

**AI Project 1 Report**  
**Team: Hazam**

**ESLAM ELSHARKAWY 201800473**  
**MOHAMED ISMAIL 201800595**  
**MOHAMED HAZEM 201802190**

## **Workload Distribution**

### **Mohamed Ismail:**

1. Problem Formulation and Modeling

### **Eslam El Sharkawy:**

1. Informed Search Algorithms
  - a. A\*
  - b. GFS
  - c. Hill Climbing
  - d. Local Beam Search
  - e. Simulated Annealing

### **Mohamed Hazem:**

1. Uninformed Search Algorithms
  - a. BFS
  - b. DFS
  - c. IDS
  - d. UCS

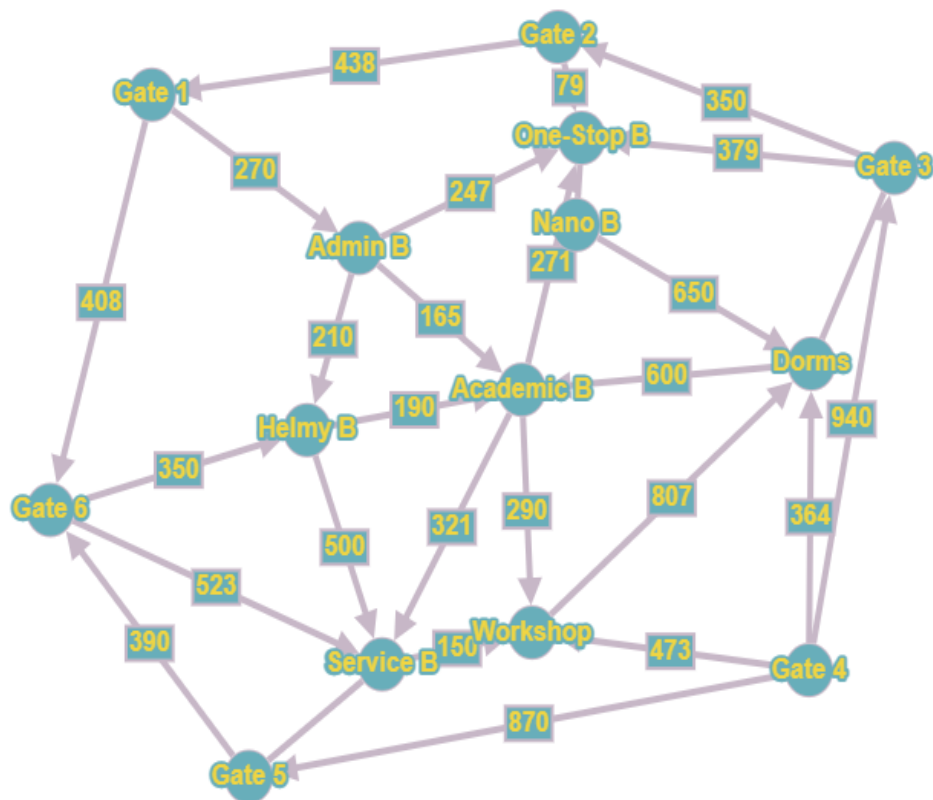
## Problem Formulation

We solved the problem using a maze approach.

First we started by extracting multiple images for zewail city from google maps



Then we marked the landmarks and extracted x, y location relative to the center of the city. These x, y values are used for the euclidean heuristics.



After that we divided the city map into a grid. Each grid cell is 20m squared. Then we converted this grid to ascii where symbol # is a wall or unreachable area, the symbol = is a car road, the symbol - is passenger road, x is a car crossing and \$ is a car slow down. And this is the final result

```
_ZC_ROAD_MAP = '''
#####
#####A#####
###===$$======$$=====###
###-#####=-#####=-#####
###=#####=-#####=-#####
###=#####$#####=-#####
#F=-#####=-#####-----H---B##
##=-#####=-#####I-----##
##=-#####=-#####-----##
##=-###-====xx=====--#####=-##
##$-$-====G=====--#####$##
##=-#-----#####-----#####=-##
##=-#-L--#####--K--#####=-##
#Exx-##=-#####-##=-#####xxC#
#=----==#-#####-##=-#####=-#
#$-$-#=====J=====#####$-$-#
#=-#-====#####-----#####=-#
#=-M-#-====#####-----#=-#
#=-=====#####P=-#
#=-N-#-=====-----#####-=-#
#=------=====-----#####-=-#
#=----###---xx---#####-=-#
#=-#-#####=-#####=-#
#####D#####
#####
'''
```

And these are the step costs for all reachable areas

```
step_cost_map = {
    '=': 1, # cars road
    '-': 2, # passengers road
    'x': 3, # crossing
    '$': 4, # car slow down
}
```

## Project Structure

```
├── dump
│   └── zc_map.py
├── lib
│   ├── __pycache__
│   │   ├── settings.cpython-310.pyc
│   │   └── utils.cpython-310.pyc
│   ├── settings.py
│   └── utils.py
├── main.py
├── models
│   ├── problem.py
│   ├── __pycache__
│   │   ├── problem.cpython-310.pyc
│   │   ├── tree.cpython-310.pyc
│   │   └── zc_map.cpython-310.pyc
│   ├── tree.py
│   └── zc_map.py
├── settings.json
├── solvers
│   ├── a_star.py
│   ├── bfs.py
│   ├── dfs.py
│   ├── greedy_best_first.py
│   ├── hill_climbing.py
│   ├── ids.py
│   ├── local_beam_search.py
│   ├── __pycache__
│   │   ├── a_star.cpython-310.pyc
│   │   ├── bfs.cpython-310.pyc
│   │   ├── dfs.cpython-310.pyc
│   │   ├── greedy_best_first.cpython-310.pyc
│   │   ├── hill_climbing.cpython-310.pyc
│   │   ├── ids.cpython-310.pyc
│   │   ├── local_beam_search.cpython-310.pyc
│   │   ├── simulated_annealing.cpython-310.pyc
│   │   └── ucs.cpython-310.pyc
│   ├── simulated_annealing.py
│   └── ucs.py
```

7 directories, 31 files

## Showcasing Results

You start the program by running the `main.py` file.

```
$ python ./main.py
0 # # # # # # # # # # # # # # # # # # # # # # # # 
1 # # # # # # # # # # A # # # # # # # # # # # # # 
2 # # #                                     # # # 
3 # # #   # # # # # # #                 # # # # # # # # 
4 # # #       # # # # # #             # # # # # # # # 
5 # # #           # # # # # #         # # # # # # # # 
6 # F          # # # # # # #         # # # # #        H      B # # 
7 # #            # # # # # # #     # # # # I              # # 
8 # #            # # # # # # #     # # # # #          # # 
9 # #            # # # # # # #                # # # # # # # # 
10 # #               #                  G                   # # # # # # # # 
11 # #               #                    # # # # #         # # # # # # # # 
12 # #               #                     L    # # # # # K    # # # # # # # C # 
13 # E               #                      # # # # # # # # # # # # # # 
14 #                 # # # # # # # # # # # # # # # # # # # # # # # 
15 #                   #                       J                   # # # # # # # # 
16 #                   #                      # # # # # # # # # # # # # 
17 #                   M #                     # # # # # # # # # # # 
18 #                   # # # # # # # # # # # P                        # 
19 #                   N #                               # # # # # # # # 
20 #                                           # # # # # # # # # # # 
21 #                                   # # # # # # # # # # # # # # # # 
22 #                                       # # # # # # # # # # # # # # # 
23 # # # # # # # # # # # D # # # # # # # # # # # # # # # # # # # # # # 
24 # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # 

{
  "gate_1": "A",
  "gate_2": "B",
  "gate_3": "C",
  "gate_4": "D",
  "gate_5": "E",
  "gate_6": "F",
  "admin_building": "G",
  "one_stop_shop": "H",
  "science_vally": "I",
  "academic_building": "J",
  "nano_building": "K",
  "helmy_building": "L",
  "service_building": "M",
  "work_shops": "N",
  "dorms": "P"
}

enter your current location:
```

This is the prompt that appears when you start the program.

First it asks for your current location. You can enter a specific zewail city landmark or you can enter general x, y in this form y,x.



After that we have the algorithms prompt.

```
}
enter your current location: gate_1
enter your distantiation: service_building
['gfs', 'a_star', 'bfs', 'dfs', 'ids', 'ucs', 'sim', 'hill', 'local']
enter solving algorithm: 
```

After we choose a certain algorithm for example a\_star. The solution appears if it exists and the program draws a map to follow to reach your destination.

```
enter your distantiation: service_building
['gfs', 'a_star', 'bfs', 'dfs', 'ids', 'ucs', 'sim', 'hill', 'local']
enter solving algorithm: a_star
#####
###          |          ###
### #####| #####
### #####| #####
### #####| #####
#F #####| ##### H B##
## #####| #####I  ##
## #####| #####
## #####| #####
## #      |-----| #####
## #      |#####
## # L #####| K #####
#E # ##|###|### C#
#   ##|###|### #
#   # J |#####
#   #   |#####
# +| #   |### #
# ----|###| P #
# N # ----|### #
#           #####
#           #####
#####D#####
#####
(['down', 'down', 'down', 'down', 'down', 'down', 'down', 'right', 'right', 'right', 'down', 'down', 'down', 'down', 'down', 'down', 'down', 'down', 'down', 'down', 'left', 'left', 'left', 'left', 'left', 'up', 'left', 'left', 'left', 'left', 'left', 'left', 'up', 'left'], 46)
```

```
enter your distantiation: service
['gfs', 'a_star', 'bfs', 'dfs', 'ids', 'ucs', 'sim', 'hill', 'local']
enter solving algorithm: a_star
#####
#####*#####
###          |          ###
### #####| #####
### #####| #####
### #####| #####
#F #####| ##### H B##
## #####| #####I  ##
## #####| #####
## #####| #####
## #      |-----| #####
## #      |#####
## # L #####| K #####
#E # ##|###|### C#
#   ##|###|### #
#   # J |#####
#   #   |#####
# +| #   |### #
# ----|###| P #
# N # ----|### #
#           #####
#           #####
#####D#####
#####
```



## BFS Test

```
enter your distantiation: service
['gfs', 'a_star', 'bfs', 'dfs', '']
enter solving algorithm: bfs
#####
#####*#####
###          |          ###
### #####| #####
### #####| #####
### #####| #####
#F #####| ##### H B##
## #####| #####I ##
## #####| #####
## #####- -| #####
## # -| G #####
## # - -| #####
## #| L ##### K #####
#E #|### ##### C#
# - -|### #####
# | # J #####
# | # #####
# +| # #####
# ##### P #
# N # #####
# #####
# #####
# #####
#####D#####
#####
```

Path cost is 40 (Not Optimal)

## IDS

```
enter your distantiation: service
['gfs', 'a_star', 'bfs', 'dfs', '']
enter solving algorithm: ids
#####
#####*#####
##-----|#####
##|#####
##|#####
##|#####
##-|#####H B##
#|-#####I##
#-|#####
#|-#####
##|-#G#####
#--|######
#|--#L#####K#####
#--|#########C#
#|--########
---|#J#####
|--######
#+M######P#
#N######
######
#########
######
#####D#####
#####
```

Path Cost is 68 (Not Optimal)

## UCS

```
enter your distantiatiion: service
['gfs', 'a_star', 'bfs', 'dfs', '']
enter solving algorithm: ucs
#####
#####*#####
###          |          ###
### #####| #####
### #####| #####
### #####| #####
#F #####-| ##### H B##
## #####| #####I ##
## #####| #####
## #####---| #####
## # -| G #####
## #-| #####
## #| L ##### K #####
#E #|### ##### C#
# ---|### #####
# | # J #####
# | # #####
# +M # #####
# ##### P #
# N # #####
# #####
# ### #####
# #####
#####D#####
#####
```

Path Cost is 38 (Optimal)

## DFS

Always Stuck. We think that is because the complex nature of the map

```

enter your distantiation: service
['gfs', 'a_star', 'bfs', 'dfs', '']
enter solving algorithm: gfs
#####
#####*#####
###          |          ###
### ##### | #####
### ##### | #####
### ##### | #####
#F ##### | ##### H B##
## ##### | #####I  ##
## ##### | #####  ##
## ##### |          ##### #
## #      | ---      ##### #
## #      | ##### | ##### #
## # L    | ##### | K    ##### #
#E #      | ##### | ##### C#
#      | ##### | ##### #
# #      | J      | ##### #
# #      | ##### | ##### #
# + | --- | ##### | ##### #
# -- |   | ##### | P    #
# N #   | ----- | ##### #
#          ##### #
#      ###      ##### #
#      ##### #
#####D#####
#####

```

**Path Cost 50 (Not Optimal)**

A\*

```
enter your distantiatiion: service
['gfs', 'a_star', 'bfs', 'dfs', '
enter solving algorithm: a_star
#####
#####*#####
###          |          ###
### #####| #####
### #####| #####
### #####| #####
#F #####| ##### H B##
## #####| #####I ##
## #####| #####
## #####| #####
## # |--- #####
## # #####| #####
## # L #####| K #####
#E # ### #####| ### ##### C#
# # # #####| ### #####
# # J | #####
# # #####| #####
# + | # #####| ###
# - - - - - | ##### P
# N # - - - - - | #####
# #####
# #####
# #####
#####D#####
#####
```

Path Cost 46 (Optimal)

## Hill Climbing

```
['gfs', 'a_star', 'bfs', 'dfs', '
enter solving algorithm: hill
#####
#####*#####
###      +---      ###
### #####      #####
### #####      #####
### #####      #####
#F #####      H   B##
## #####      I     ##
## #####      ##
## #####      #####
## #          G      #####
## #          #####
## # L ##### K      #####
#E # ### ##### ### ##### C#
#   ### ##### ### #####
#   #          J      #####
#   #          #####
# M #          #####
#   #####          P   #
# N #          #####
#           #####
#   ###          #####
#   #####
#####D#####
#####
(['down', 'right', 'left', 'left'
```

Always stuck in a local minima due to car slow downs

## Simulated Annealing

```
['gfs', 'a_star', 'bfs', 'dfs', '
enter solving algorithm: sim
#####
#####A#####
###                                     ###
### #####      #####      #####
### #####      #####      #####
### #####      #####      #####
#F #####      #####      H      *##
## #####      #####I      | ##
## #####      #####      | ##
## #####      #####      | ##
## #          G          ##### | ##
## #          #####      ##### | ##
## # L ##### K          ##### | ##
#E # ### ##### ###      ##### | C#
#      ### ##### ###      ##### | - #
# #          J          ##### | #
# #          #####      ##### | #
# M #          #####      ### | #
#          #####      + - - | #
# N #          #####      #
#          #####      #
#      ###          #####      #
#      #####          #
#####D#####
#####
(['down', 'down', 'down', 'down',
```

Path Cost 27 (Not Optimal)



## Local Beam search

```
['gfs', 'a_star', 'bfs', 'dfs', '
enter solving algorithm: local
#####
#####A#####
###                                     ###
### #####      #####      #####
### #####      #####      #####
### #####      #####      #####
#F #####      #####      H      *##
## #####      #####I      | ##
## #####      #####      | ##
## #####      #####      | ##
## #####      #####      | ##
## #          G          ##### | ##
## #          #####      ##### | ##
## # L ##### K          ##### | ##
#E # ### ##### ###      ##### | C#
#   ### ##### ###      ##### | #
#   #          J          ##### | #
#   #          #####      ##### | #
# M #          #####      ## | #
#   #####          +- | #
# N #          #####      #
#           #####      #
#   ###          #####      #
#   #####          #
#####D#####
#####
(['down', 'down', 'down', 'down',
```

Path Cost 23 (Not Optimal)