Lecture 4:

A* Search, Minimax search and Decision Trees

30 January 2020

Gradient Descent

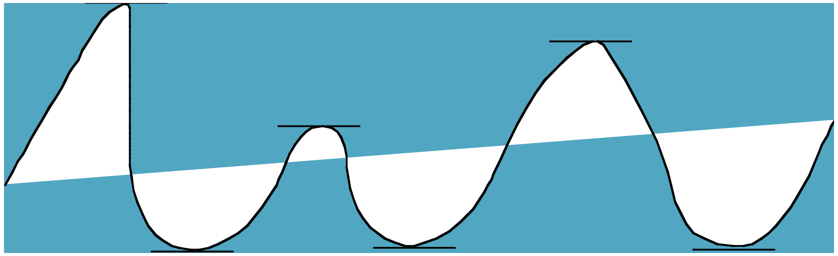
How does GD algorithm know it has reached min/max?

- $\nabla h = 0$
  - Could be a min, max or saddle point

How to determine if min/max

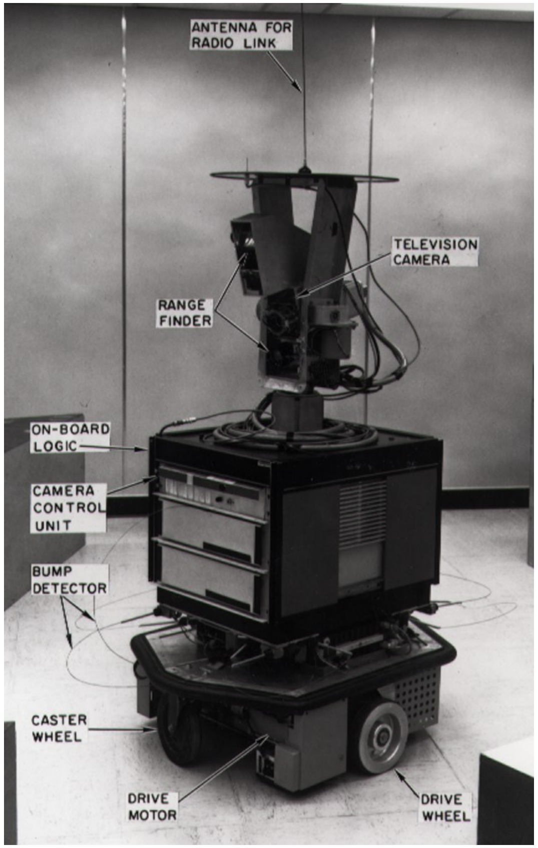
- 1d - second derivative test
  - $\nabla h > 0 \Rightarrow$ local min
  - $\nabla h < 0 \Rightarrow$ local max
  - $\nabla h = 0 \Rightarrow$ undetermined, may be saddle point
- Higher dimensions, use Hessian to perform second derivative test
  - $$\mathbf{H} = \begin{bmatrix} \dfrac{\partial^2 f}{\partial x_1^2} & \dfrac{\partial^2 f}{\partial x_1\,\partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_1\,\partial x_n} \\ \dfrac{\partial^2 f}{\partial x_2\,\partial x_1} & \dfrac{\partial^2 f}{\partial x_2^2} & \cdots & \dfrac{\partial^2 f}{\partial x_2\,\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \dfrac{\partial^2 f}{\partial x_n\,\partial x_1} & \dfrac{\partial^2 f}{\partial x_n\,\partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_n^2} \end{bmatrix}.$$

- **No, you don't need to know this!!!**



A* Search

- Order nodes by sum of path length and heuristic
- Priority queue order by path(s) + h(s)
- Will find optimal path if using admissible heuristic

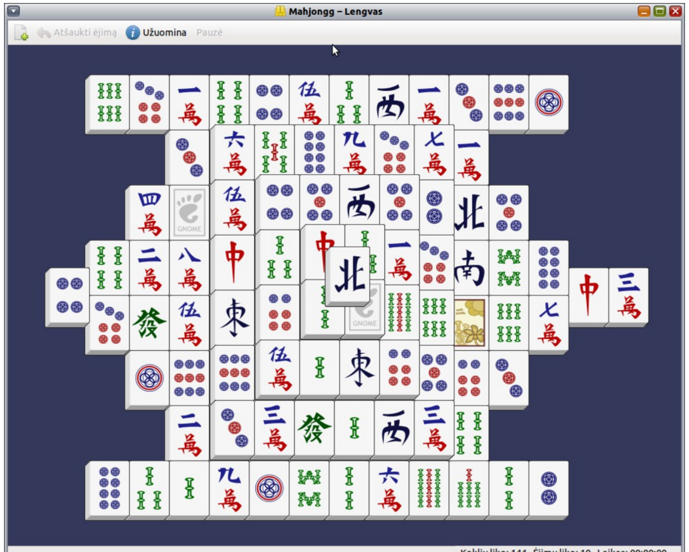



```
priority_queue.push(initial state, 0)
While(!priority_queue.empty())
    s = priority_queue.pop()
    if(s == goal)
        return
    if (!s.visited)
      • s.visited=true
      • for all actions a∈A
        s'=τ(s,a)
        s'_path_length = s_path_length + length(a)
        priority_queue.push(s', s'_path_length + h(s'))
```

RUTGERS

Admissible Heuristics

- A heuristic is admissible if it never overestimates the cost from a node to the goal
  - $h(s) \leq cost(path(s,s_{goal}))$
  - Multiple goals or paths to goal
    - h(s) less than minimum cost over all paths to goals
    - Example: Find a path to the nearest post office
      - Doesn't matter what post office you get to
      - h(s) must be less then cost of shortest path to nearest post office
- Examples
  - Robot in plane
    - Euclidean distance
  - Robot in grid
    - Manhattan distance
  - Mahjong Solitaire
    - 1/2 * number of tiles left

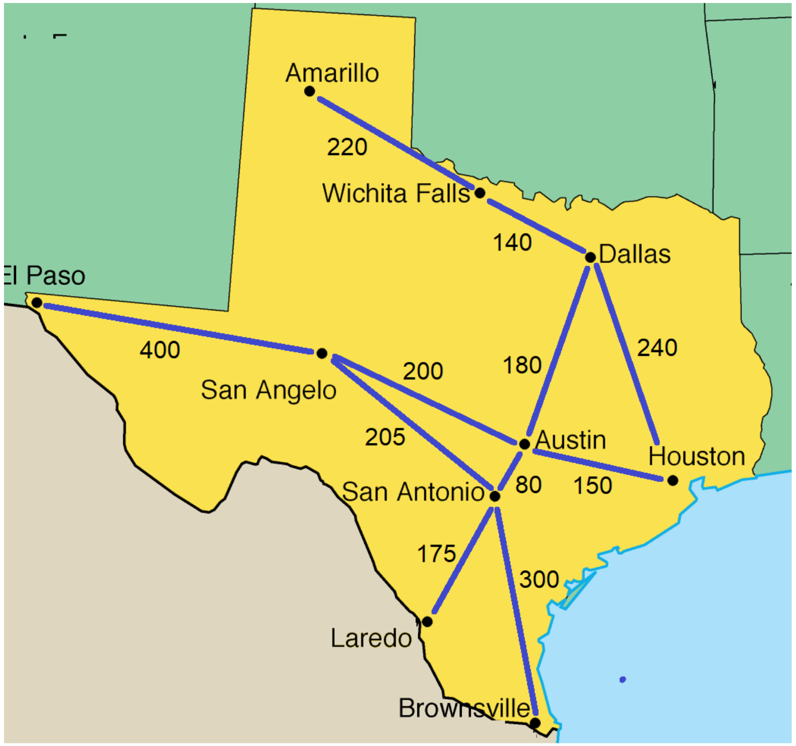


Admissible Heuristics

- Proof that search with admissible heuristic will always return optimal path to goal
- Assumptions
  - All actions have positive cost
  - h(s) is admissible
    - $h(s) \leq cost(path(s,s_{goal}))$
  - Search expands nodes using priority queue ordered by g(s)
    - $g(s) = cost(path(s_{initial},s)) + h(s)$

A* Search Example

- Map of driving routs
  - Find shortest rout
  - Heuristic = Direct Euclidean distance to goal



City	h(s) = direct distance to Amarillo
Wichita Falls	220
Dallas	330
Austin	505
Houston	605
San Antonio	445
San Angelo	315
El Paso	440
Laredo	670
Brownsville	795

A* for continuous state space

How can we apply A* to problems with continuous state spaces?

- Robot in XY-plane

A* for continuous state space

- Generate graph in space of problem
- Grid up state space
  - What resolution to use
    - More cells $\Rightarrow$ higher computation cost
  - Complexity exponential with respect to dimension of state space
    - $O(x^d)$
    - Curse of dimensional
- Generate graph in space of problem
  - Example robotics
- Adaptive grid
  - Recursively subdivide cells that are interesting
  - Example: Robotics - use higher resolution cells in difficult regions