06-0: **Overview**

- Heuristic Search - exploiting knowledge about the problem
- Heuristic Search Algorithms
  - "Best-first" search
  - Greedy Search
  - A* Search
  - Extensions to A*
- Constructing Heuristics

06-1: **Informing Search**

- Uninformed search was able to find solutions, but were very inefficient.
  - Exponential number of nodes expanded.
- By taking advantage of knowledge about the problem structure, we can improve performance.
- Two caveats:
  - We have to get knowledge about the problem from somewhere.
  - This knowledge has to be correct.

06-2: **Best-first Search**

- Uniform-cost search
  - Nodes were expanded based on their total path cost
  - Implemented using a priority queue
- Path cost is an example of an *evaluation function*.
  - We'll use the notation $f(n)$ to refer to an evaluation function.
- An evaluation function tells us how promising a node is.
- Indicates the quality of the solution that node leads to.

06-3: **Best-first Search**

- Best-first Pseudocode

```
enqueue(initialState)
do
  node = prioroty-dequeue()
  if goalTest(node)
     return node
  else
    children = successors(node)
    for child in children
        prioroty-enqueue(child, f(child))
```

- where insert-with orders our priority queue accordingly.