Heuristic Optimisation

Part 6: A* search

Sándor Zoltán Németh

http://web.mat.bham.ac.uk/S.Z.Nemeth

s.nemeth@bham.ac.uk

University of Birmingham

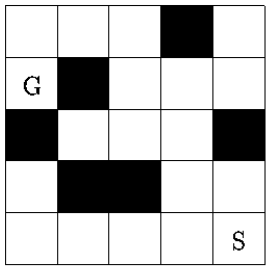
Overview

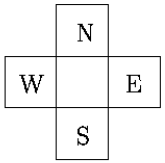
- Best-first search
- Minimizing the total cost $\Rightarrow$ A*
- Two examples of A*
- The algorithm of A*
- Observations

Best-first search

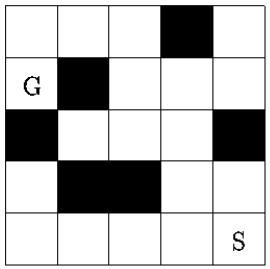
- We follow only one path at a time, but when a competing path looks more promising, we switch to that path.
- We apply a heuristic function to each of the generated nodes.

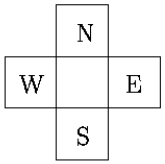
Best-first vs hill climbing



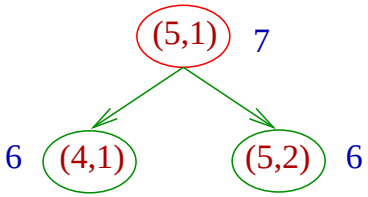


Best-first vs hill climbing

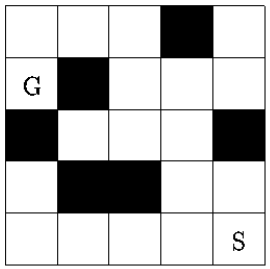


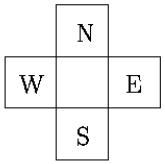


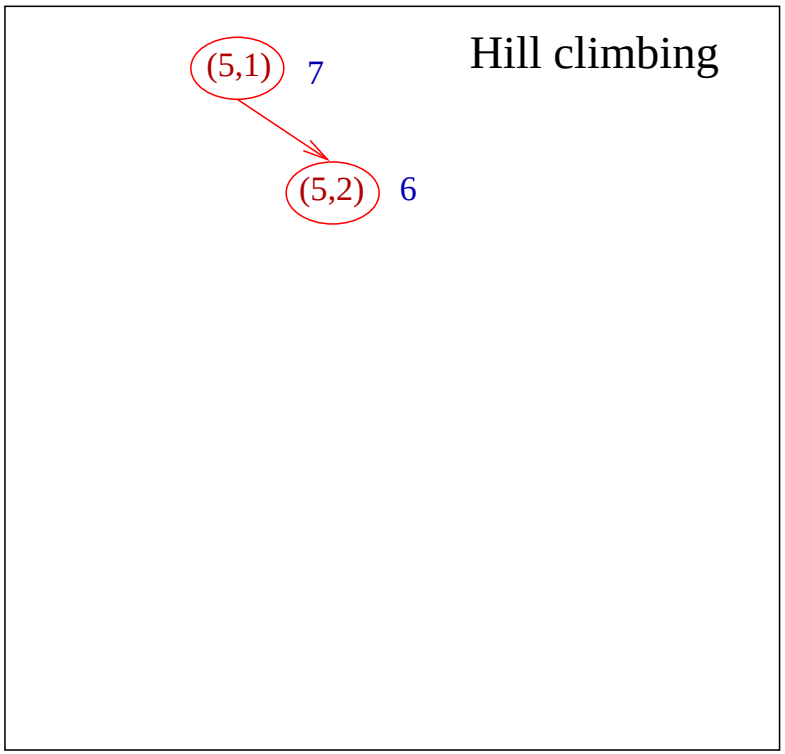
Hill climbing



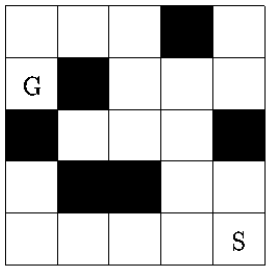
Best-first vs hill climbing

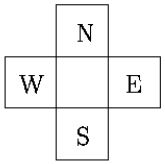


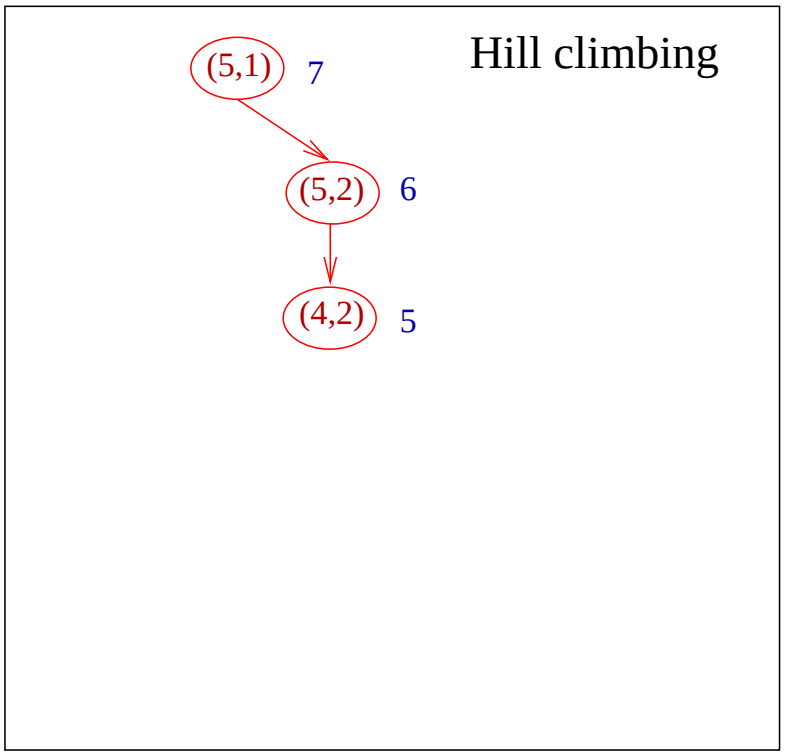




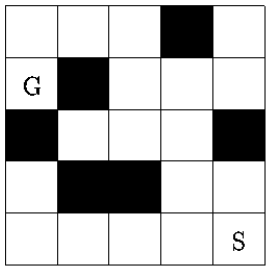
Best-first vs hill climbing

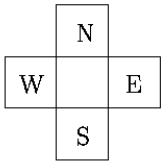


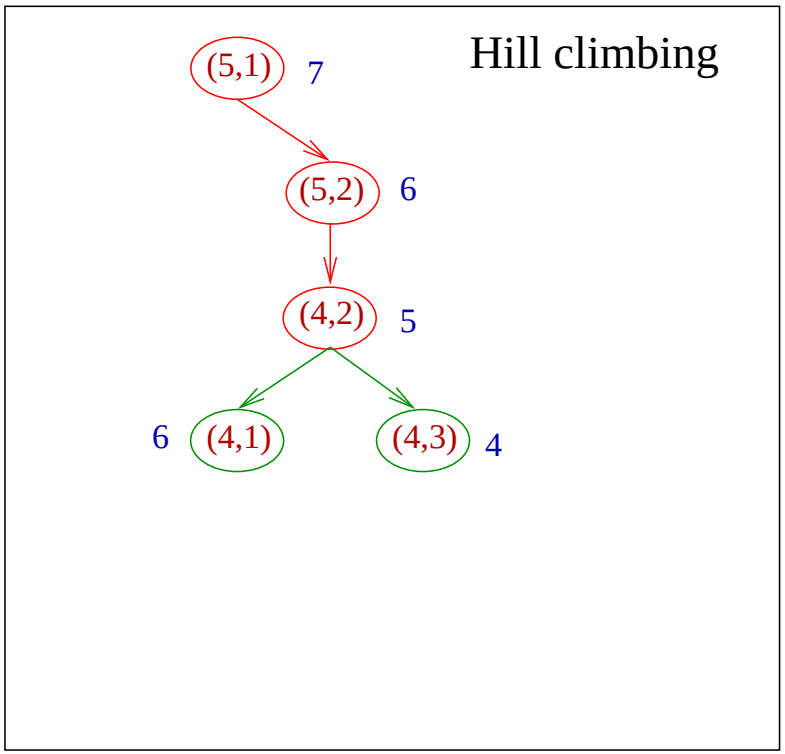




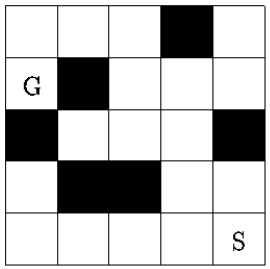
Best-first vs hill climbing

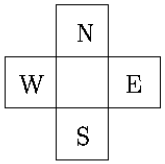


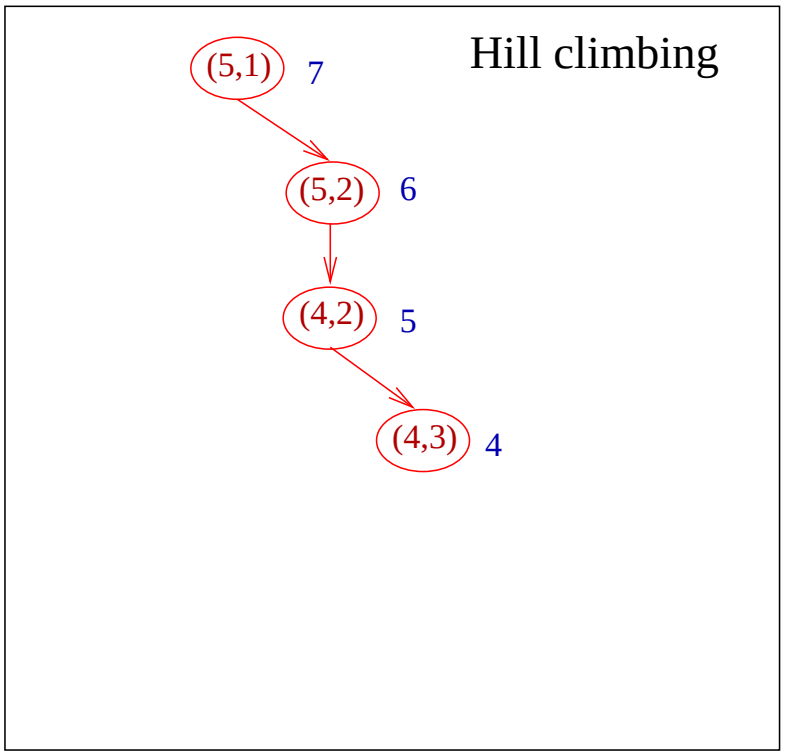




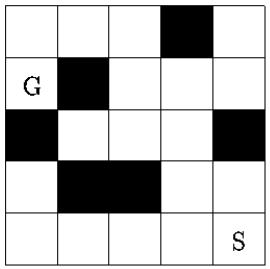
Best-first vs hill climbing

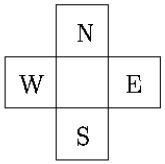


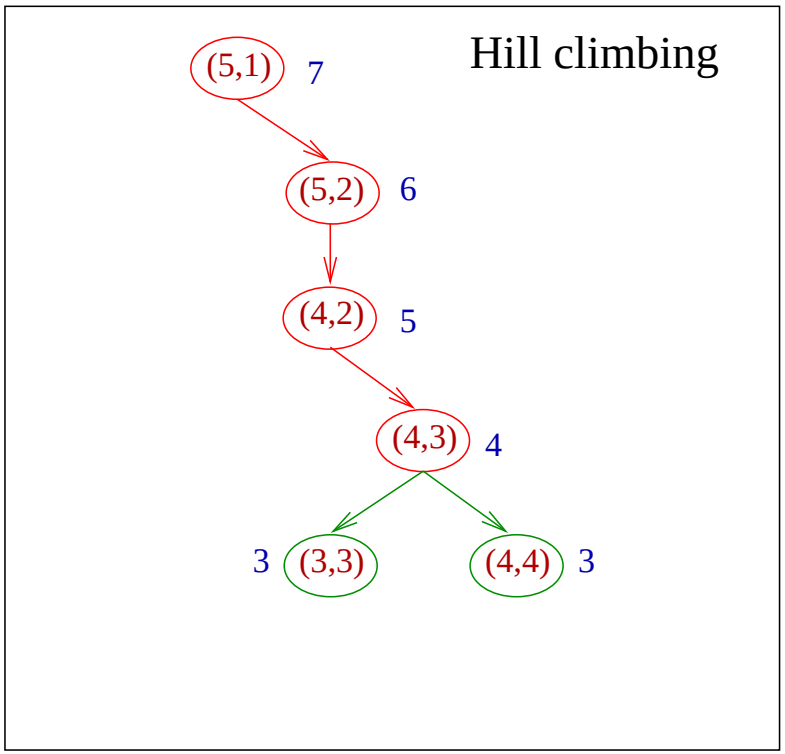




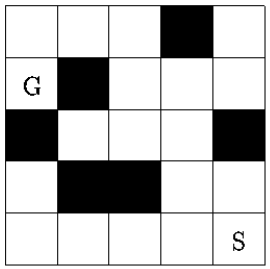
Best-first vs hill climbing

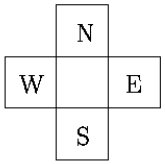


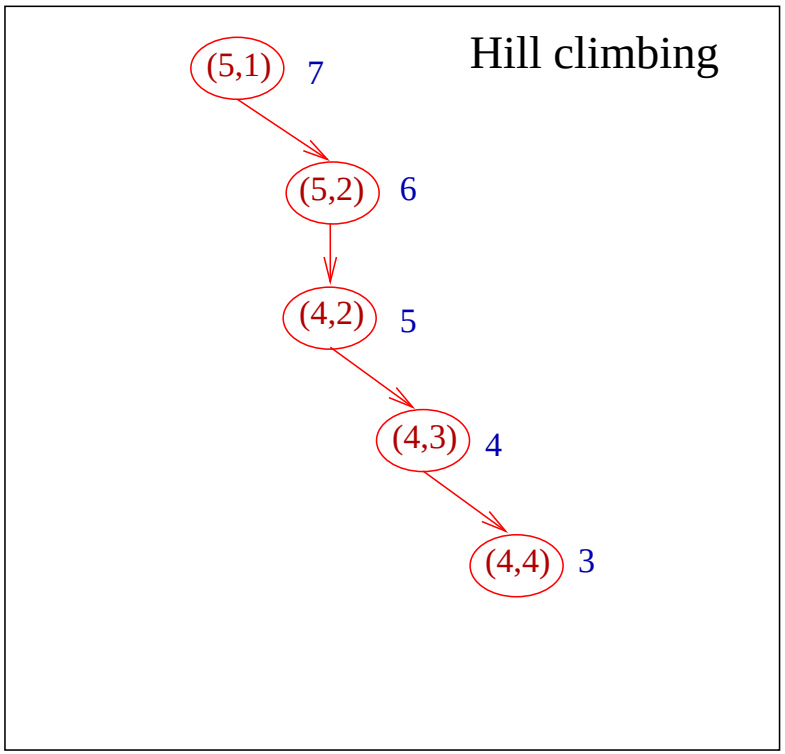




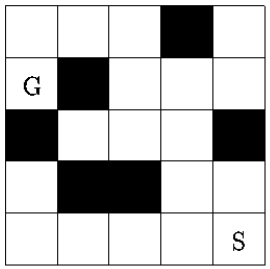
Best-first vs hill climbing

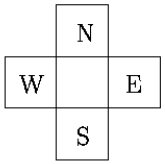


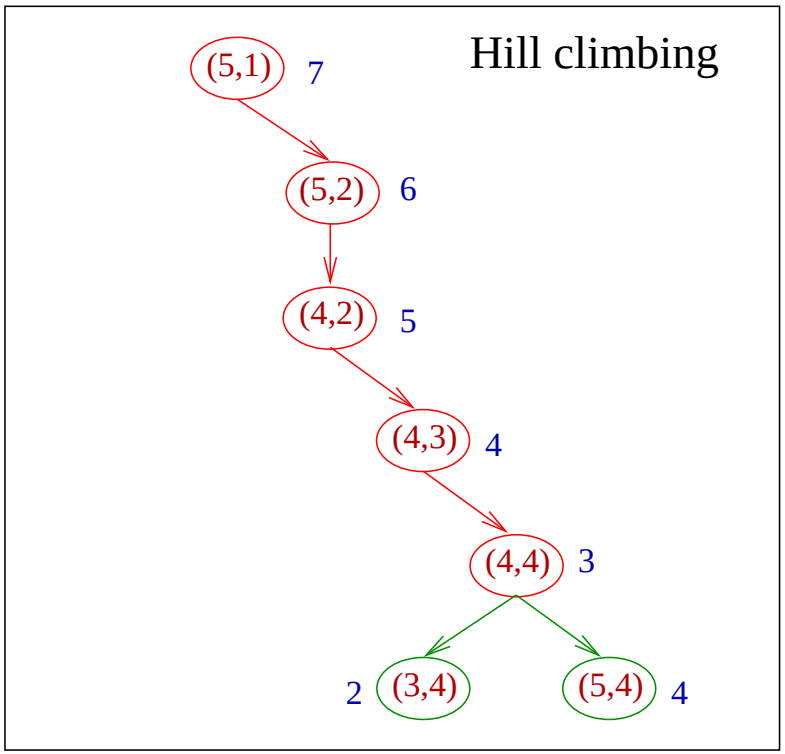




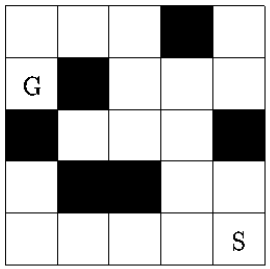
Best-first vs hill climbing

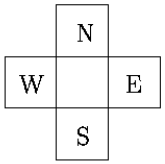


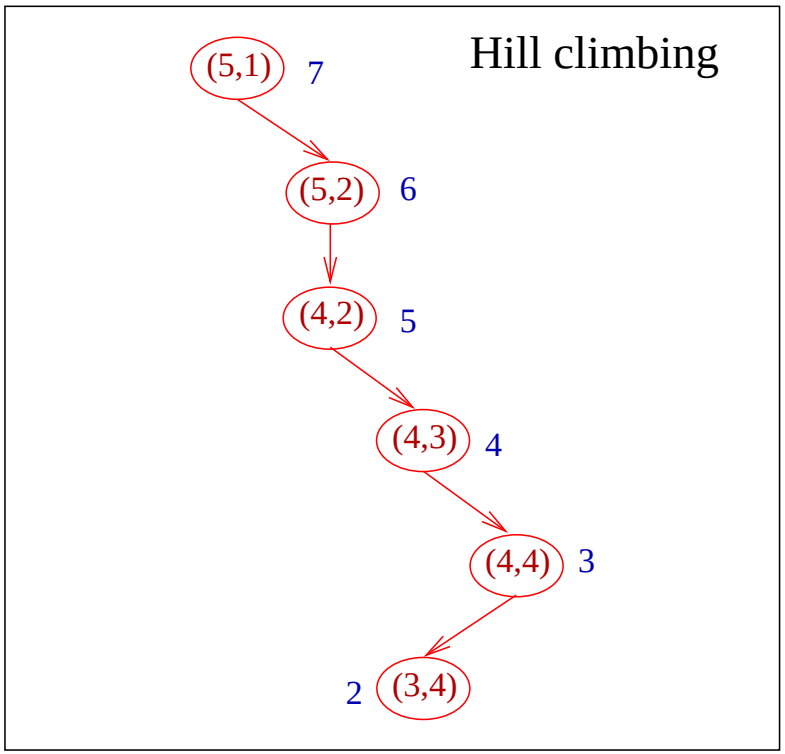




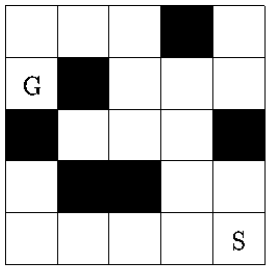
Best-first vs hill climbing

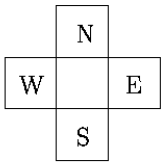


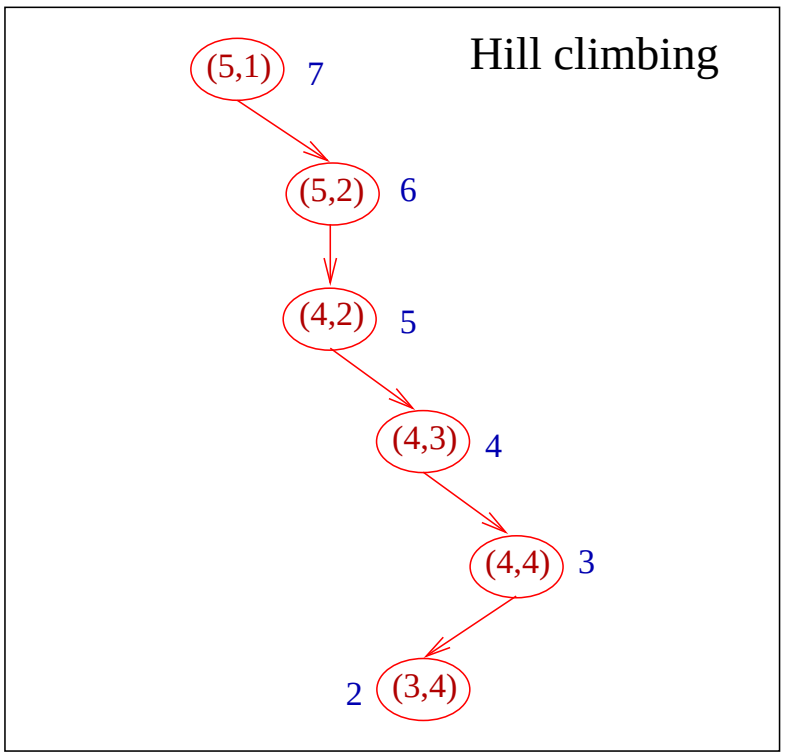


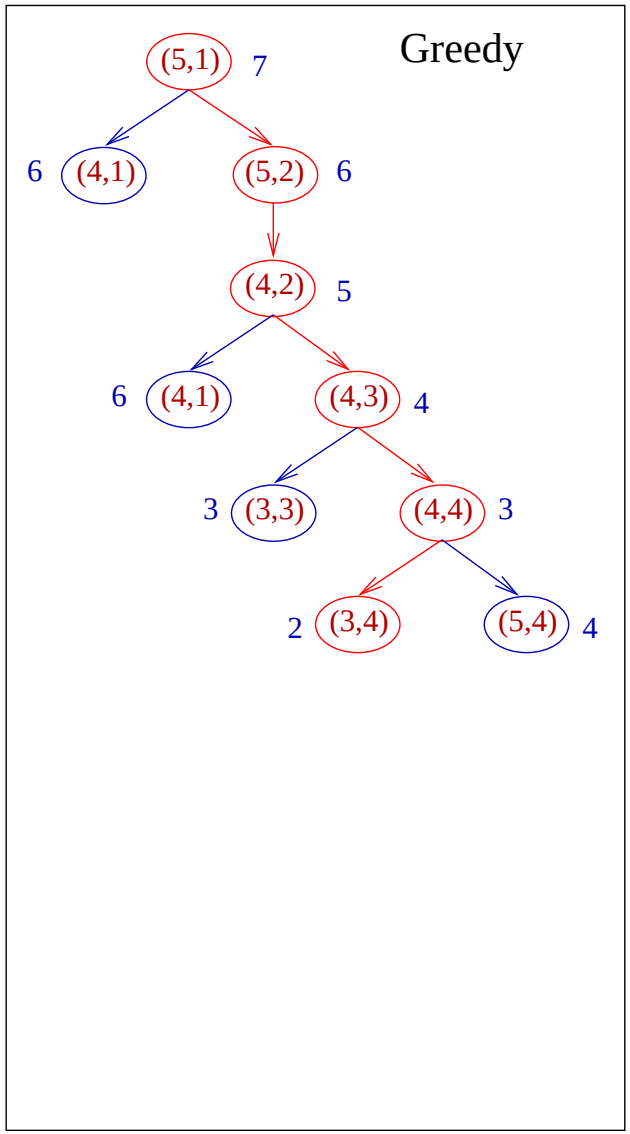


Best-first vs hill climbing

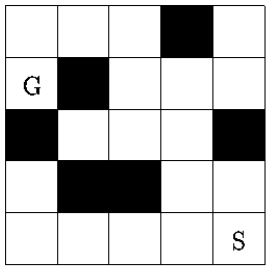


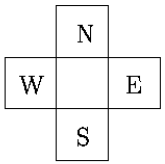


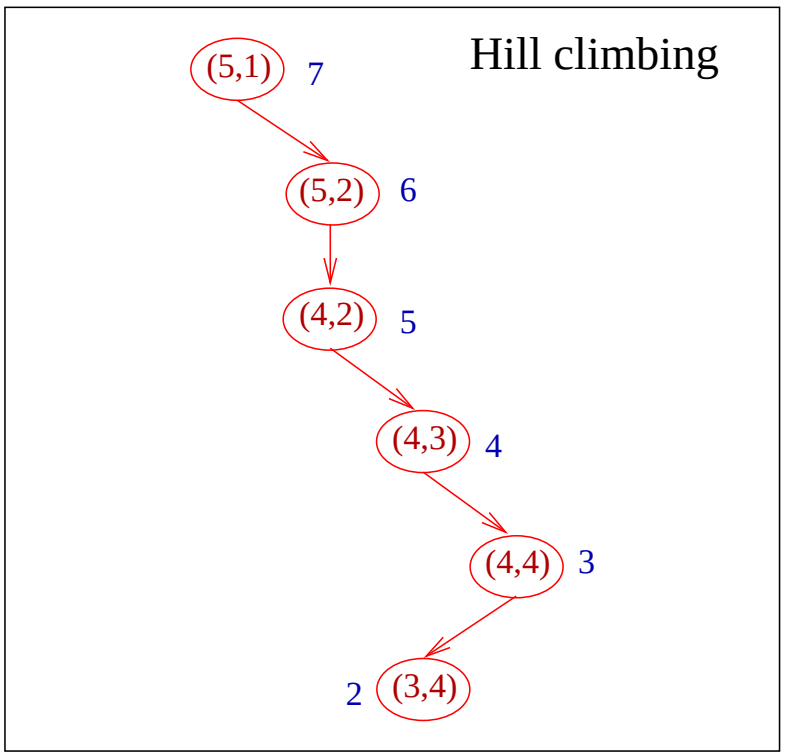


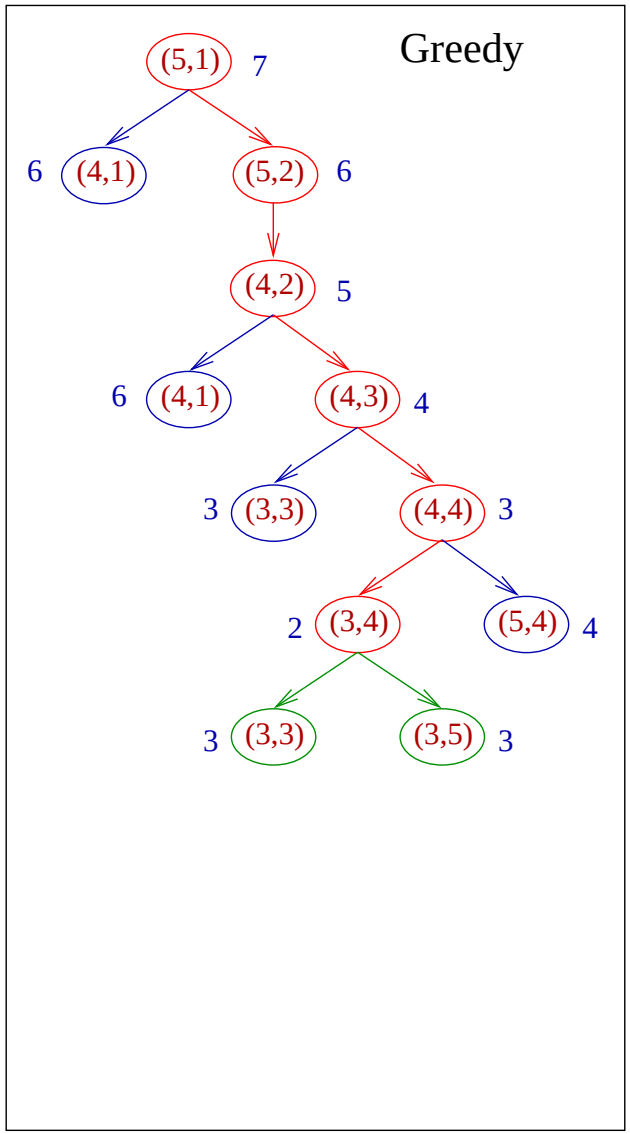


Best-first vs hill climbing

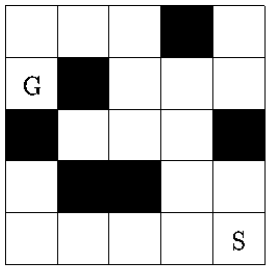


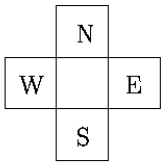


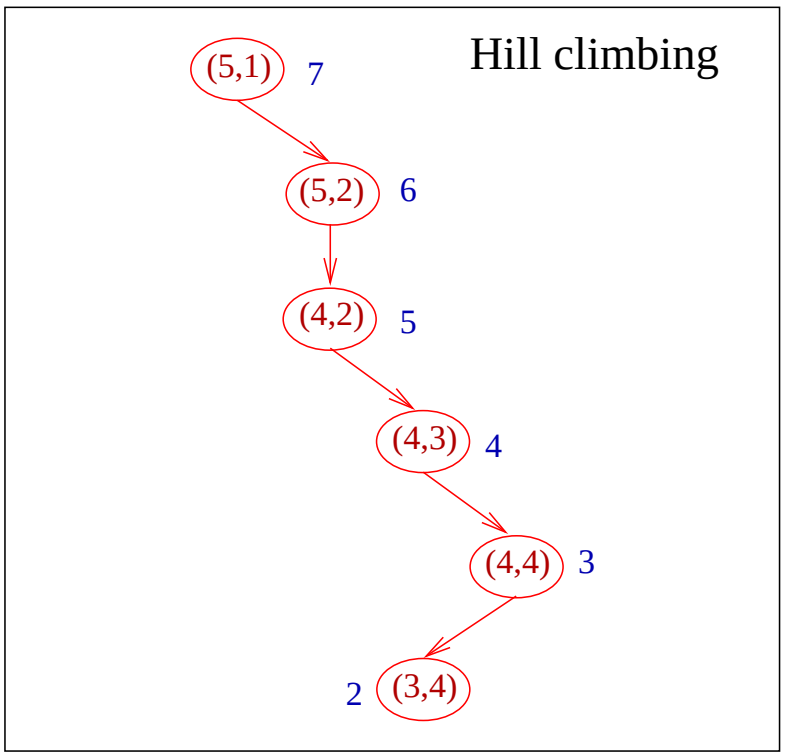


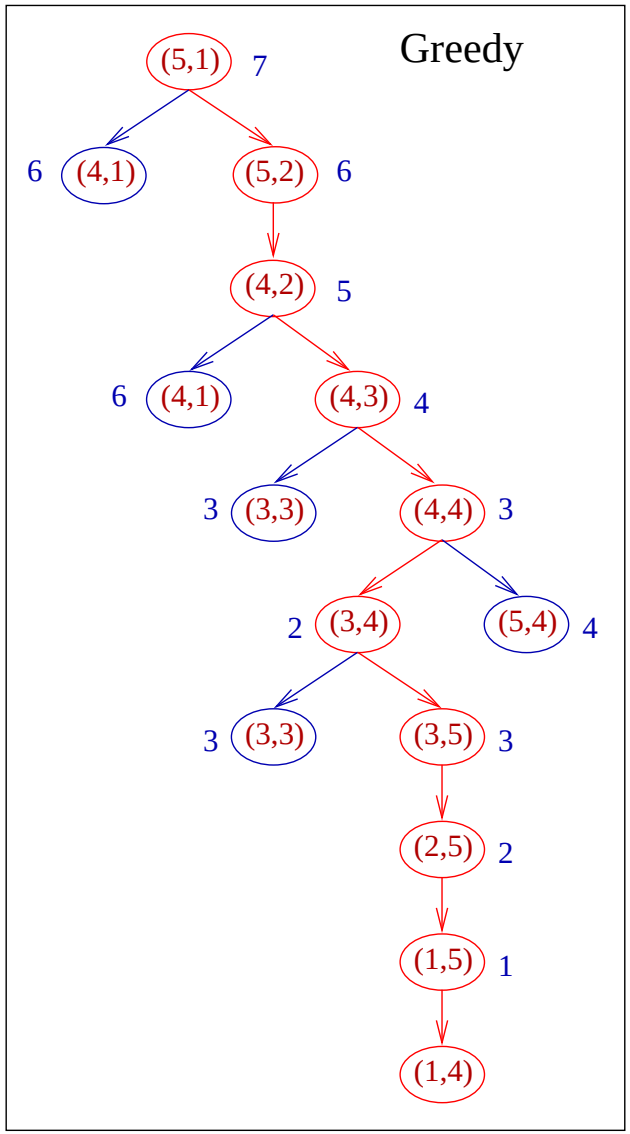


Best-first vs hill climbing









The underlying heuristic function

$$f(n) = g(n) + h(n)$$

$g(n)$ is a measure of the cost of getting from the initial state to the current state n.

$h(n)$ is an estimate of the cost of getting from the current state n to the goal state.

The underlying heuristic function

$$f(n) = g(n) + h(n)$$

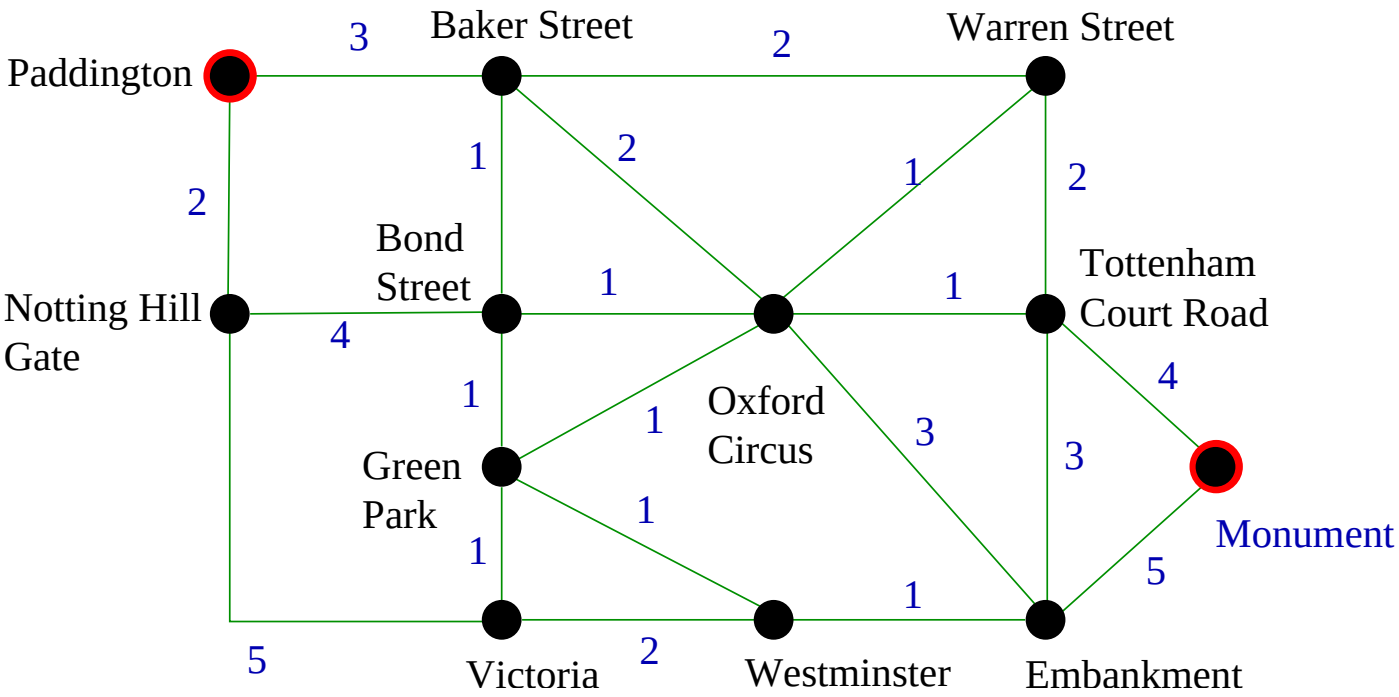
$g(n)$ is a measure of the cost of getting from the initial state to the current state n.

$h(n)$ is an estimate of the cost of getting from the current state n to the goal state.

What is a good value of h?
(large or low?)

Example 1

Underground routes in London (fragment of the map)



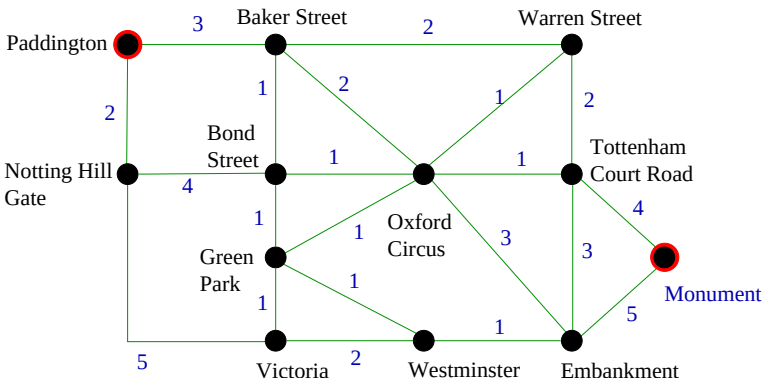
The estimated distances from Monument (h):

P	BaS	WaS	NHG	BoS	OC	TCR	GP	V	We	E	M
7	5	4	7	5	5	3	5	5	4	3	0

Example 1 – A* solution

P

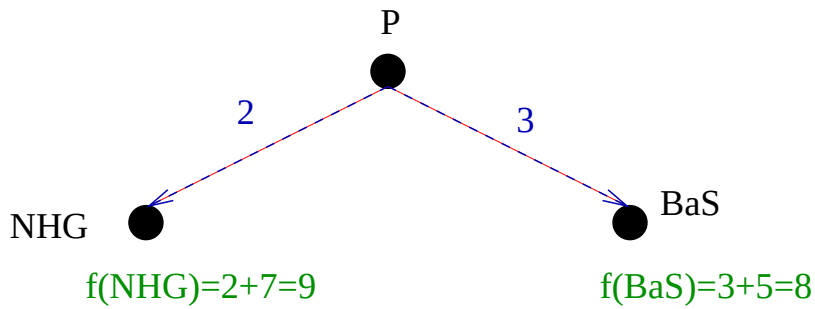
f(P)=0+7=7

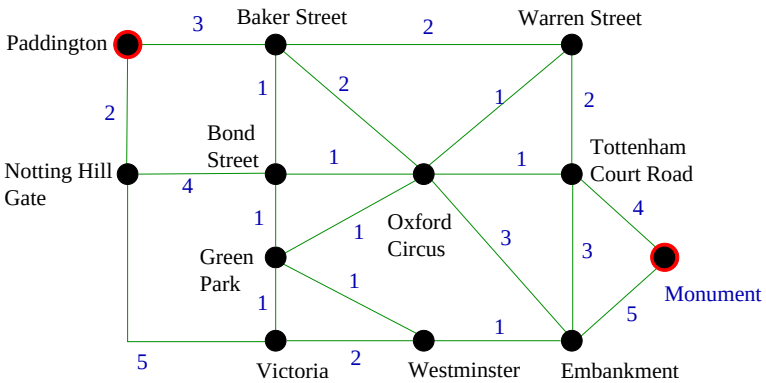


OPEN=[P]

CLOSED=[]

Example 1 – A* solution

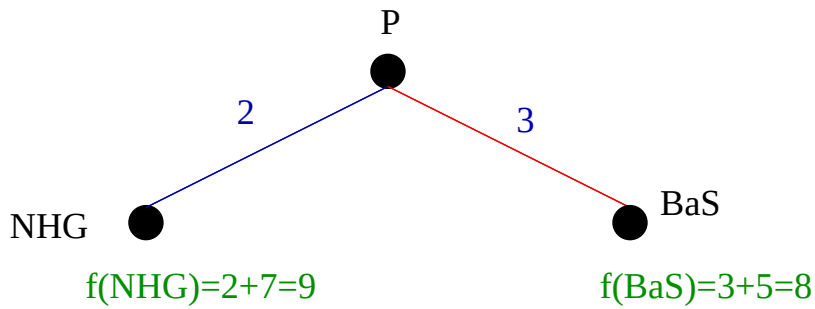


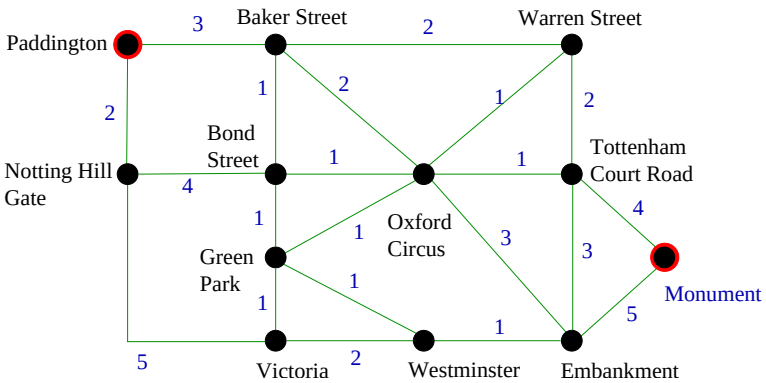


OPEN=[BaS,NHG]

CLOSED=[P]

Example 1 – A* solution

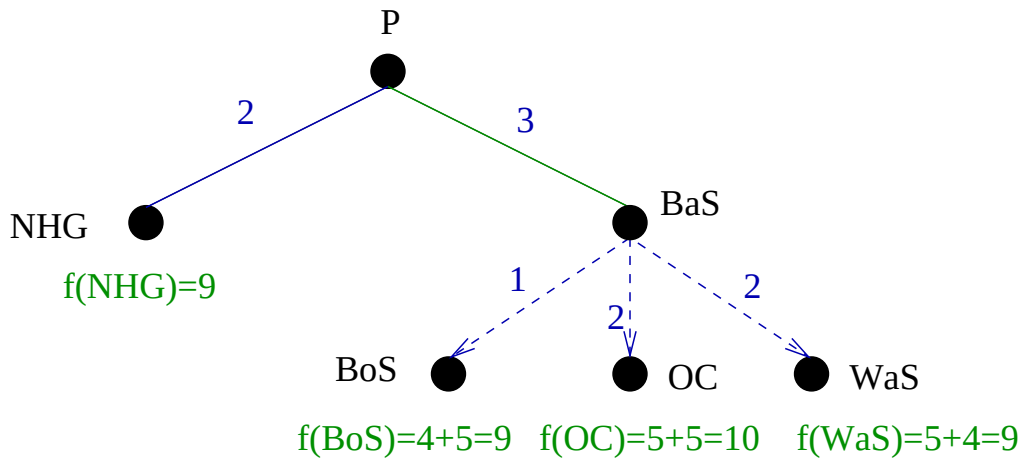


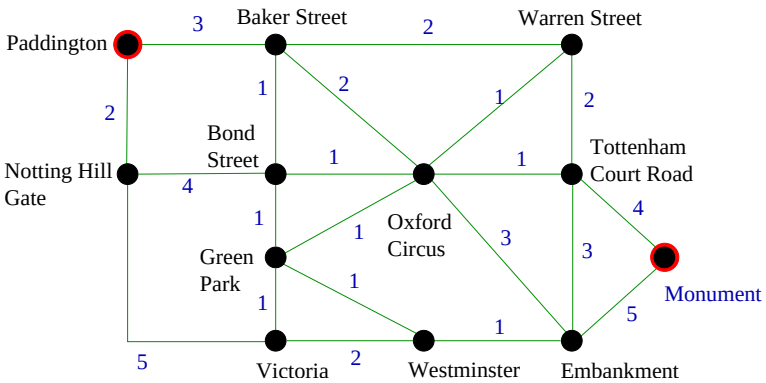


OPEN=[BaS,NHG]

CLOSED=[P]

Example 1 – A* solution

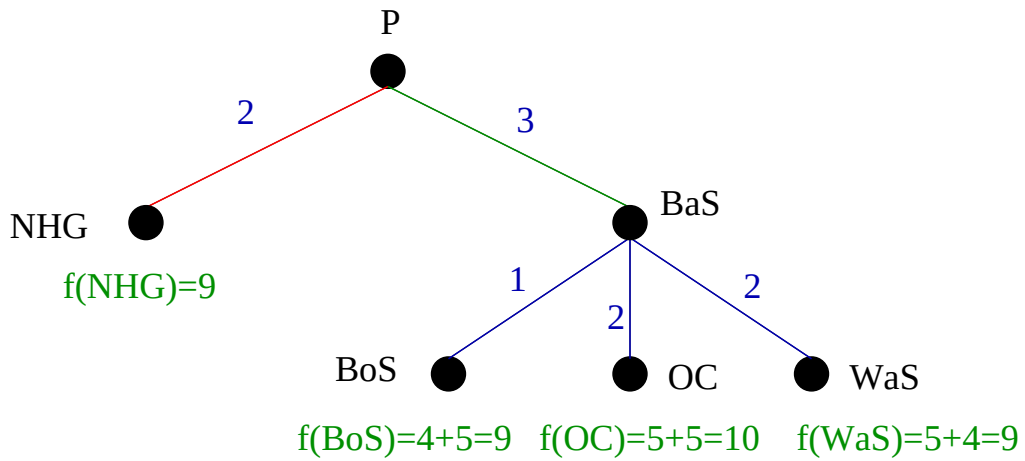


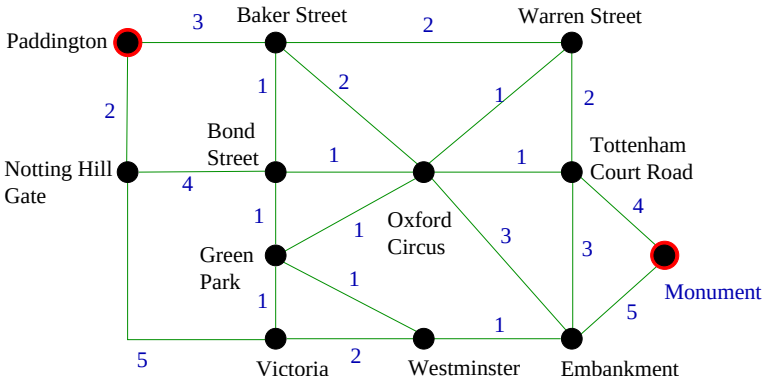


OPEN=[NHG, BoS,WaS,OC]

CLOSED=[P,BaS]

Example 1 – A* solution

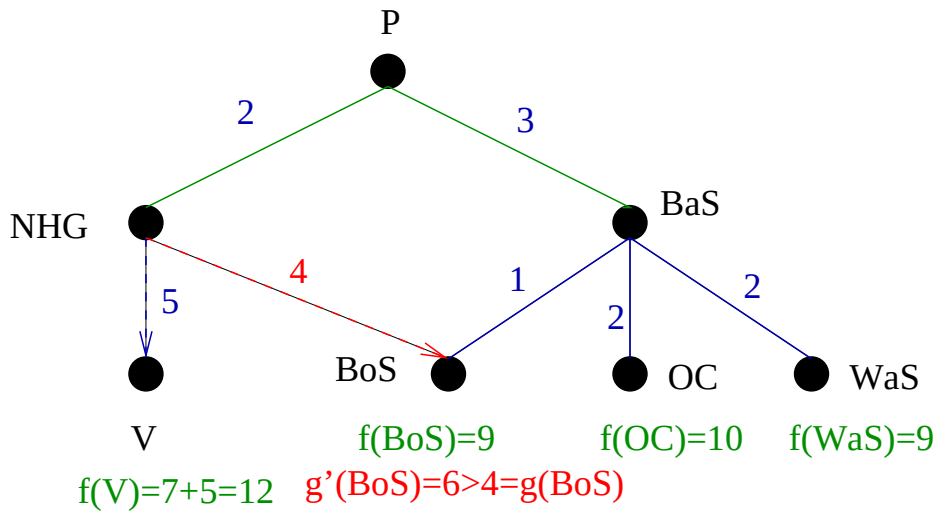


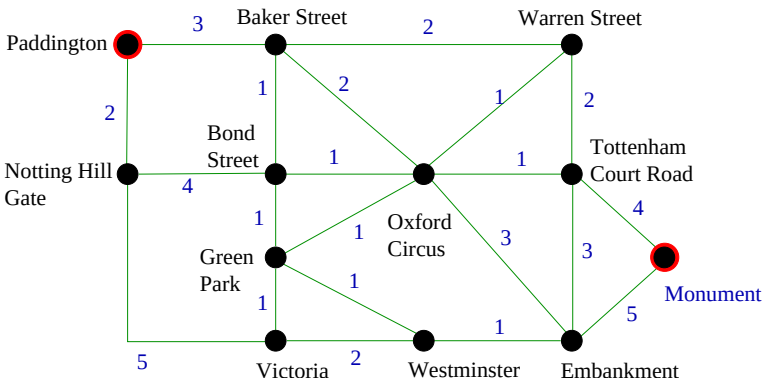


OPEN=[NHG, BoS,WaS,OC]

CLOSED=[P,BaS]

Example 1 – A* solution

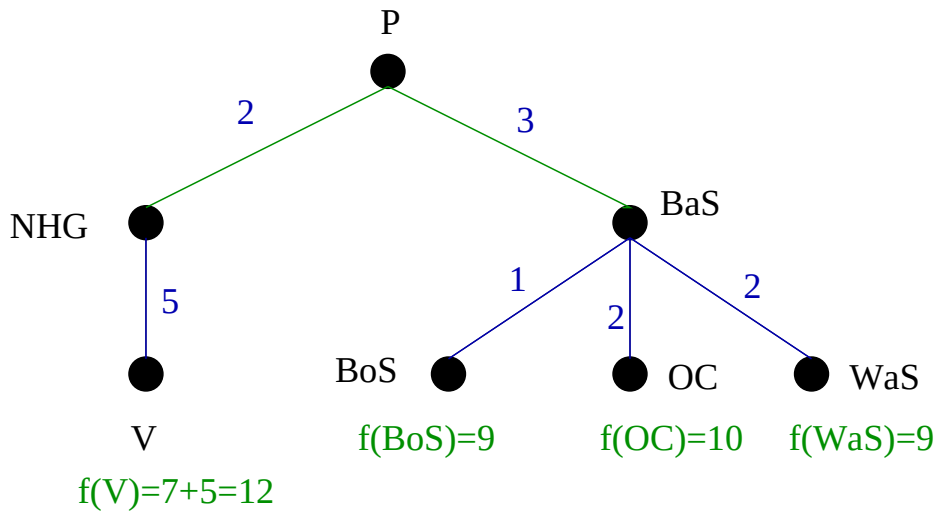


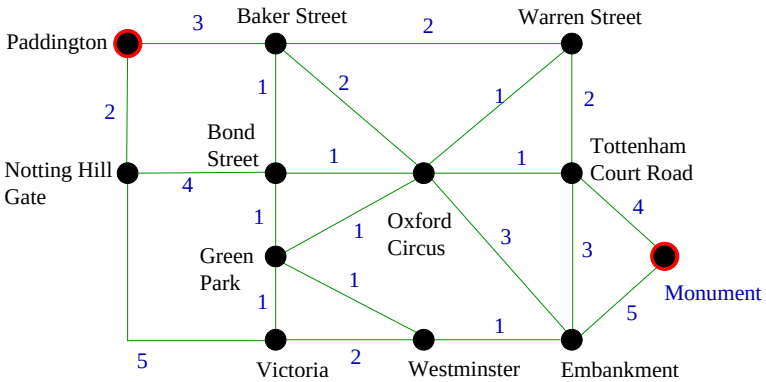


OPEN=[BoS,WaS,OC,V]

CLOSED=[P,BaS,NHG]

Example 1 – A* solution

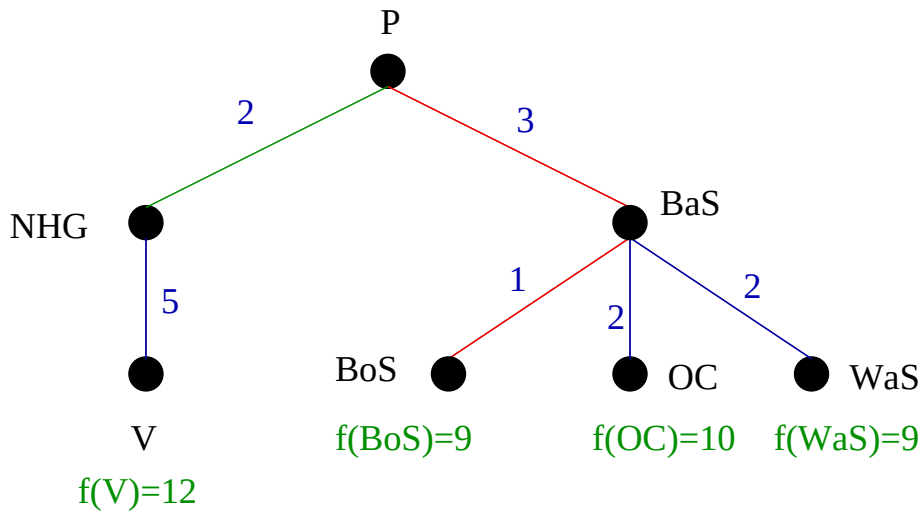


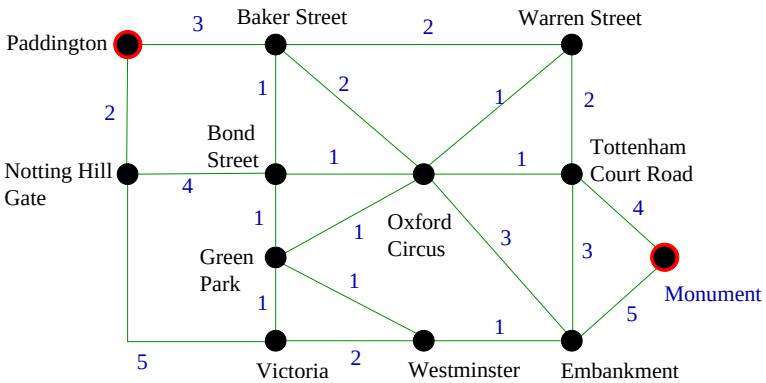


OPEN=[BoS,WaS,OC,V]

CLOSED=[P,BaS,NHG]

Example 1 – A* solution

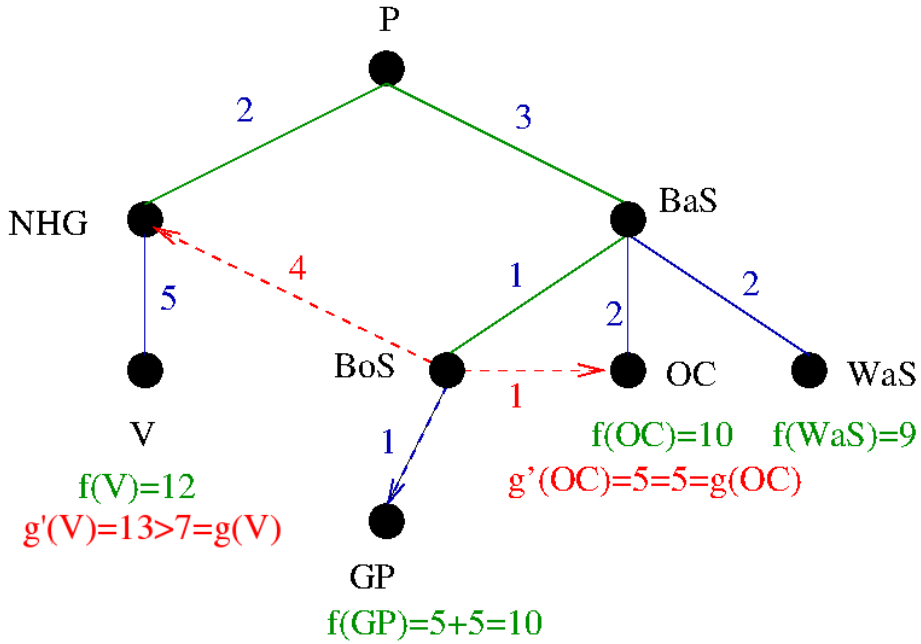


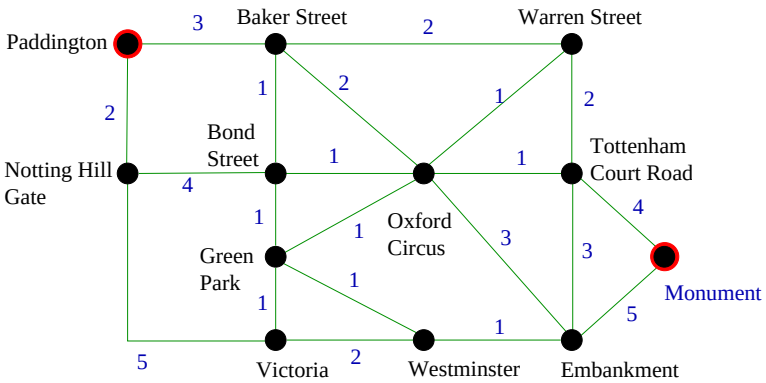


OPEN=[BoS,WaS,OC,V]

CLOSED=[P,BaS,NHG]

Example 1 – A* solution

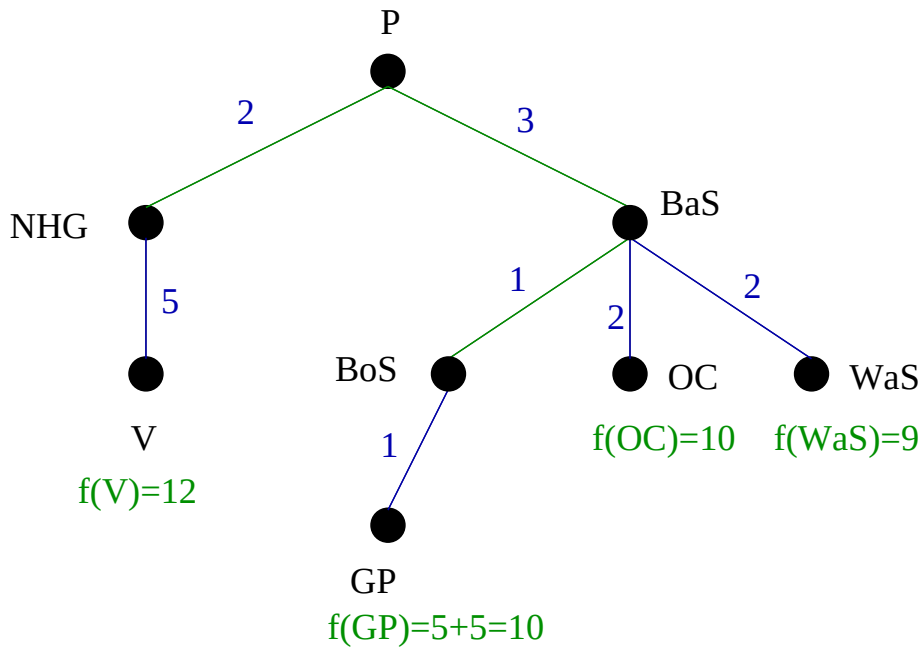


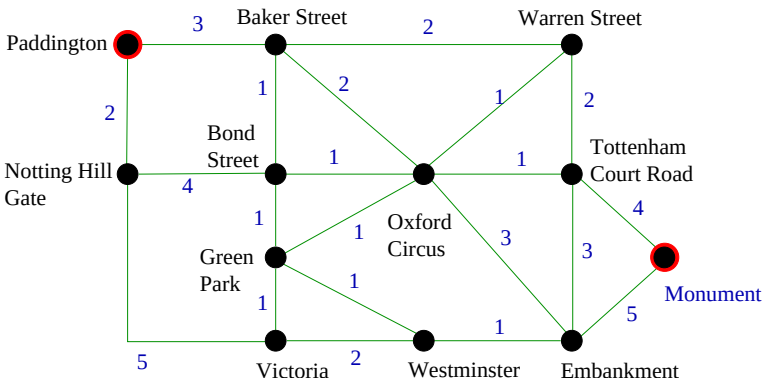


OPEN=[WaS,OC,GP,V]

CLOSED=[P,BaS,NHG,BoS]

Example 1 – A* solution

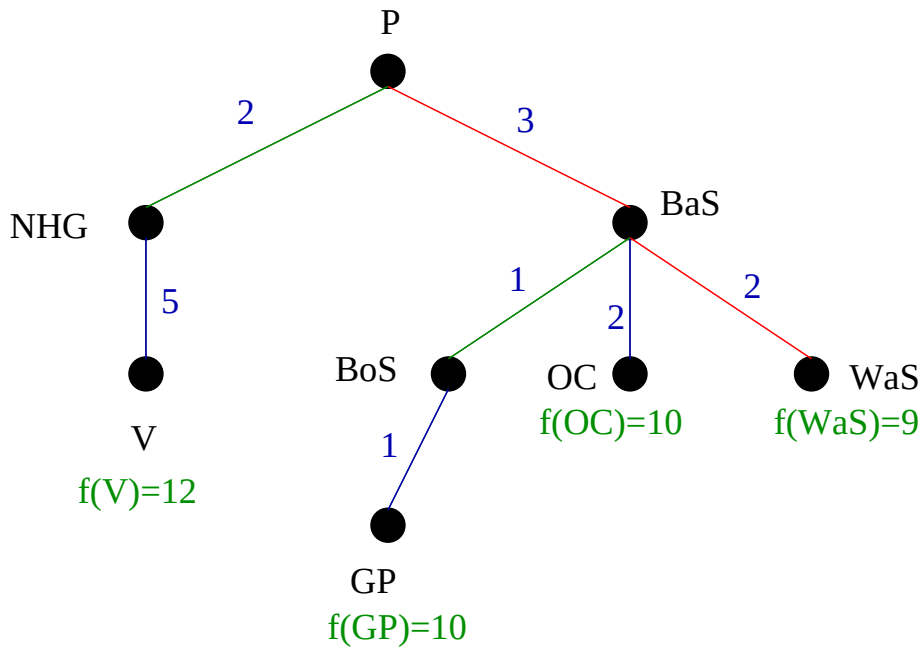


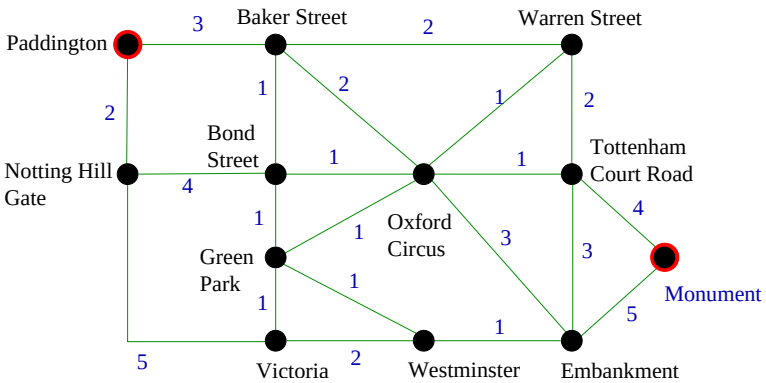


OPEN=[WaS,OC,GP,V]

CLOSED=[P,BaS,NHG,BoS]

Example 1 – A* solution

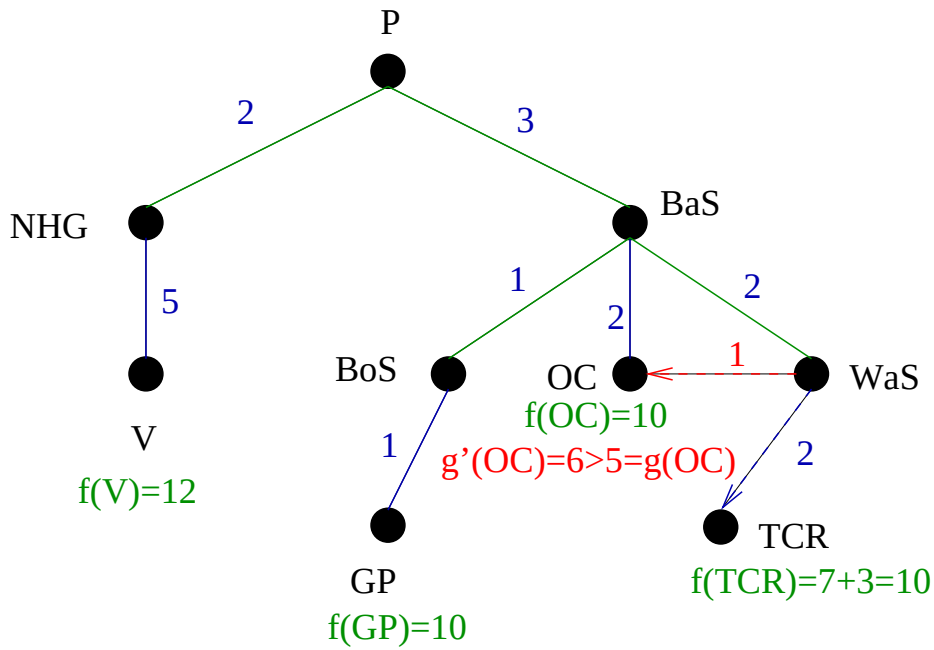


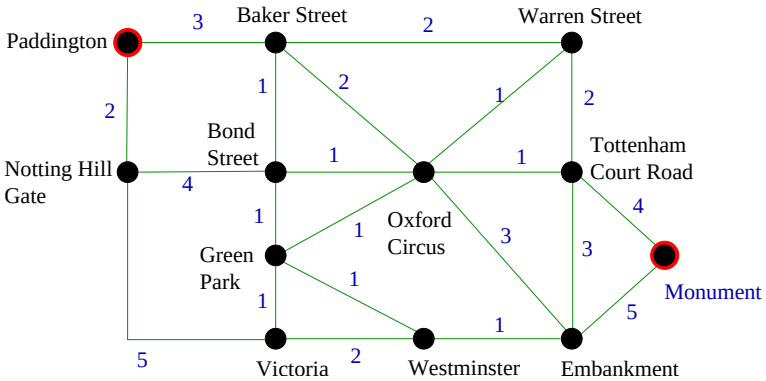


OPEN=[WaS,OC,GP,V]

CLOSED=[P,BaS,NHG,BoS]

Example 1 – A* solution

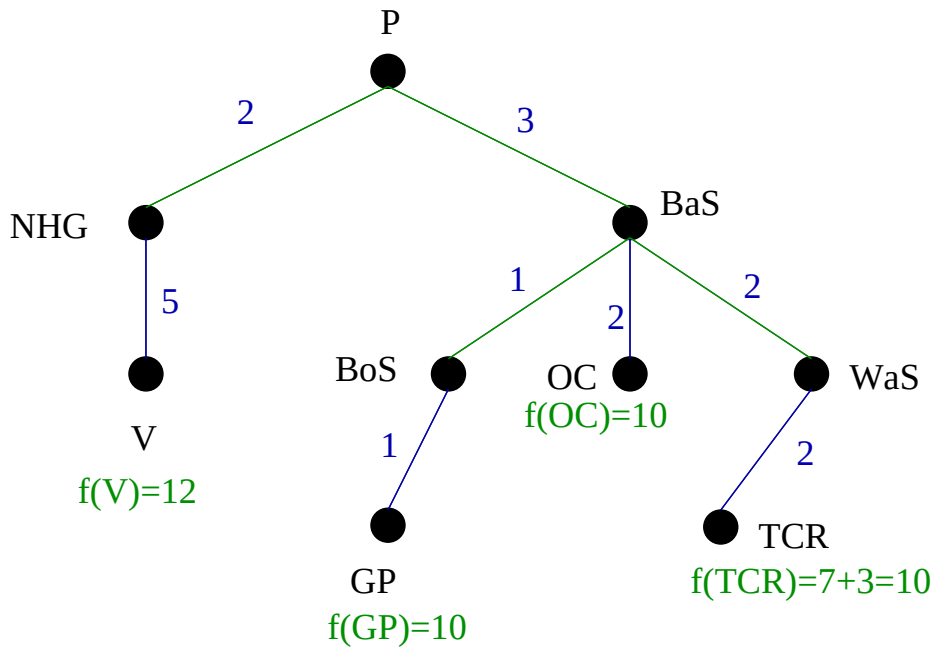


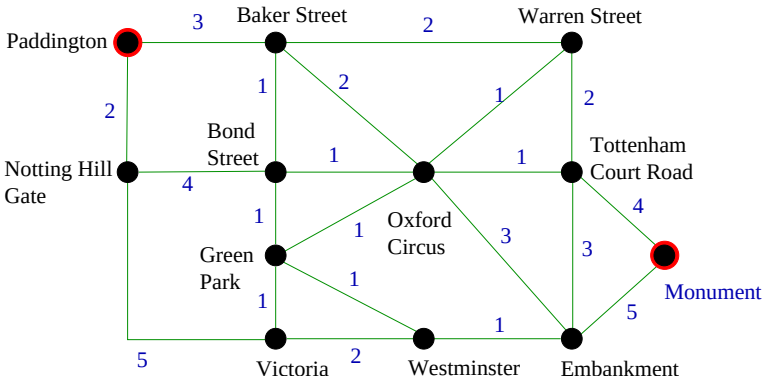


OPEN=[OC,GP,TCR,V]

CLOSED=[P,BaS,NHG,BoS,WaS]

Example 1 – A* solution

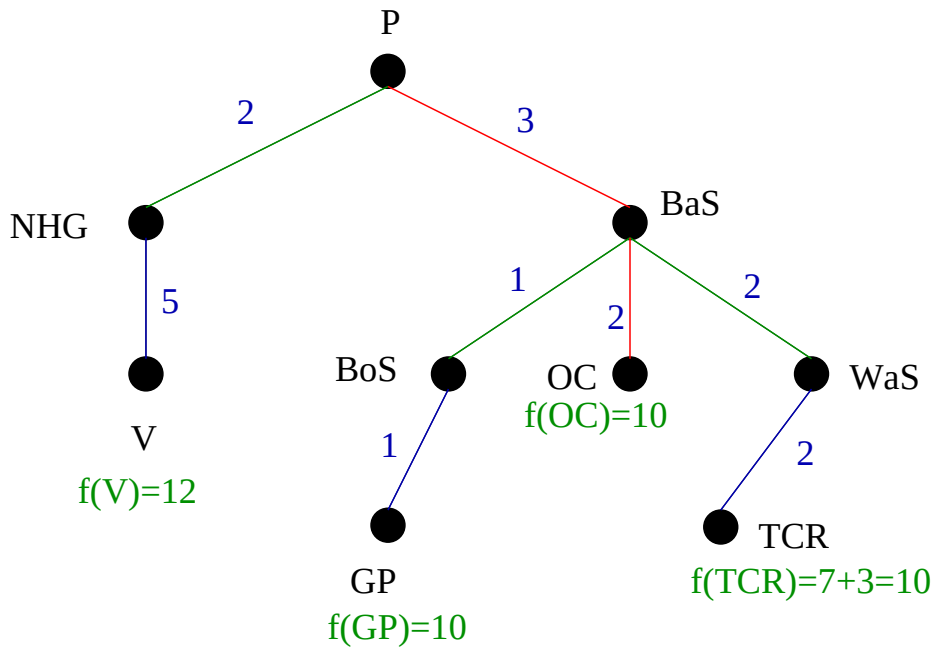


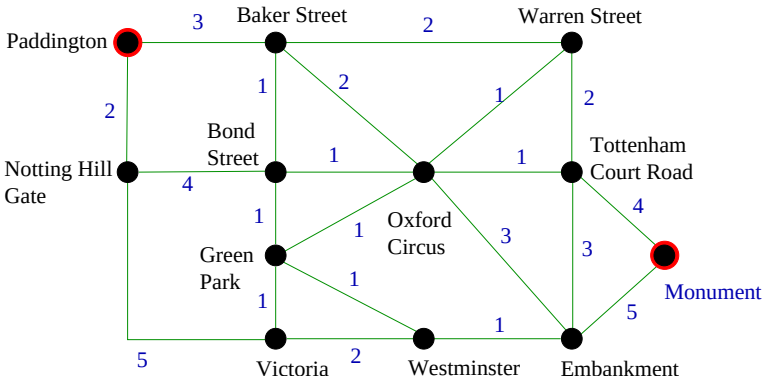


OPEN=[OC,GP,TCR,V]

CLOSED=[P,BaS,NHG,BoS,WaS]

Example 1 – A* solution

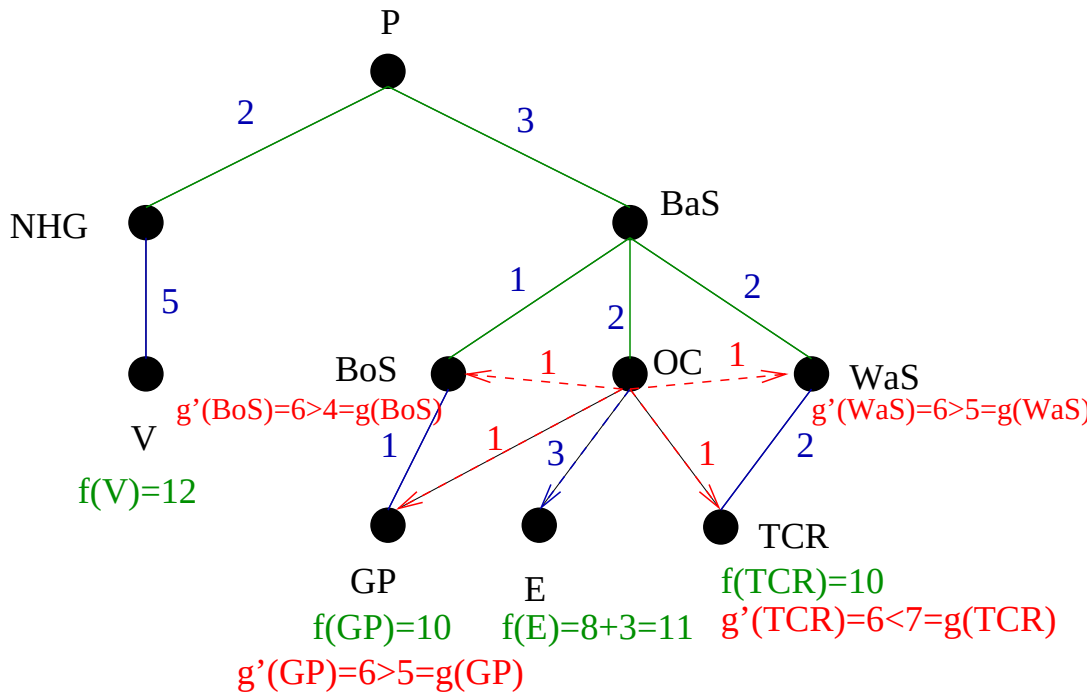


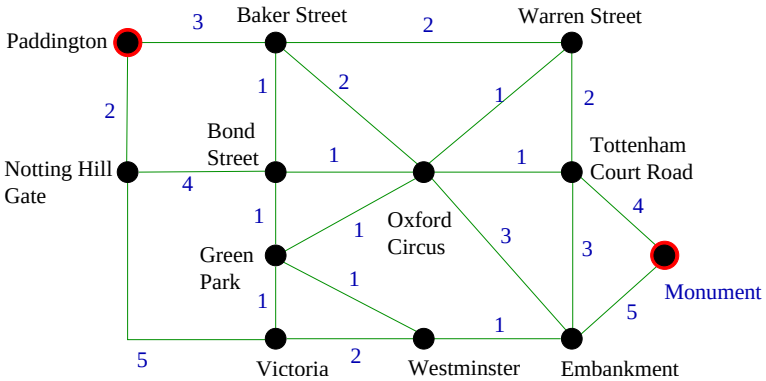


OPEN=[OC,GP,TCR,V]

CLOSED=[P,BaS,NHG,BoS,WaS]

Example 1 – A* solution

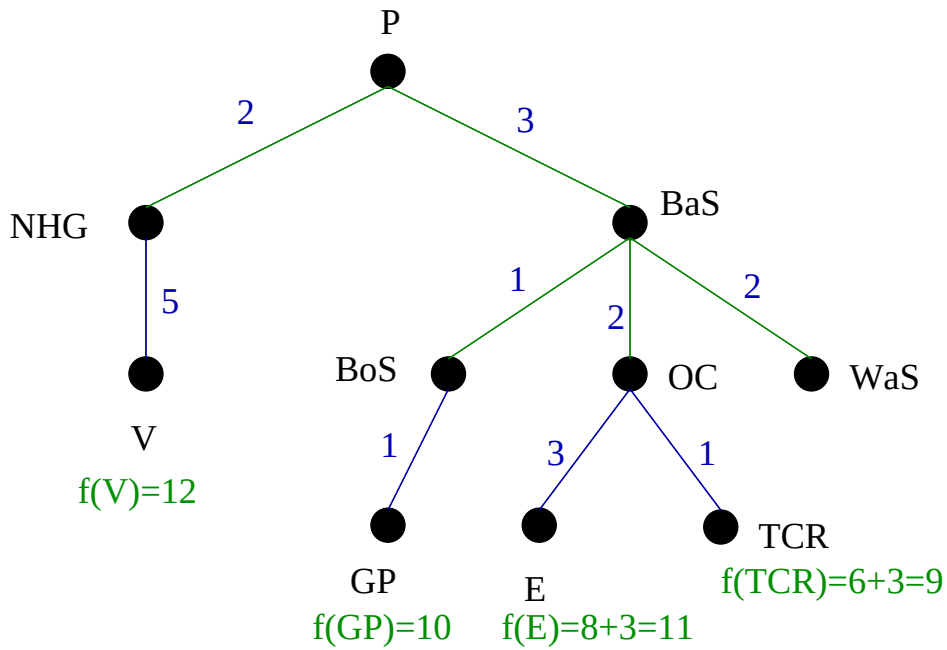


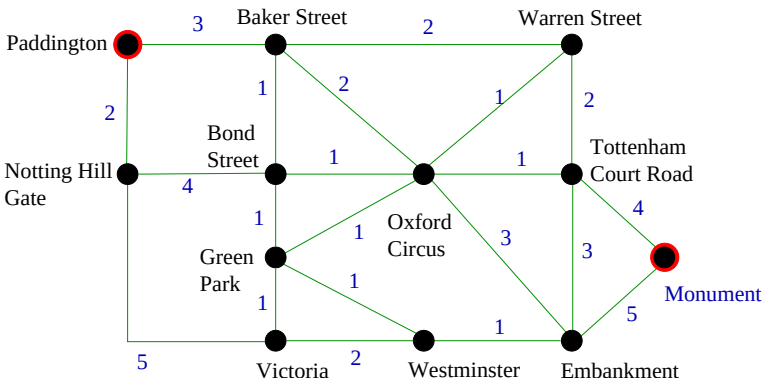


OPEN=[TCR,GP,V,E]

CLOSED=[P,BaS,NHG,BoS,WaS,OC]

Example 1 – A* solution

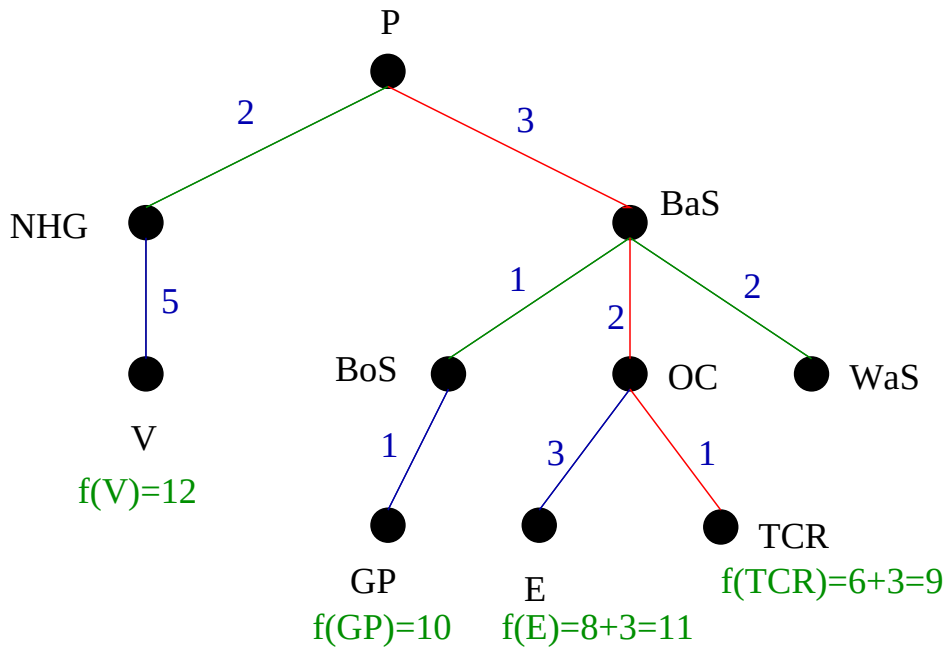


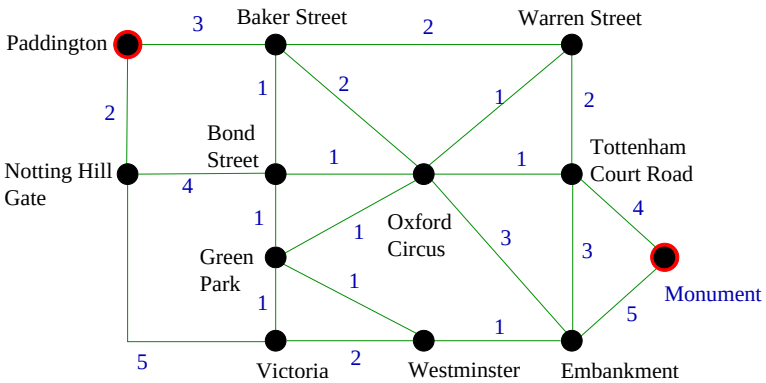


OPEN=[TCR,GP,V,E]

CLOSED=[P,BaS,NHG,BoS,WaS,OC]

Example 1 – A* solution

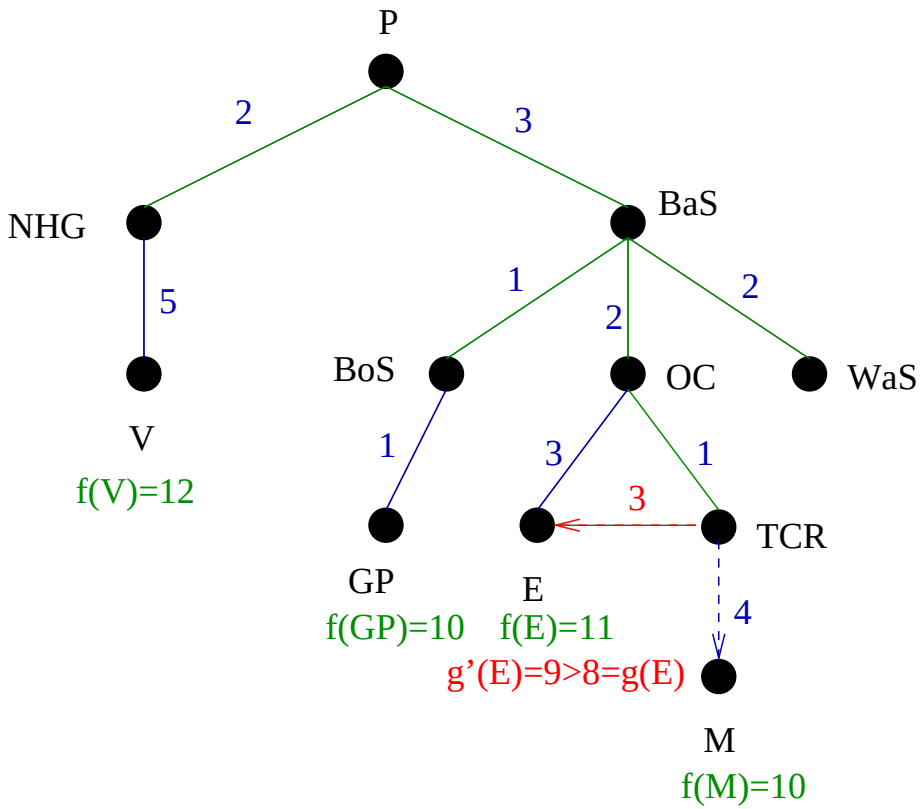


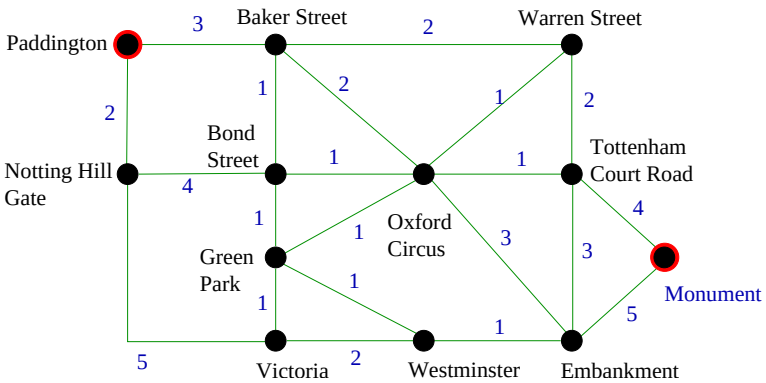


OPEN=[TCR,GP,V,E]

CLOSED=[P,BaS,NHG,BoS,WaS,OC]

Example 1 – A* solution

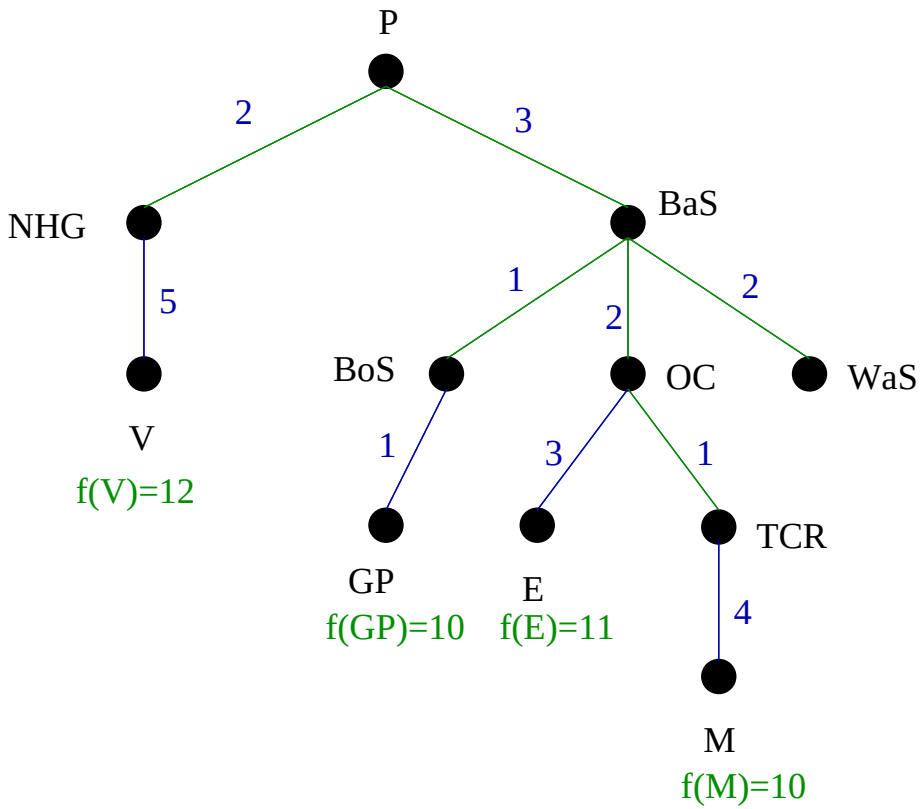


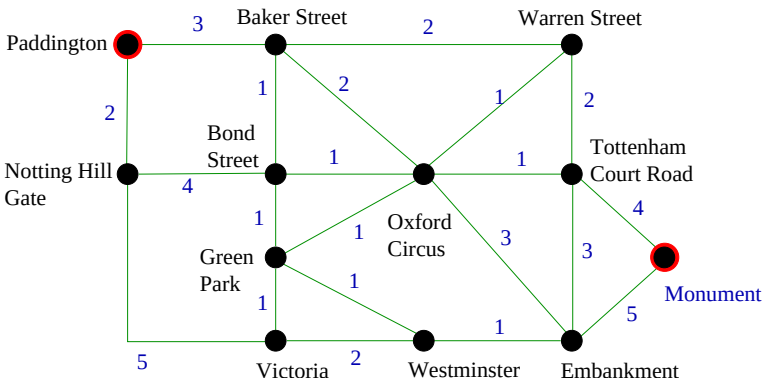


OPEN=[M,GP,V,E]

CLOSED=[P,BaS,NHG,BoS,WaS,OC,TCR]

Example 1 – A* solution

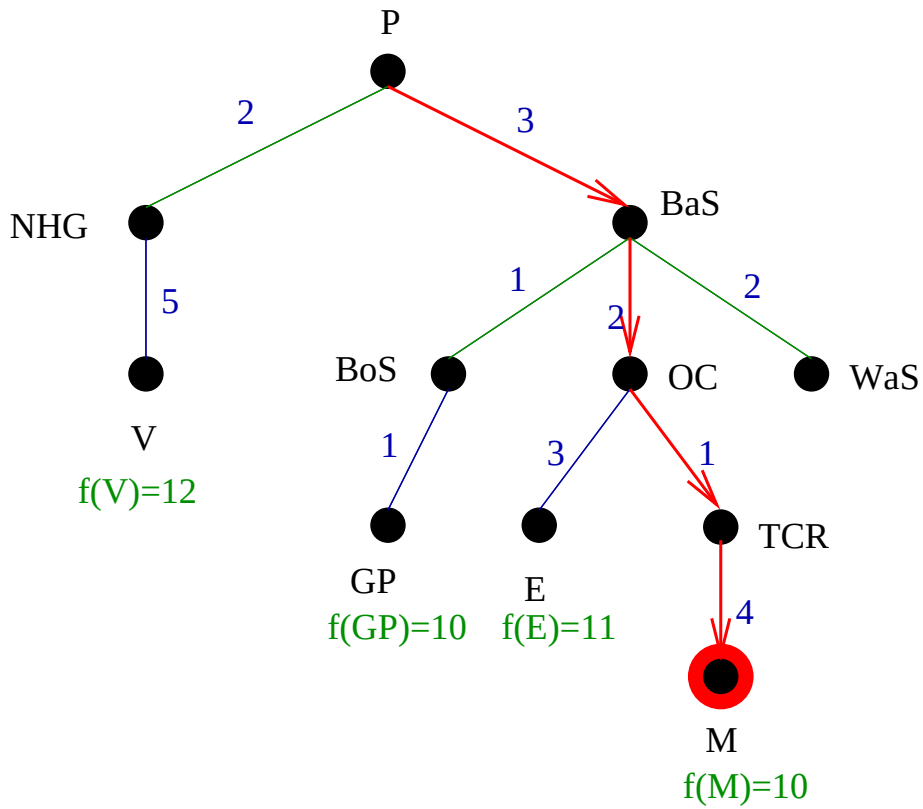


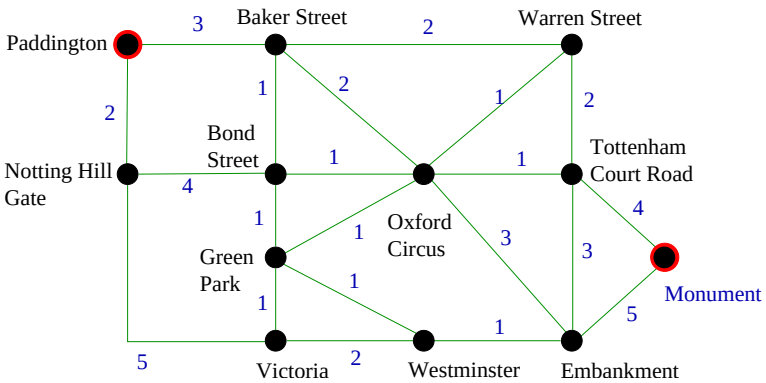


OPEN=[M,GP,V,E]

CLOSED=[P,BaS,NHG,BoS,WaS,OC,TCR]

Example 1 – A* solution





Example 2

The 8-puzzle:

	2	3
1	4	5
8	7	6

Start state

1	2	3
8		4
7	6	5

Goal state

Possible estimated costs:

- The number of tiles in wrong position:
  $h(start) = 6$

Example 2

The 8-puzzle:

	2	3
1	4	5
8	7	6

Start state

1	2	3
8		4
7	6	5

Goal state

Possible estimated costs:

- The number of tiles in wrong position:
  $h(start) = 6$
- The Manhattan distance:
  $h(start) = 1 + 0 + 0 + 1 + 1 + 1 + 1 + 1 = 6$

Example 2 – A* solution

Algorithm of A*

Let OPEN be the list of generated, but not yet examined nodes.

1. Start with OPEN containing the initial state.

Algorithm of A*

Let OPEN be the list of generated, but not yet examined nodes.

1. Start with OPEN containing the initial state.
2. Do until a goal state is found or no nodes are left in OPEN:

Algorithm of A*

Let OPEN be the list of generated, but not yet examined nodes.

1. Start with OPEN containing the initial state.
2. Do until a goal state is found or no nodes are left in OPEN:
   (a) Take the best node(lowest f value) in OPEN and generate its successors.

Algorithm of A*

Let OPEN be the list of generated, but not yet examined nodes.

1. Start with OPEN containing the initial state.
2. Do until a goal state is found or no nodes are left in OPEN:
   (a) Take the best node(lowest f value) in OPEN and generate its successors.
   (b) For each successor do:

Algorithm of A*

Let OPEN be the list of generated, but not yet examined nodes.

1. Start with OPEN containing the initial state.
2. Do until a goal state is found or no nodes are left in OPEN:
   (a) Take the best node(lowest *f* value) in OPEN and generate its successors.
   (b) For each successor do:
      i. if it has not been generated before, evaluate it and add it to OPEN.

Algorithm of A*

Let OPEN be the list of generated, but not yet examined nodes.

1. Start with OPEN containing the initial state.
2. Do until a goal state is found or no nodes are left in OPEN:
   (a) Take the best node(lowest *f* value) in OPEN and generate its successors.
   (b) For each successor do:
      i. if it has not been generated before, evaluate it and add it to OPEN.
      What does evaluation mean?

Algorithm of A*

Let OPEN be the list of generated, but not yet examined nodes.

1. Start with OPEN containing the initial state.
2. Do until a goal state is found or no nodes are left in OPEN:
   (a) Take the best node(lowest *f* value) in OPEN and generate its successors.
   (b) For each successor do:
      i. if it has not been generated before, evaluate it and add it to OPEN.
      ii. otherwise change the parent, if this path is better and modify accordingly the costs for the depending nodes.

Algorithm of A*

Let OPEN be the list of generated, but not yet examined nodes.

1. Start with OPEN containing the initial state.
2. Do until a goal state is found or no nodes are left in OPEN:
   (a) Take the best node(lowest *f* value) in OPEN and generate its successors.
   (b) For each successor do:
      i. if it has not been generated before, evaluate it and add it to OPEN.
      ii. otherwise change the parent, if this path is better and modify accordingly the costs for the depending nodes.
      When is a path better then the other?

How to perform step 2.(b)ii.

Let CLOSED be the list of generated and examined nodes.

How to perform step 2.(b)ii.

Let CLOSED be the list of generated and examined nodes.

For a node that has a copy in the OPEN list we only have to modify the parent and the g, f values.

How to perform step 2.(b)ii.

Let CLOSED be the list of generated and examined nodes.

For a node that has a copy in the OPEN list we only have to modify the parent and the g, f values.

A node that has a copy in the CLOSED list has already been examined.

How to perform step 2.(b)ii. – 2

So, we need to modify the g values (and the f values, too) of the nodes below this node.

How to perform step 2.(b)ii. – 2

So, we need to modify the g values (and the f values, too) of the nodes below this node.

We can do a depth-first traversal of the tree rooted at this node.

How to perform step 2.(b)ii. – 2

So, we need to modify the g values (and the f values, too) of the nodes below this node.

We can do a depth-first traversal of the tree rooted at this node.

Each branch ends if there are no more successors or the present g value of a node is better.

Observations

$f(n)$ is the shortest estimated path from the start state to the goal state that passes through state n.

If the cost of each step is 1, A* will find the path consisting of the fewest number of steps.

Particular cases of A*

$g = 0 \Rightarrow$ greedy best-first search

Particular cases of A*

$g = 0 \Rightarrow$ greedy best-first search

$h = 0$ and $g = 0 \Rightarrow$ random search

Properties of *h*

- If *h* is a perfect estimator of the distance from the current node to the goal node, A* will never leave the optimal path.
- The better *h* estimates the real distance, the closer A* is to the "direct" path.
- If *h* never overestimates the real distance, A* is guaranteed to find the optimal solution.
- If *h* may overestimate the real distance, the optimal path can be found only if all paths in the search graph longer than the optimal solution are expanded.

Search tree or search graph?

More general form $\Rightarrow$ search graph

More simple form $\Rightarrow$ search tree

Search trees generate nodes faster, BUT duplicate nodes.

Search tree or search graph?

More general form $\Rightarrow$ search graph

More simple form $\Rightarrow$ search tree

Search trees generate nodes faster, BUT duplicate nodes.

Is node duplication a problem?

Recommended reading

S. Russell and P. Norvig: Artificial Intelligence, A Modern Approach
Section 4.1

E. Rich and K. Knight: Artificial Intelligence
Section 3.3