German University in Cairo
Department of Computer Science
Dr. Mohamed Khalgui

Introduction to Artificial Intelligence, Winter Term 2024
Project 1: Package Delivery

Due Saturday, December 8th by 23:59

Project Description In this project, you will be implementing a search agent that responsible for managing delivery of several packages from some stores to customer locations via a group of trucks. The goal is to deliver all packages to their targets. Additionally, we need to minimise delivery time by spending less time in traffic.

The city is assumed to be a grid with distributed storage locations and distributed customer destinations. Streets (grid-lines) connect these locations and each street has a traffic level. Some of the roads are blocked due to construction and there are also several underground tunnels connecting parts of the city that have lower traffic. The time it takes to travel this street is proportional to its traffic level. We assume that the products are available in all these M stores and the objective is to distribute while reducing the delivery cost without any delay.

Every morning, M distribution agents (each for each store) are responsible for distributing N products to N different locations in the city, which we assume to have a rectangular grid shape $m \times n$.. The coordinates of any destination are denoted as (X,Y). In the example below, the coordinates of destination $D1$ are (1,4), $D2$ are (5,2) and $D3$ are (7,3). To go from one location to another, the delivery agent follows a path which is a sequence of grid segments/tunnels. Taking a segment has a cost equal to the degree (an integer from 1 to 4) of the traffic jam at that time t. The cost of taking a tunnel is always equal to the Manhattan distance between its two entrances.

The objective of this project is to determine which trucks deliver which packages and to find the best path for each truck to take while reducing delivery costs and assisting each customer within a reasonable time.

We wish to develop an application based on the different search strategies seen in your course to find at a particular time t the different paths leading each agent from the storage stores to the first closest destination, then the 2nd destination, up to the N-th destination. The agent will choose the least expensive path to go to all destinations. In this application, we compute the path to attend each destination for each agent and we keep the lowest. Each agent will deliver the products with the lowest costs to attend the related destinations compared with the other agents.

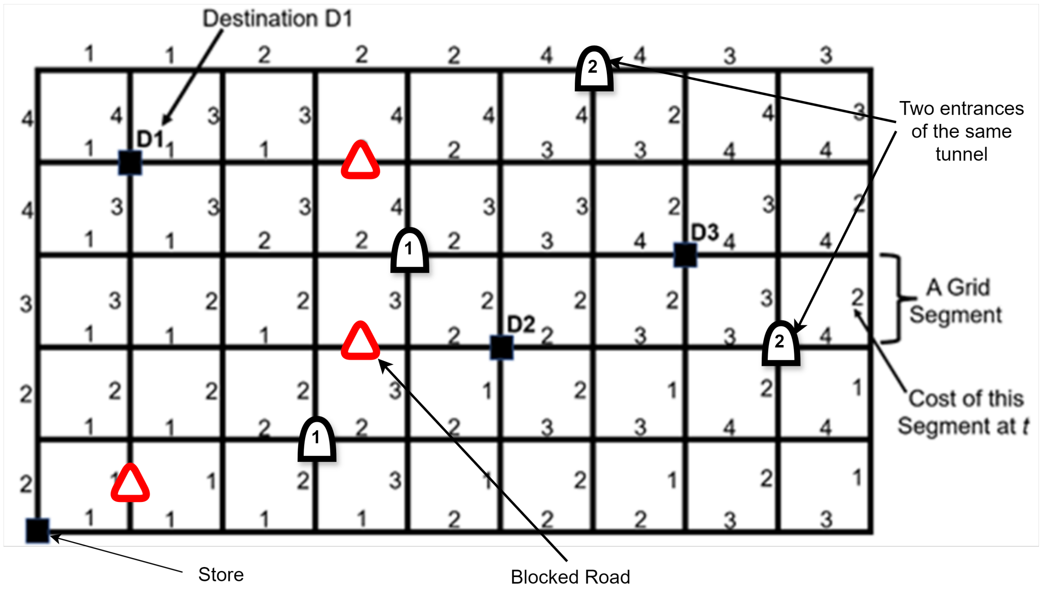
For simplicity, you can assume once a delivery has been made, the truck immediately returns to the store and can now make a new delivery

The grid elements are restricted to the following:

a) Storage stores S_i in the location (x_i, y_i): These are the starting points for the distribution truck, where products are sorted before delivery.

b) Distribution Trucks T_i : This is the truck starting from store S_i that delivers products to destinations.

c) Destination D_i in the location (x_i, y_i): The customer receives the package from the delivery truck.



d) Obstacles O_i: which exist on some roads segments making it impossible for trucks to use them.

e) Tunnels R_i : pairs points on the grid that you can use on the grid to travel while avoiding traffic.

Actions:

- **up/down/left/right**: each truck can move in either of the 4 directions through one of the edges of the grid (only one edge at a time). This action has a cost equal to the traffic level of this segment.
- **tunnel**: a truck can go through a tunnel only if it is at one of its two entrances. Each tunnel connects between two points only and goes both ways. The traffic of a tunnel is always equal to the Manhattan distance between its two entrances.

Implementation: In this project, you will implement a search agent that tries to find a path that achieves the goal of delivering all packages. Additionally, for optimal strategies, the solution has to find a plan that delivers in the shortest total time possible. Several search strategies will be implemented and each will be used to plan the agent's solution to this problem. The search is to be implemented as described in the lectures (Lecture 2):

a) Breadth-first search.

b) Depth-first search.

c) Iterative deepening search.

d) Uniform cost search.

e) Greedy search with at least two heuristics.

f) A* search with at least two *admissible* heuristics. A trivial heuristic (e.g., $h(n) = 1$) is not acceptable.

Different solutions should be compared in terms of run-time, number of expanded nodes, memory (RAM) utilisation and CPU utilisation. **You are required to implement this agent using Java.**

Your implementation should have the following classes:

- `GenericSearch`, which has the generic implementation of a search problem (as defined in Lecture 2).
- `DeliverySearch`, which extends `GenericSearch`, implementing the Package Delivery search problem for each truck-product pair separately.
- `DeliveryPlanner`, which implements the planning of which trucks to send for each product.

You can implement any additional classes you want.

Inside `DeliverySearch` you will implement `solve` as the key function which will be the basis for testing :

- `GenGrid()` randomly generates a grid. T `GenGrid()` should return a string representation of a grid; the format of the string is indicated below.
- `path(...)` uses search to find the path for a truck by which it will reach a certain destination. The parameters can be whatever your implementation requires. It uses the strategy that is given to `solve`. The function returns a `String` of 3 elements, in the following format:
  `plan;cost;nodesExpanded`
  where:
  - `plan`: the sequence of actions that lead to the goal (if such a sequence exists) where the is a sequence of movements separated by commas.
    For example: `up,down,left,up,tunnel,right,down,up,tunnel,left,down`.
  - `cost`:the total time spent in traffic.
  - `nodesExpanded`: is the number of nodes chosen for expansion during the search.
- `plan(`*initalState*`,`*strategy*`, `*visualize*`)` finds the final plan including which truck will deliver which products. It calls `path` to get the paths of each truck. The parameters are as follows:
  - `initalState` is a string providing the parameters of the initial state as follows:

    ```
    m;n;P;S;
    CustomerX_1,CustomerY_1,
    CustomerX_2,CustomerY_2,..
    CustomerX_i,CustomerY_i;
    TunnelX_1,TunnelY_1,TunnelX_1,TunnelY_1,
    TunnelX_2,TunnelY_2,TunnelX_2,TunnelY_2,...
    TunnelX_i,TunnelY_i,TunnelX_i,TunnelY_i,
    ```

    * `m, n` grid dimensions, width and height.
    * `P` number of packages to be delivered and also the number of customers; one package for each customer, $1, \leq P \leq 10$.

    * S is the number of stores and also the number of available trucks. $1 \leq S \leq 3$
    * Customer Locations
    * pairs of locations of tunnel entrances
- `traffic` a string showing the traffic of each segment in the city. A traffic level of 0 indicates a blocked road. The input is formatted as follows:

```
SrcX_1,SrcY_1,DstX_1,DstY_1,Traffic_1;
SrcX_2,SrcY_2,DstX_2,DstY_2,Traffic_2;....
SrcX_i,SrcY_i,DstX_i,DstY_i,Traffic_i;
```

  initialState and traffic do not have any spaces or new lines but are formatted this way her for readability
- `strategy` is a string indicating the search strategy to be applied:
    * `BF` for breadth-first search,
    * `DF` for depth-first search,
    * `ID` for iterative deepening search,
    * `UC` for uniform cost search,
    * `GR`i for greedy search, with $i \in \{1, 2\}$ distinguishing the two heuristics.
    * `AS`i for A* search with $i \in \{1, 2\}$ distinguishing the two heuristics.
- `visualize` is a Boolean parameter which, when set to `true`, results in your program's side-effecting displaying the location of the trucks on the city grid at every step of the plan. *A GUI is not required, printing to the console would suffice.* The main value of this part is to help you debug and understand.

The function's output is a string showing pairs of (store, destination) and the output of path for this trip.

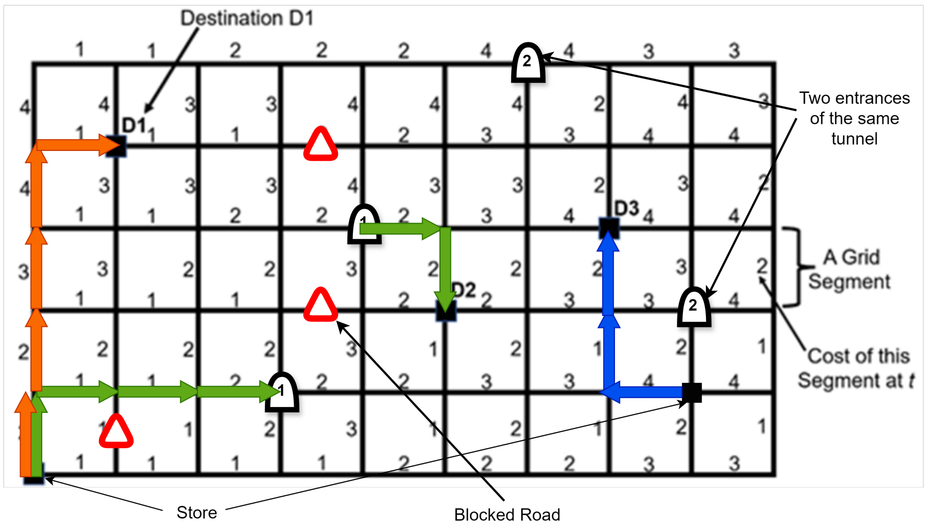


Figure 1: **Example:** In this figure, the bold arrows present examples of paths to be followed by the delivery truck starting at (0,0) and (8, 1) to distribute the packages to the destinations D1, D2, D3.

Groups: You may work in groups of at most four.

Deliverables

a) **Source Code submitted through the form.**

- You should implement an abstract data type for a search-tree node as presented in class. (*This does not necessarily mean that your implementation should include Java abstract classes.*)
- You should implement an abstract data type for a generic search problem, as presented in class.
- You should implement the generic search procedure presented in class(Lecture 2 slides), taking a problem and a search strategy as inputs.You should make sure that your implementation of search minimises or eliminates redundant states and that all your search strategies terminate within the time limits of the automated public test cases that will be posted.
- You should implement a DeliverySearch subclass of the generic search problem. This class must contain `solve` as a static method.
- You should implement all of the search strategies indicated above (together with the required heuristics).
- Your program will be subject to both public and private test cases.
- Your source code should have two packages. A package called `tests` and a package called `code`. All your code should be located in the `code` package and the test cases (when posted) should be imported in the `tests` package.

b) **Project Report, including the following.**

- A class diagram showing the main classes and functions in your implementation.
- A discussion of how you implemented the various search algorithms.
- A discussion of the heuristic functions you employed and, in the case of greedy or A*, an argument for their admissibility.
- A comparison of the performance of the different algorithms implemented in terms of **completeness, optimality, RAM usage, CPU utilization, and the number of expanded nodes**. You should comment on the differences in the RAM usage, CPU utilization, and the number of expanded nodes between the implemented search strategies.
- Proper citation of any sources you might have consulted in the course of completing the project.
- If you use code available in library or internet references, make sure you *fully* explain how the code works and be ready for an oral discussion of your work.
- If your program does not run, your report should include a discussion of what you think the problem is and any suggestions you might have for solving it.

c) **A PPT presentation should be prepared for a fixed defense with the TA: Ms. Salma El-Sayed.**

Grading Criteria

The following are *some* of the main points regarding how the submission is evaluated:

- The report is a major component of the grade around 20%

- There are 10% progress grades given during the first 2 Checking Meetings
- Some JUnit tests will be posted and a part of the grade will be based on passing them.
- The Evaluation will cover the remaining weight and will be graded based on the code and presentation.
- The incorrect implementation of search and each strategy (not using search or incorrectly implementing the strategies will result in a grade deduction)
- Optimal strategies should result in optimal solutions with respect to path cost.
- Some test scenario parameters will be provided for which the project should return a valid output.

Important Dates

a) **First Checking Meeting with TA:** Thursday, 24th of October

b) **Second Checking Meeting with TA:** Thursday, 14th of November

c) **Submission of Report + Codes:** Saturday, 8th of December
Submission to: `https://forms.gle/6meXMeehn4e9gsXS71`

d) **Project Defense:** To be determined for every team between 9th and 13th of December. No evaluations will be held after the 13th of December.