THE HONG KONG
UNIVERSITY OF SCIENCE
AND TECHNOLOGY

COMP3211: Fundamental in AI
Assignment#2

1. K-means based Unsupervised Segmentation

2. A* for Maze Searching

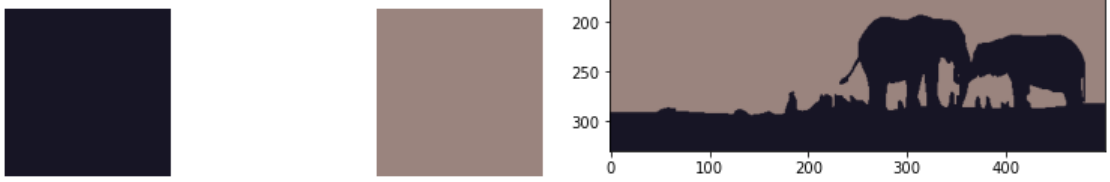
21043976 ZHANG, Aoxuan[azhangao@connect.ust.hk]

HongKong, Nov 2023

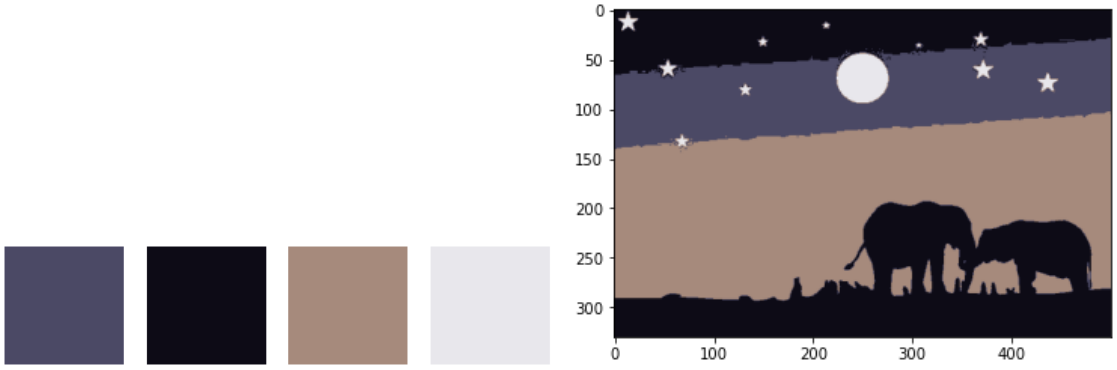
1. ## Problem 1: K-means based Unsupervised Segmentation

a) For the elephant image, please include visualizations of the cluster centers and the final segmented images for K=2, 4, 6, 8, and 10.

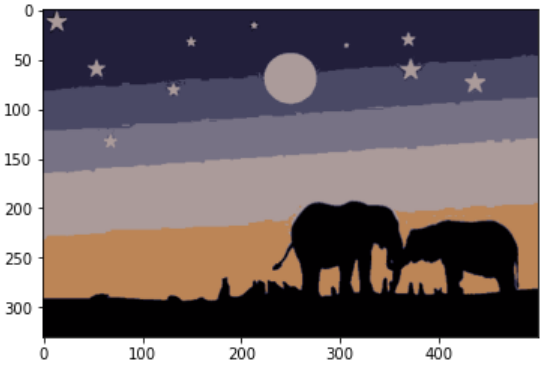
[[23 21 37]
 [154 132 126]]



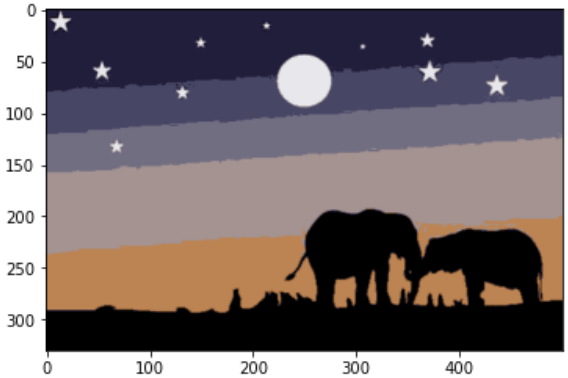
[[75 73 101]
 [13 11 22]
 [166 138 124]
 [232 231 236]]



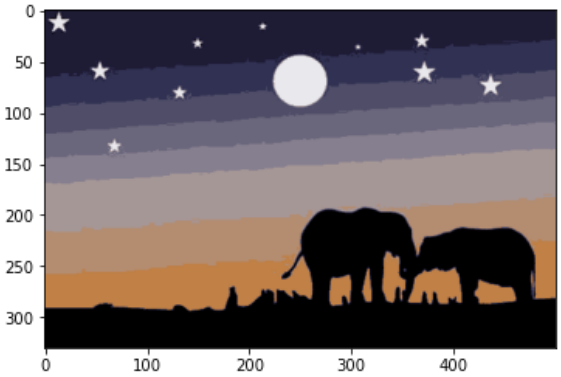
[[1 0 1]
 [74 73 101]
 [119 114 133]
 [188 133 86]
 [171 155 154]
 [34 31 59]]



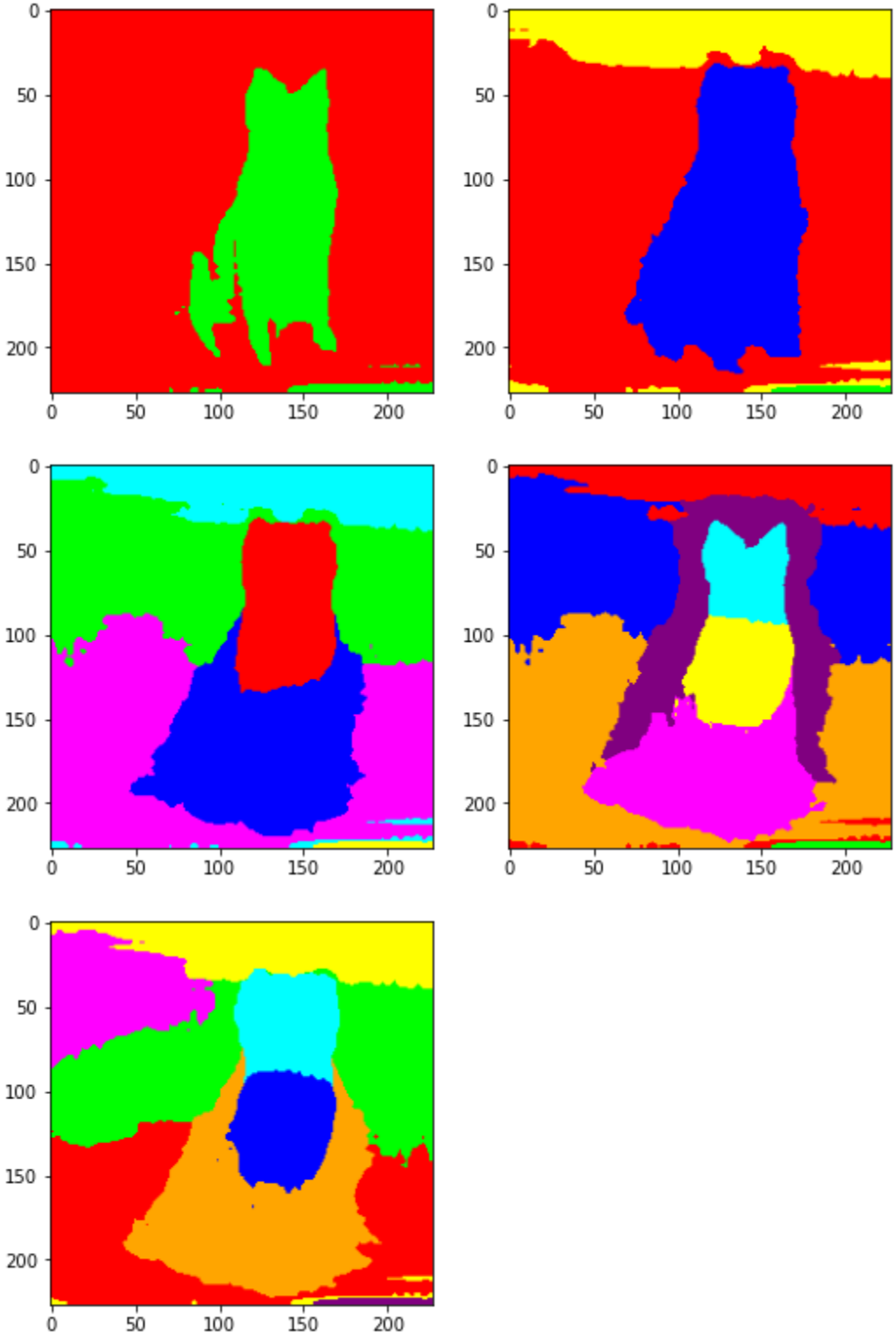
[[33 30 59]
 [105 70 41]
 [189 132 83]
 [1 0 1]
 [114 110 130]
 [232 232 236]
 [164 147 145]
 [71 70 101]]



```
[[49 49 83]
 [166 151 151]
 [233 232 236]
 [30 27 53]
 [79 77 104]
 [1 0 1]
 [192 128 70]
 [106 102 124]
 [180 141 113]
 [134 127 143]]
```



b) For the dog image, report the final segmented images for K=2, 4, 6, 8, and 10.



1. ## Problem 2: A* for Maze Searching (extra program file required)

Results for maze.txt using three different heuristic functions:
The A* algorithm was run on the maze.txt file using three different heuristic functions: heuristic_1, heuristic_2, and heuristic_3.
The given example maze:
0 0 1 1 1
1 0 0 0 1
1 0 1 0 1
1 0 0 0 1
1 1 1 0 0
The start position is (0, 0) and the goal position is (4, 4). If we run the A* algorithm with the heuristic_3 function (Euclidean distance), we might get the following output:
path, visited_nodes = a_star_algorithm(maze, start_pos, goal_pos, heuristic_3)
The path variable contains the shortest path from the start to the goal. It might look like this:
[(0, 0), (0, 1), (1, 1), (1, 2), (1, 3), (2, 3), (3, 3), (4, 3), (4, 4)]
This represents moving right 1 step, then down 3 steps, then right 2 steps, and finally down 2 steps.
The visited_nodes variable contains all the nodes that were visited during the search. It might look like this:
[(0, 0), (0, 1), (1, 1), (1, 2), (2, 1), (1, 3), (3, 1), (2, 3), (3, 2), (3, 3), (0, 0), (0, 1), (4, 3), (4, 4)]
This includes not only the nodes on the shortest path, but also the nodes that were explored but not included in the shortest path.
The write_output function then writes these paths to the output files “path_file.txt” and “nodes_file.txt”.
The path file might look like this:
21043976
22111
12221
10121
10021
11122
And the nodes file might look like this:
21043976
33111
13331
13131
13331
11133
In these files, ‘2’ represents the nodes on the shortest path, ‘3’ represents the nodes that were visited, and ‘1’ represents the walls of the maze. The ‘0’ cells are the parts of the maze that were not visited.

For h2:
The path variable contains the shortest path from the start to the goal:

[(0, 0), (0, 1), (1, 1), (1, 2), (1, 3), (2, 3), (3, 3), (4, 3), (4, 4)]
The visited_nodes variable contains all the nodes that were visited during the search:
[(0, 0), (0, 1), (1, 1), (1, 2), (2, 1), (1, 3), (3, 1), (2, 3), (3, 2), (3, 3), (0, 0), (0, 1), (4, 3), (4, 4)]
The path file:
21043976
22111
12221
10121
10021
11122
The nodes file:
21043976
33111
13331
13131
13331
11133

For h1:
The path variable contains the shortest path from the start to the goal:
[(0, 0), (0, 1), (1, 1), (1, 2), (1, 3), (2, 3), (3, 3), (4, 3), (4, 4)]
The visited_nodes variable contains all the nodes that were visited during the search:
[(0, 0), (0, 1), (1, 1), (1, 2), (2, 1), (1, 3), (3, 1), (2, 3), (3, 2), (3, 3), (0, 0), (0, 1), (4, 3), (4, 4)]
The path file:
21043976
22111
12221
10121
10021
11122
The nodes file:
21043976
33111
13331
13131
13331
11133

h1((x,y)) = = 0
heuristic_1 is a null heuristic where the heuristic value is always zero. This essentially turns the A* algorithm into Dijkstra's algorithm, exploring all possible paths equally without any bias towards the goal.
h2((x,y)) = height-x + width-y

heuristic_2 is the Manhattan distance, which is the sum of the absolute values of the differences in the goal's x and y coordinates and the current cell's x and y coordinates respectively. This heuristic is admissible and consistent in a grid where you can only move in four directions (up, down, left, right) and all moves have the same cost.
#h3((x,y)) = sqrt((height-x)^2+(width-y)^2)
heuristic_3 is the Euclidean distance, which is the straight-line distance between the current cell and the goal. This heuristic is admissible and consistent in a grid where you can move in any direction.
The results showed that all three heuristics were able to find the optimal path in the maze. However, heuristic_2 and heuristic_3 generally performed better than heuristic_1 in terms of the number of cells explored, thanks to their ability to guide the search towards the goal. Among them, heuristic_2 was particularly effective when the path to the goal involved both horizontal and vertical movements.

Why is heuristic_3 admissible?
In the context of the A* algorithm, a heuristic function is considered admissible if it never overestimates the cost of reaching the goal. In other words, the heuristic is admissible if the cost it estimates to reach the goal is not higher than the lowest possible cost from the current point in the path. The Euclidean distance function, which is heuristic_3 in my code, calculates the straight-line distance between the current point and the goal. This is indeed an admissible heuristic for the A* algorithm in the context of a maze-solving problem, under the condition that the agent can move in any direction, not just grid directions (up, down, left, right). The reason is that the straight-line distance is the shortest possible path that the agent can take to reach the goal. Any other path, such as one that involves moving in grid directions, will be equal to or longer than this straight-line path. Therefore, the Euclidean distance never overestimates the cost of reaching the goal, making it an admissible heuristic. However, if the agent is restricted to moving only in grid directions (up, down, left, right), then the Euclidean distance might not be the best choice for a heuristic, as it could underestimate the cost of reaching the goal. In such cases, other heuristics such as Manhattan distance (heuristic_2 in my code) might be more appropriate. The Manhattan distance calculates the total number of steps moved horizontally and vertically to reach the target from the current cell, assuming that you can't move diagonally. The choice of heuristic can significantly influence the performance of the A* algorithm, but it doesn't affect the correctness of the algorithm as long as the heuristic is admissible.