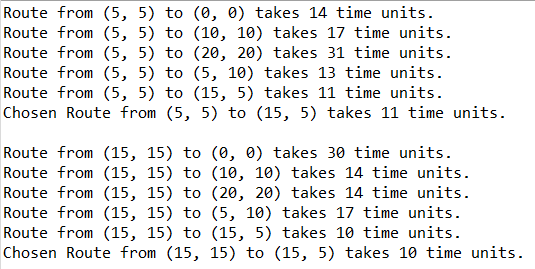
EvacuationPlanning planning = new EvacuationPlanning(shelters, roadConditions, affectedAreas);

planning.planEvacuation();

}

}

Output:



**Algorithm Analysis**

Correctness:

The Greedy algorithm efficiently determines the best evacuation routes based on real-time conditions. It ensures that residents are evacuated to the nearest shelters quickly and safely.

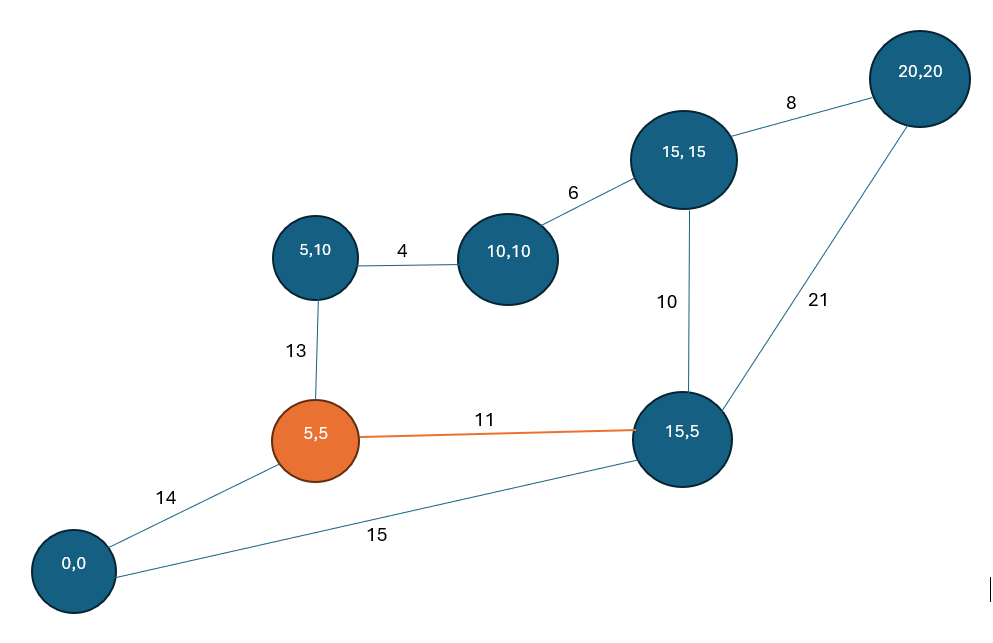
**Time Complexity:**

Best Case: O(m log m) for finding the nearest shelter, where m is the number of shelters.

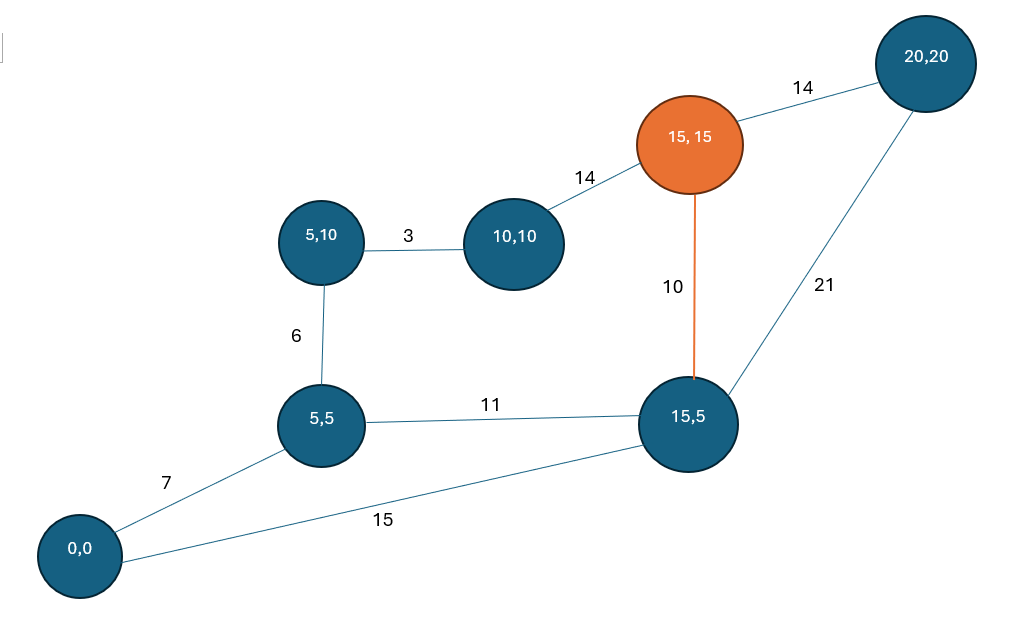
Average Case: O(m log m)

Worst Case: O(m log m)

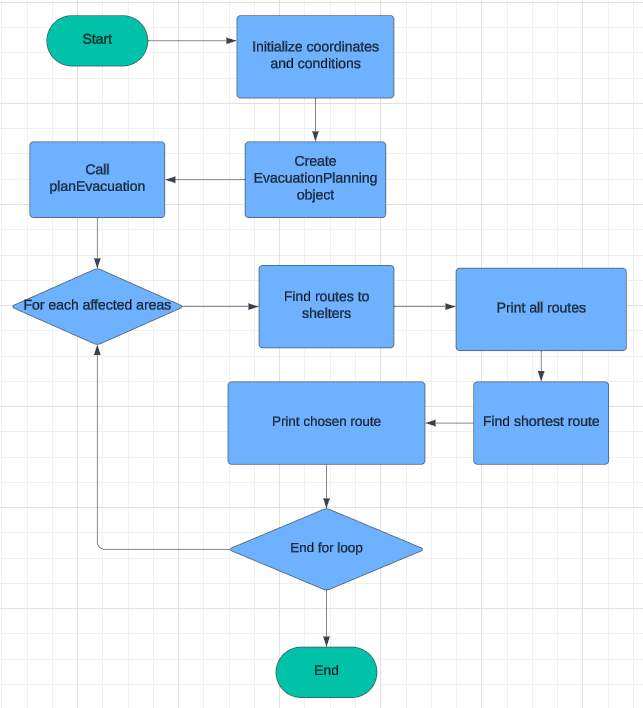
**Program Demonstration:**

**When affected area is 5, 5**

**When affected area is 15, 15**



**Flowchart:**

****