Artificial Intelligence
Greedy and A* Search

Portland Data Science Group

Created by Andrew Ferlitsch
Community Outreach Officer

June, 2017

Greedy Algorithm

- A search method of selecting the best local choice at each step in hopes of finding an optimal solution.

- Does not consider how optimal the current solution is.

- At each step, uses a heuristic to estimate the distance (cost) of each local choice from the goal.

- Steps:
  1. Define a heuristic function h(x) to estimate the distance to the goal from any state.
  2. From the current state, determine the search space (actions) for one step ahead.
  3. Select the action from the search space that minimizes the heuristic function.

Greedy - Completeness

- The **Greedy** method will always find a solution if the heuristic function $h(x)$, for all x, is always less than or equal the actual distance (cost) to the goal node, $h^*(x)$.

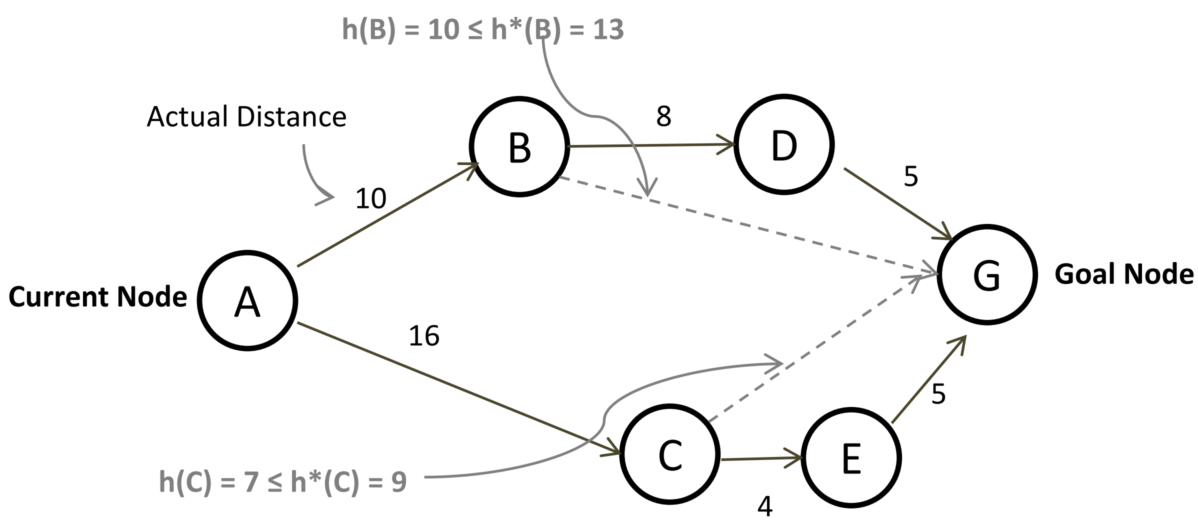
$$\underbrace{h(x)}_{\text{Estimate}} \leq \underbrace{h^*(x)}_{\text{Actual}}, \text{ for all } x$$

- When the heuristic $h(x) \leq h^*(x)$, the method is said to be complete, since a solution will be found.

Greedy – Straight Line Distance (SLD)

- One well-known heuristic that meets this criteria is the **Straight Line Distance (SLD)** in problems where the search space can be represented in a **Euclidean Space.**

$h_{SLD}(x)$

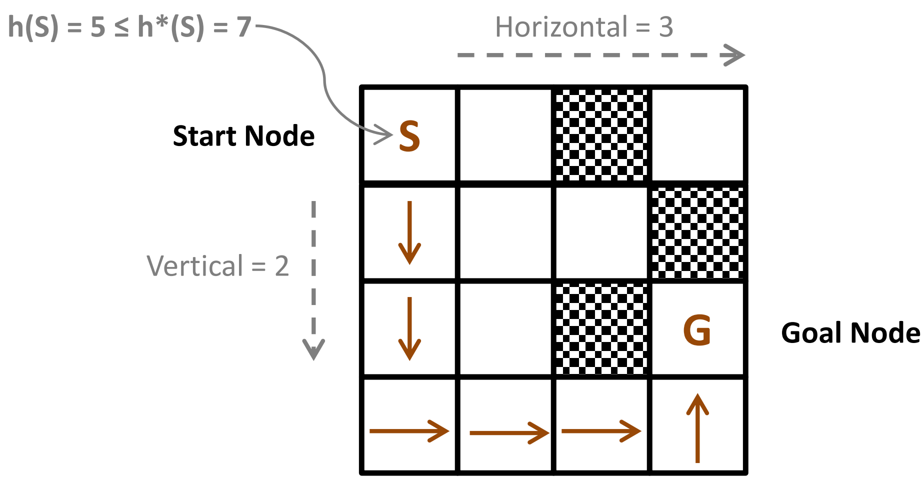


Greedy will select node C, since h(C) < h(B)

Greedy – Manhattan Distance

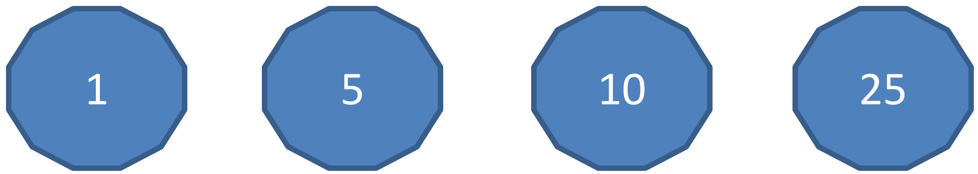
- In search spaces represented by a 2D grid (such as GridWorld), the **Manhattan Distance h_{MAN}** is a heuristic which meets the criteria of always been less than or equal to the actual distance.

h_{MAN} = the sum of the vertical and horizontal distance



Least Coin Problem – Greedy Search

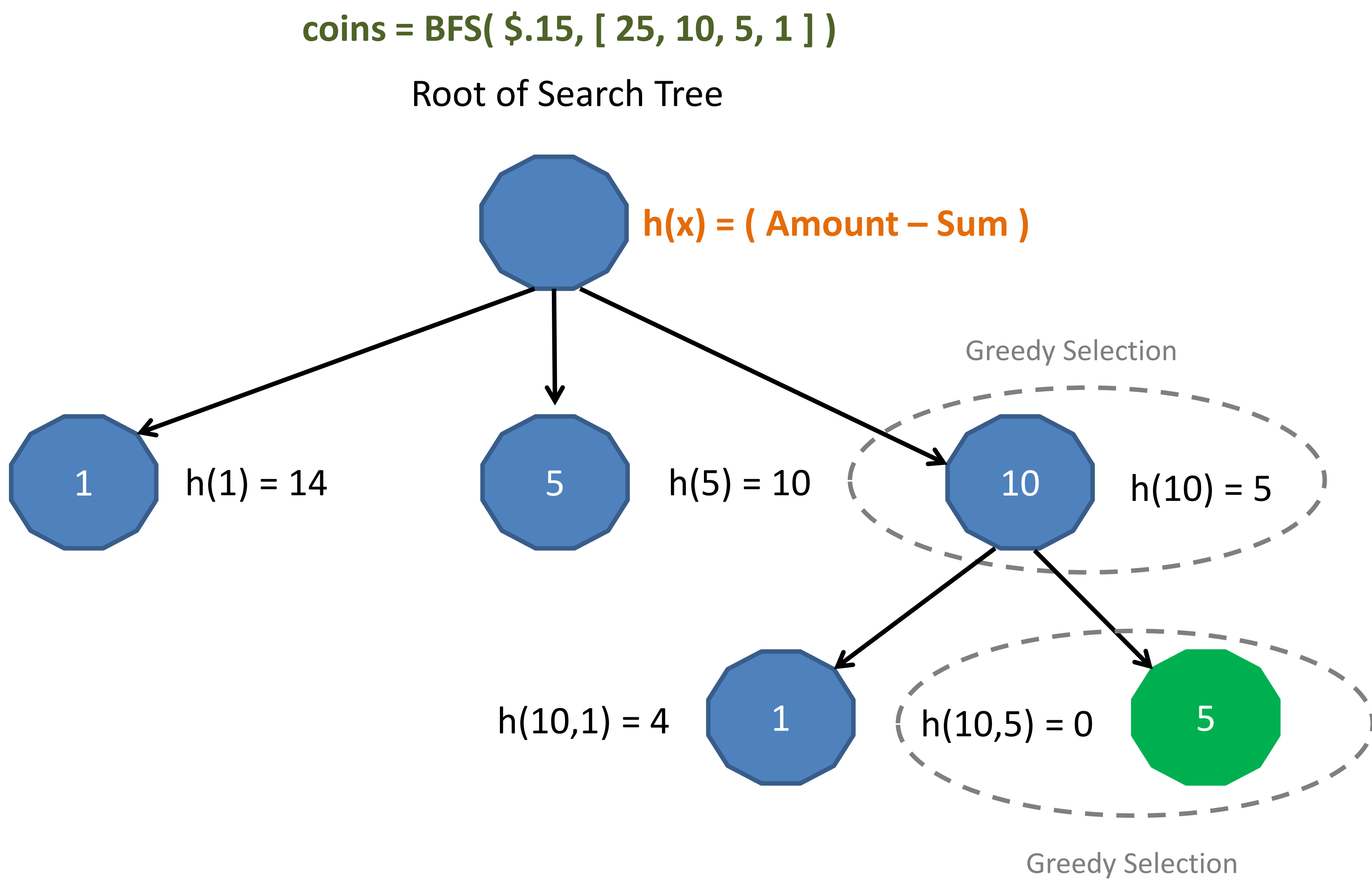
Problem: For any money amount, calculate the least number of coins to carry in your pocket.



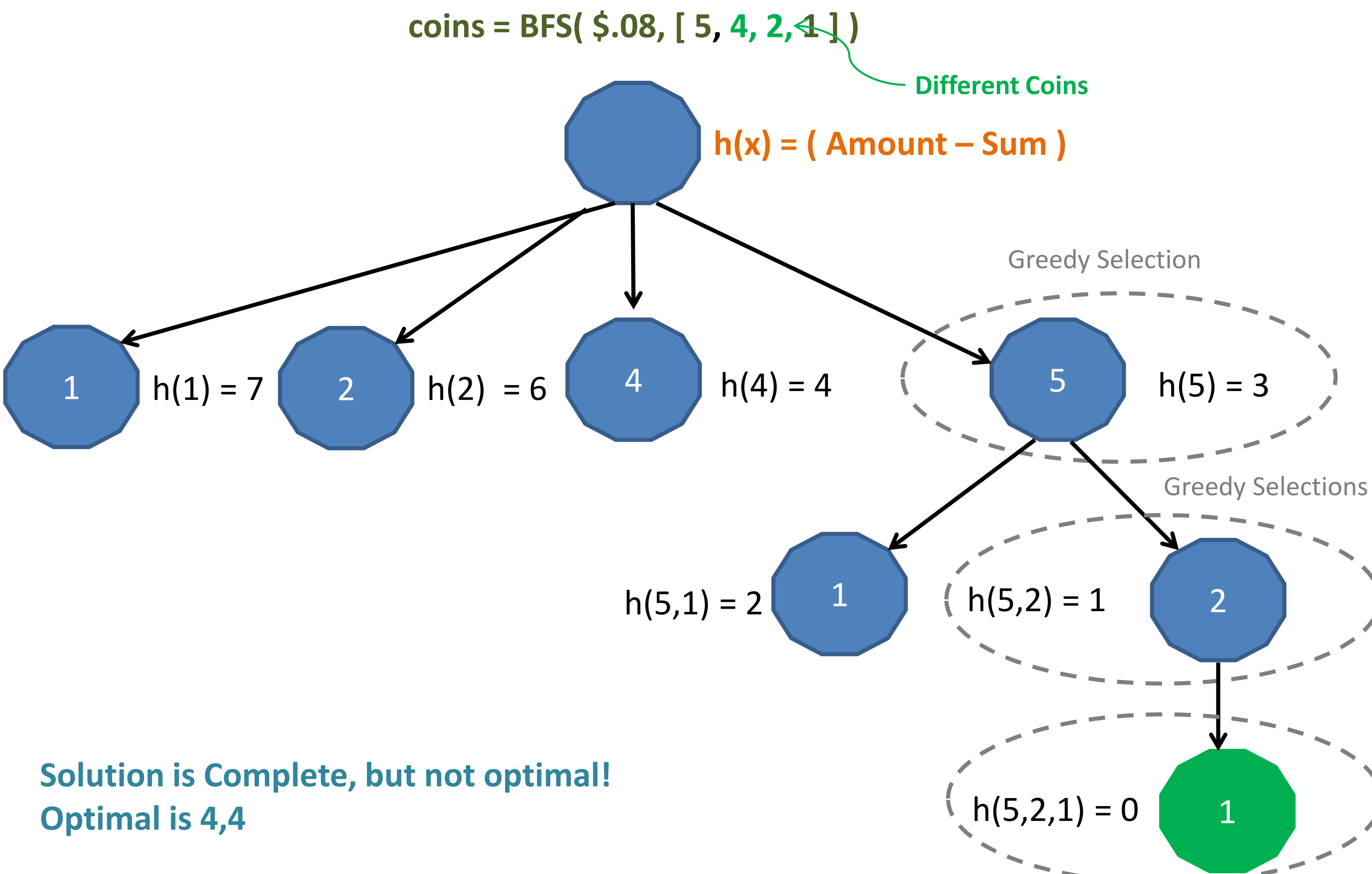
Greedy Selection, which coin gets the closet to the solution.

```
def Greedy(amount):
        sum = 0
        number_of_coins = 0
        while sum < amount:
                coin = largest_coin( amount – sum )
                sum = sum + coin
                number_of_coins = number_of_coins + 1
```

Greedy Search Example



Greedy Search Example



Solution is Complete, but not optimal!
Optimal is 4,4

BFS Greedy – Algorithm

function Greedy(root , goal)

Heap = Priority Queue

initialize the frontier to the root node (Heap) *Difference from BFS*

initialize the visited (explored) to the empty set (Heap)

while the frontier is not empty
    remove (delete min) the next *node* from the frontier
    add (insert) the (node removed from frontier) *node* to the visited

    if the *node* matches to goal node
        return found goal

    for each child (neighbor) of the *node*
        if child not in frontier or visited
            set key of child to h(child) *Difference from BFS*
            add (insert) child to the frontier

A* Algorithm

- A **search method** of selecting the best local choice at each step in hopes of finding an optimal solution.

- Does consider how optimal the current solution is.

- At each step, uses a heuristic h(x) to estimate the distance (cost) of each local choice from the goal, plus the accumulated distance (cost) g(x) to the current state .

- Steps:
  1. Define a heuristic function h(x) to estimate the distance to the goal from any state.
  2. Define a function g(x) for measuring the actual distance from the start state to the current state.
  2. From the current state, determine the search space (actions) for one step ahead.
  3. Select the action from the search space that minimizes g(x) + h(x).

A* - Optimality

- **The A* method will always find the optimal solution if the heuristic function $h(x)$, for all x, is always less than or equal the actual distance (cost) to the goal node, $h^*(x)$.**

$$h(x) \leq h^*(x), \text{ for all } x$$

Estimate

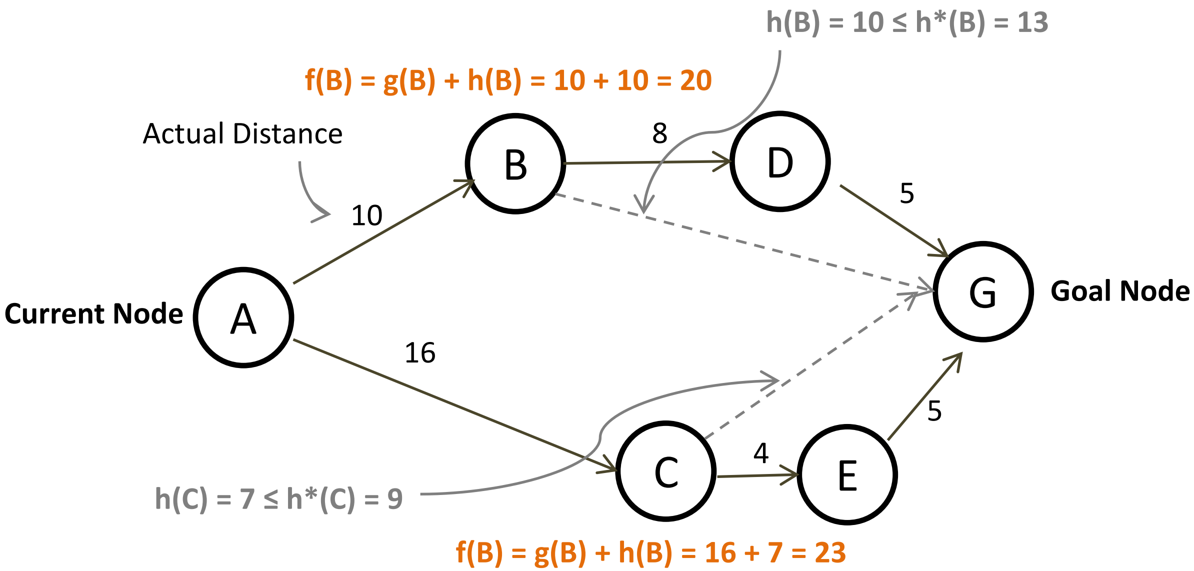
Actual

- **When the heuristic $h(x) \leq h^*(x)$, and the evaluation function is $f(x) = g(x) + h(x)$, the method is said to be optimal, since it will always find the optimal solution.**

A* – SLD Example

- **In the previous SLD example, Greedy picked a suboptimal solution (node B), while A* will pick an optimal solution (node C).**

$$f(x) = g(x) + h_{SLD}(x)$$



A* will select node B, since g(B) + h(B) < g(C) + h(C)

BFS A* – Algorithm

Heap = Priority Queue

function AStar(root , goal)

initialize the <u>frontier</u> to the <u>root node</u> (Heap) *Difference from BFS*

initialize the <u>visited</u> (explored) to the empty set (Heap)

while the <u>frontier</u> is not empty

remove (delete min) the next *<u>node</u>* from the <u>frontier</u>

add (insert) the (node removed from <u>frontier</u>) *<u>node</u>* to the <u>visited</u>

if the *<u>node</u>* matches to <u>goal node</u>

return <u>found goal</u>

for each <u>child</u> (neighbor) of the *<u>node</u>*

if <u>child</u> not in <u>frontier</u> or <u>visited</u>

set key of <u>child</u> to g(node) + h(child) *Difference from BFS*

add (insert) <u>child</u> to the <u>frontier</u>