The Pacman Domain: Abstraction is used to identify only relevant information for a domain, so as to simplify the search or planning process. The appropriateness of abstraction often depends on PEAS. For example, in the Pacman Domain considered, when the ghosts are separated from the Pacman, they do not need to be considered. Abstraction can significantly reduce the state space to consider, which is important (as we will soon see) for search.

Abstraction can sometimes be used to simplify a domain. For example, the reason for keeping the ghosts permanently scared is so that we can ignore the ghosts. Without this abstraction, the domain becomes more complicated since we then must take the ghosts into account (consider instead the problem of eating all dots).

State space graph: It is difficult to keep the state space graph in memory other than the smallest problems given its exponential characterization.

Search tree: It is conceptually very useful but even more expensive to create than the state space graph.

State space graph vs search tree: They correspond to two important search methods: graph search and tree search. We will explain search strategies using search tree and let you explore graph search in your project.

Search with search tree: Imagine how you would search for a specific state first. A central concept in search is fringe, which is maintained as a set of candidate nodes to be explored next. All search strategies are essentially about how to select which one to explore first. A similar notion is used in graph search where care must be taken when visiting the same states.

Depth-First Search: DFS is often implemented as a stack. LIFO. The behavior of DFS resembles a sweep from the left to right part of the tree.

Breadth-First Search: BFS is often implemented as a queue. FIFO. The behavior of BFS resembles a sweep from top to bottom part of the tree.

Iterative deepening: If the branching factor is b, how much more computation is done with iterative deepening than with BFS? It is only a constant factor (b) more computation.

Cost sensitive search: Cost sensitive search is used when the costs of actions are not always the same.

Properties of UCS: Why do we need the costs to be positive? Since otherwise we cannot be sure the path that first reaches the goal is the optimal cost path anymore! If we have negative
cost edges, can we just add a constant to all edge costs to make them positive? No. This could have changed the optimal path by biasing towards the shorter path (consider adding 2 to each edge in the scenario below). Can the costs be zeros? Well, if no zero-cost loops are present.