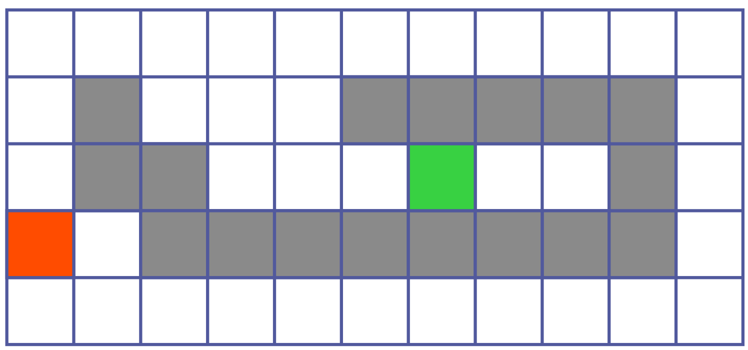
16.410-13 Recitation 9 Problems

Problem 1: A* Search

Robot Navigation Consider the maze given in the following figure. Name 3 admissible heuristics for this problem. What would be a good admissible heuristics?



Write down the steps that an A* search would go through using the best heuristic you can think of. You may find it convenient to write the cost-to-go values for each tile on the figure.

Traveling in Romania You are taking your vacation in East Europe. You have just crossed the border from Hungary to Romania. It turned out Romania is a beautiful country.

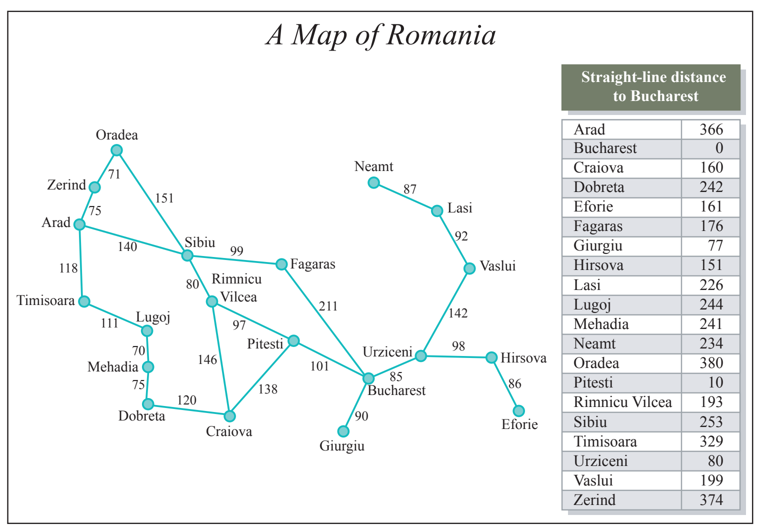
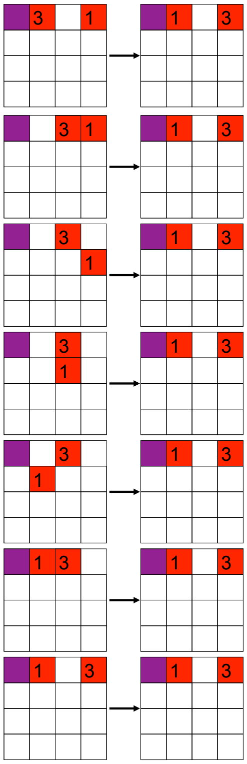


Image by MIT OpenCourseWare.

You have started in Arad on the Hungary border. You would like to see Bucharest. Use the Euclidean distance cost-to-go heuristic in answering the following questions (heuristic values are given above).

- Find a path that gets you to Bucharest using the Greedy search. Draw the search tree.
- Use A* to find an optimal that takes you to Bucharest. Draw the search tree.
- Describe how you would construct a "controller" that takes you to Bucharest from anywhere in Romania in case you get lost somewhere. Show the execution of the Dijkstra algorithm.

Problem 2: Admissible Heuristics



Puzzle Recall the 4-puzzle. Remember the heuristics that you have learned in the class. Describe the Manhattan distance heuristic. Prove that the Manhattan distance heuristic is an admissible heuristic.

Now, consider the following heuristic. Two tiles t_j and t_k are in a *linear conflict* if t_j and t_k are the same line, the goal positions of t_j and t_k are both in that line, t_j is to the right of t_k, and goal position of t_j is to the left of the goal position of t_k. The *linear conflict heuristic* moves any two tiles that are in linear conflict without colliding them. See the figure in left.

Is this heuristic admissible? Either prove that it is admissible, or disprove by a counter-example. Is this heuristic better than the Manhattan distance heuristic.

Rubik puzzle Rubik puzzle is one of the hardest combinatorial problems to date (see the figure below). We would like to solve it using the A* algorithm. What would be a good admissible heuristic for this problem?

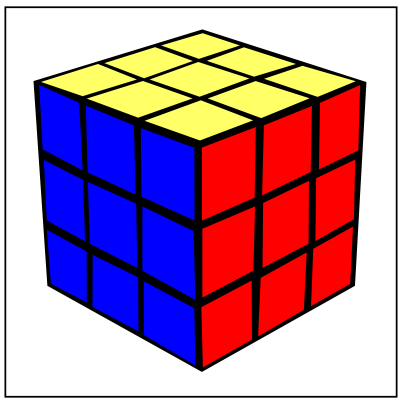
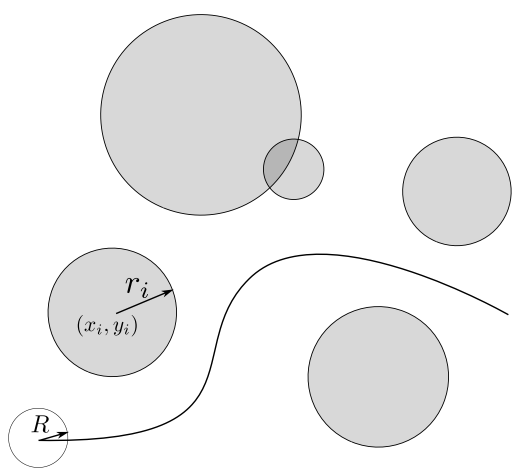


Image by MIT OpenCourseWare.

Problem 3: Collision Checking

Recall that sampling-based motion planning algorithms require an "oracle" that checks whether or not a path collides with an obstacle or not. Assume that you have circle-shaped rigid body robot that moves on a plane (2 dimensions). The radius of the robot is R. Each obstacle in the environment is also shaped as a circle. The obstacles are given in the form of a list such that each obstacle is described by the triple (x_i, y_i, r_i), where x_i the x-axis coordinate, y_i is the y-axis coordinate, and r_i is the radius of the obstacle.



Checking collision with a single obstacle Devise a method to quickly check whether a given straight path collides with a given obstacle (there is only one obstacle). You can use vector operations (e.g., the dot product or vector multiplications and simple addition and multiplication).

HINT: Remember the configuration space idea.

Efficiently checking collision with multiple obstacles Devise a method to store the obstacles in a data structure so that checking collision with obstacles is more efficient than checking collision each obstacle one by one. Analyze the complexity of your collision checking algorithm. How does it scale with the number of obstacles? Compare this to the complexity of checking collision each obstacle one by one.

16.410 / 16.413 Principles of Autonomy and Decision Making
Fall 2010