Outlines

- Informed Search
- Heuristic
- Heuristic Search
- Heuristic Function
- Best First Search
- Greedy Best First Search
- A* Search

Informed Search

- Uninformed search strategies can find solutions to problems by systematically generating new states and testing them against the goal.
- Unfortunately, these strategies are incredibly inefficient in most cases.
- Uninformed search uses problem-specific knowledge—can find solutions more efficiently.

Heuristic

- Heuristic means:
  - *To discover*
  - *Helping to discover or learn*
  - *Serving to discover*
  - *An aid to discover*
- Heuristics are "rules of thumb", educated guesses, intuitive judgments or simply common sense.
- A heuristic contributes to the reduction of search in a problem-solving activity.

Heuristic

- "Educated guess" is a heuristic that allows a person to reach a conclusion without exhaustive research.
- With an educated guess a person considers what they have observed in the past, and applies that history to a situation where a more definite answer has not yet been decided.

Heuristic Search

- Uses domain-dependent (heuristic) information in order to search the space more efficiently.

Ways of using heuristic information:

- Deciding which node to expand next, instead of doing the expansion in a strictly breadth-first or depth-first order.
- In the course of expanding a node, deciding which successor or successors to generate, instead of blindly generating all possible successors at one time.
- Deciding that certain nodes should be discarded, or pruned, from the search space.

Heuristic Search

- A moment's reflection will show ourselves constantly using heuristics in the course of our everyday lives.

- If the sky is grey we conclude that it would be better to put on a coat before going out.
- We book our holidays in August because that is when the weather is best.

Heuristic Function

- A heuristic function, h(n), provides an estimate of the cost of the path from a given node to the closest goal state.
- Must be zero if node represents a goal state.

Best First Search

- Depth first search is good because it allows a solution to be found without all competing branches to be expanded.
- Breadth first search is good because it does not get trapped on dead-end paths.
- Best First Search combines the advantages of the two.
- At each step of the best first search process, we select the most promising of the nodes we have generated so far.
- This is done by applying an appropriate heuristic function to each of them.

Best First Search

- An algorithm in which a node is selected for expansion based on an evaluation function f(n).
  - *Traditionally the node with the lowest evaluation function is selected*
  - *Not an accurate name…expanding the best node first would be a straight march to the goal.*
  - *Choose the node that appears to be the best*

Best First Search

- There is a whole family of Best-First Search algorithms with different evaluation functions.
- Each has a heuristic function h(n).
- h(n) = estimated cost of the cheapest path from node n to a goal node.
- Example: In route planning the estimate of the cost of the cheapest path might be the straight line distance between two cities.

Best First Search

- **g(n)** = Cost from the initial state to the current state n.
- **h(n)** = Estimated cost of the cheapest path from node n to a goal node.
- **f(n)** = Evaluation function to select a node for expansion (usually the lowest cost node).

Greedy Best First Search

- Greedy Best-First search tries to expand the node that is closest to the goal assuming it will lead to a solution quickly

$$\mathbf{f(n) = h(n)}$$

- f(n) = Function that gives an evaluation of the state.
- h(n) = The cost of getting from the current state to a goal state.

Greedy Best First Search

- The Greedy Best-First-Search algorithm works as uniform cost search, except that it has some estimate (called a heuristic) of how far from the goal any node is.
- Instead of selecting the node closest to the starting point, it selects the node closest to the goal.
- Greedy Best-First-Search is not guaranteed to find a shortest path.
- However, it runs much quicker than uniform cost search algorithm because it uses the heuristic function to guide its way towards the goal very quickly.

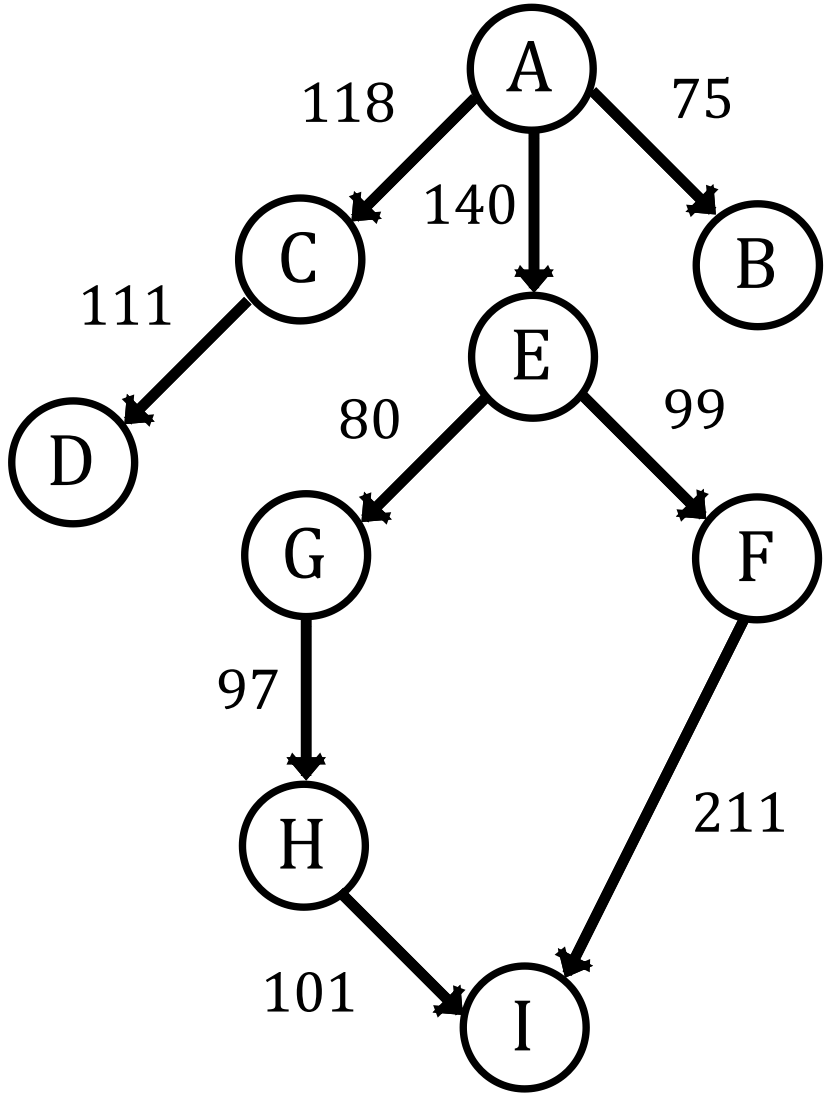
Greedy Best First Search

Algorithm:

- Step 01: Initialize Queue with Starting Node.
- Step 02: Pick the path with minimum heuristic cost f(n)=h(n) from Queue.
- Step 03: If the minimum path is goal, stop algorithm you found the solution.
- Step 04: For each neighbor node v add expanded path < v ,P > to Queue.
- Step 05: Repeat Step 02 to Step 04 until Queue is empty.

Greedy Best First Search (Problem 01)

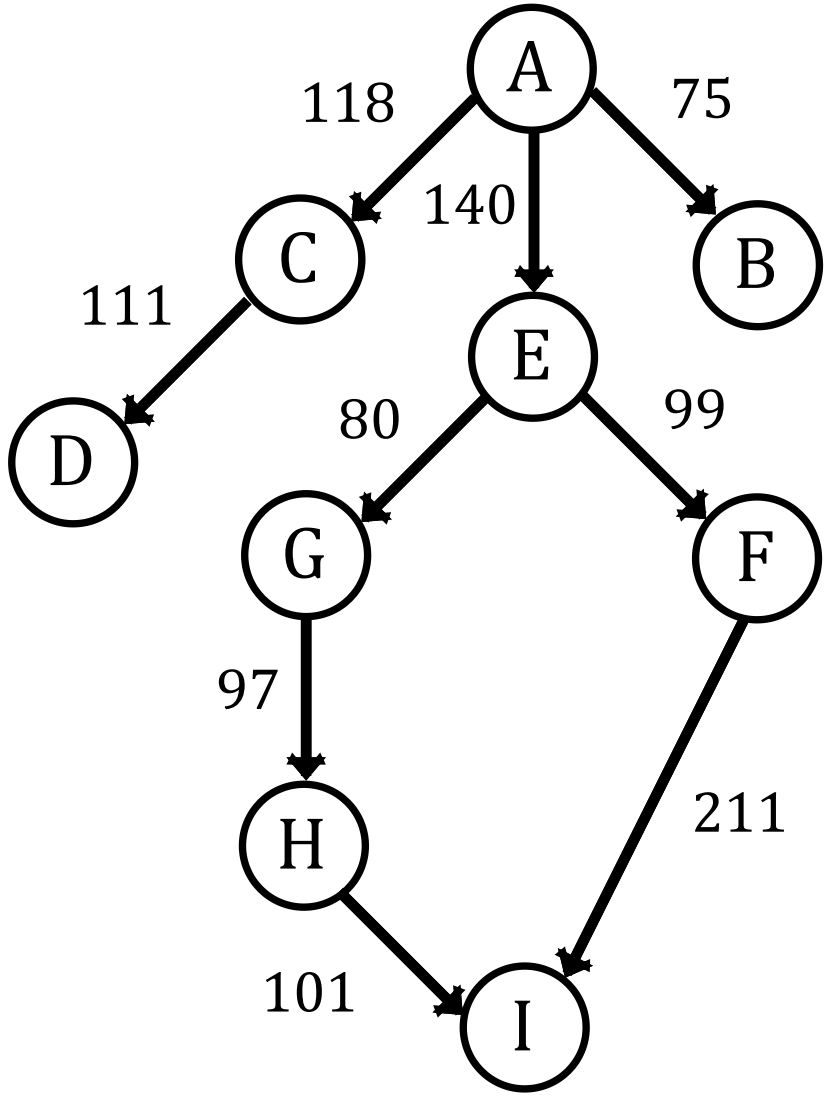
- **A = Initial State**
- **I = Goal State**



Greedy Best First Search (Problem 01)

f(n) = h(n) = Straight Line Distance

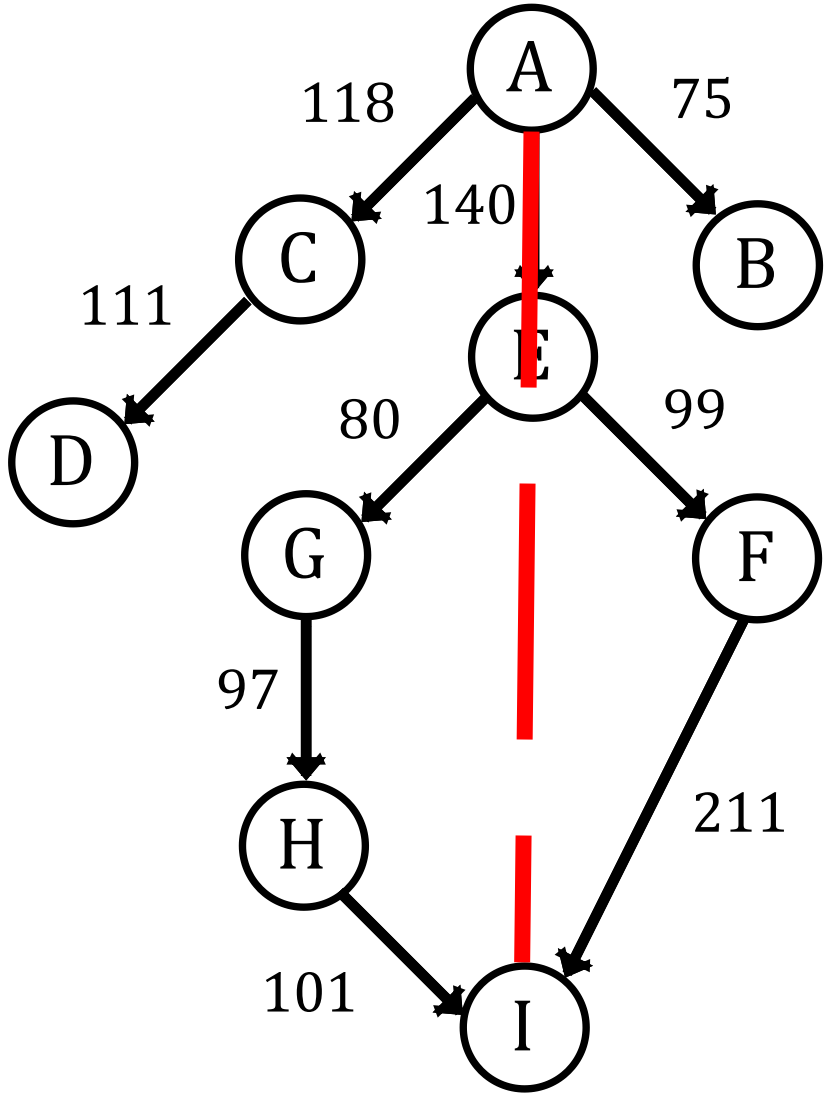
State	Heuristic: h(n)
A	366
B	374
C	329
D	244
E	253
F	178
G	193
H	98
I	0



Greedy Best First Search (Problem 01)

f(n) = h(n) = Straight Line Distance

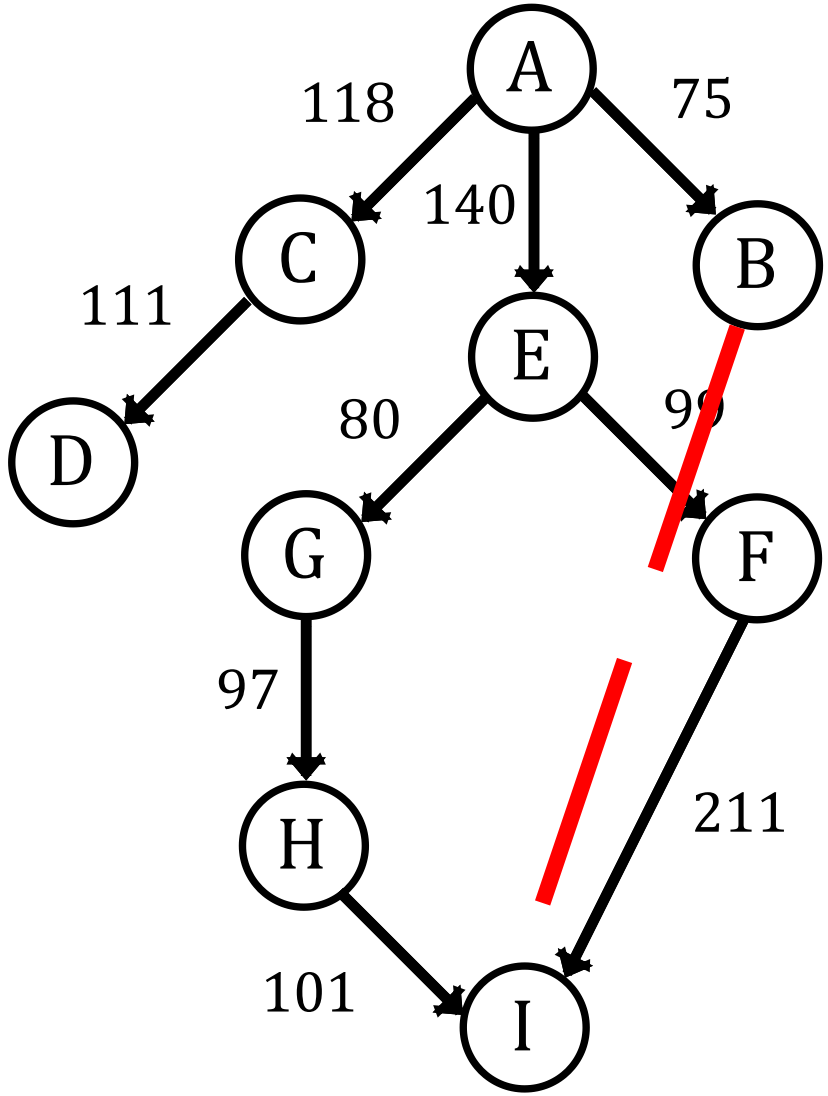
State	Heuristic: h(n)
A	**366**
B	374
C	329
D	244
E	253
F	178
G	193
H	98
I	0



Greedy Best First Search (Problem 01)

f(n) = h(n) = Straight Line Distance

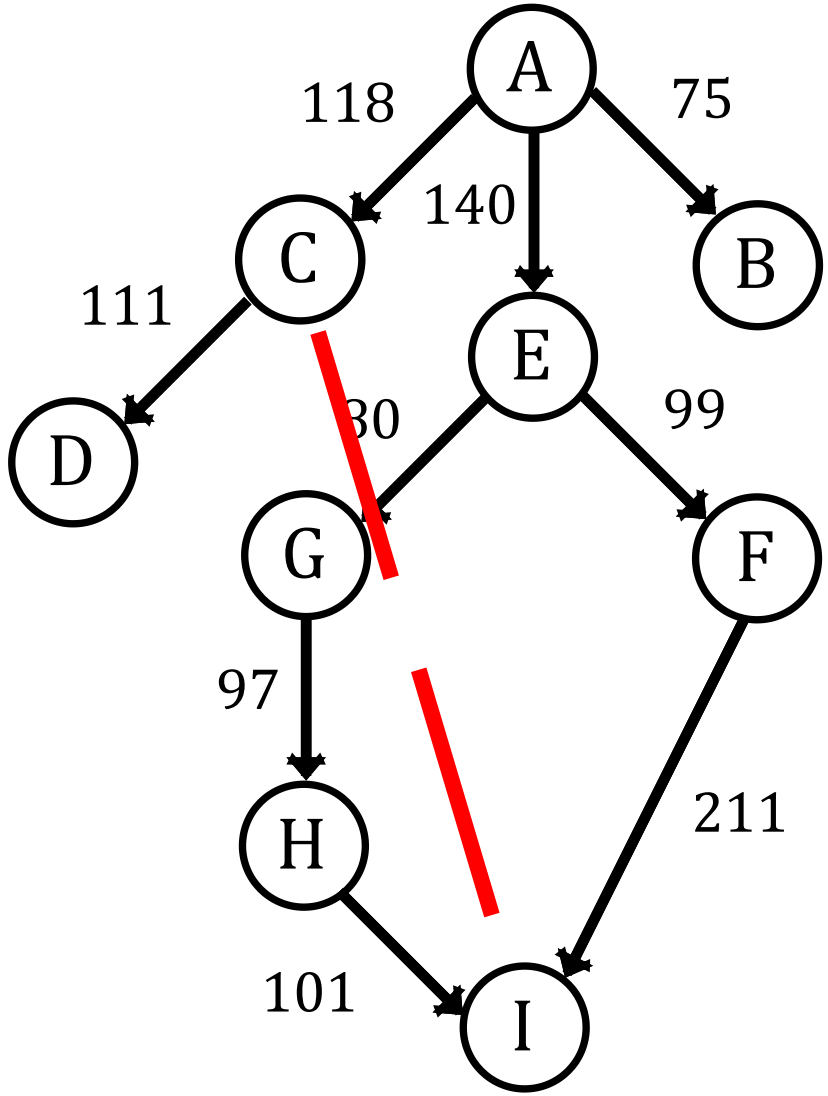
State	Heuristic: h(n)
A	366
B	**374**
C	329
D	244
E	253
F	178
G	193
H	98
I	0



Greedy Best First Search (Problem 01)

f(n) = h(n) = Straight Line Distance

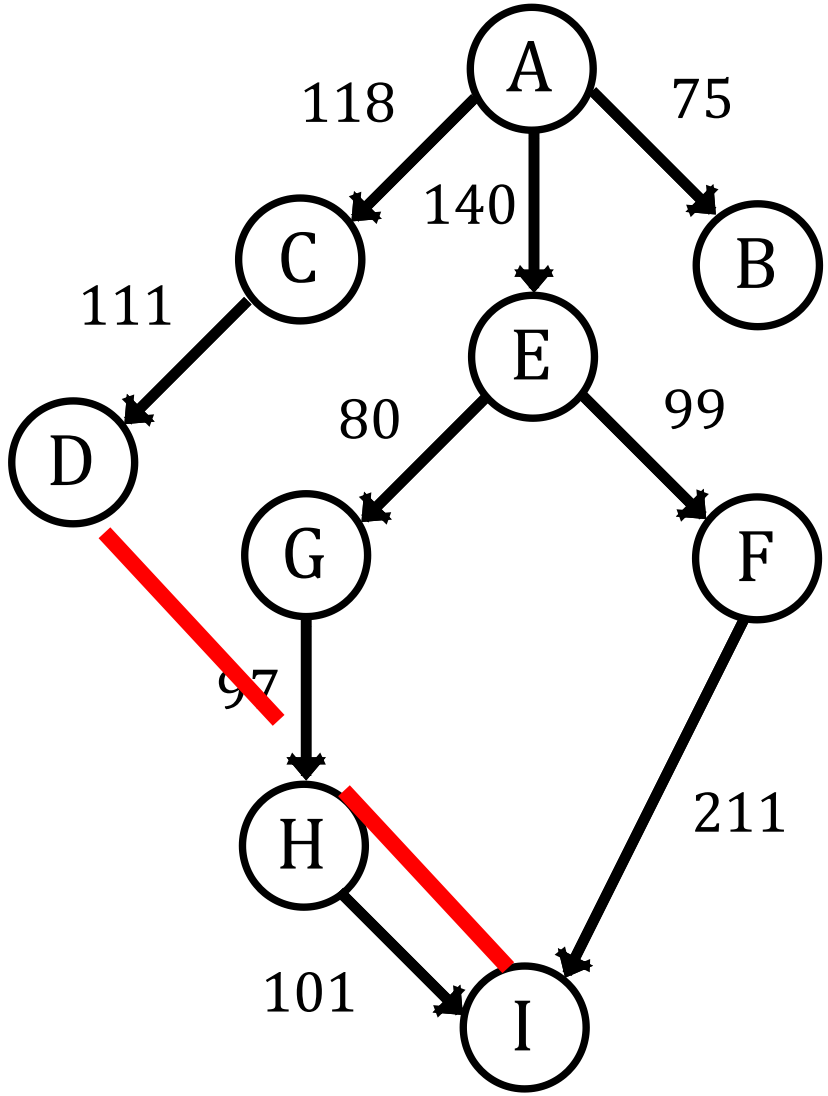
State	Heuristic: h(n)
A	366
B	374
C	**329**
D	244
E	253
F	178
G	193
H	98
I	0



Greedy Best First Search (Problem 01)

f(n) = h(n) = Straight Line Distance

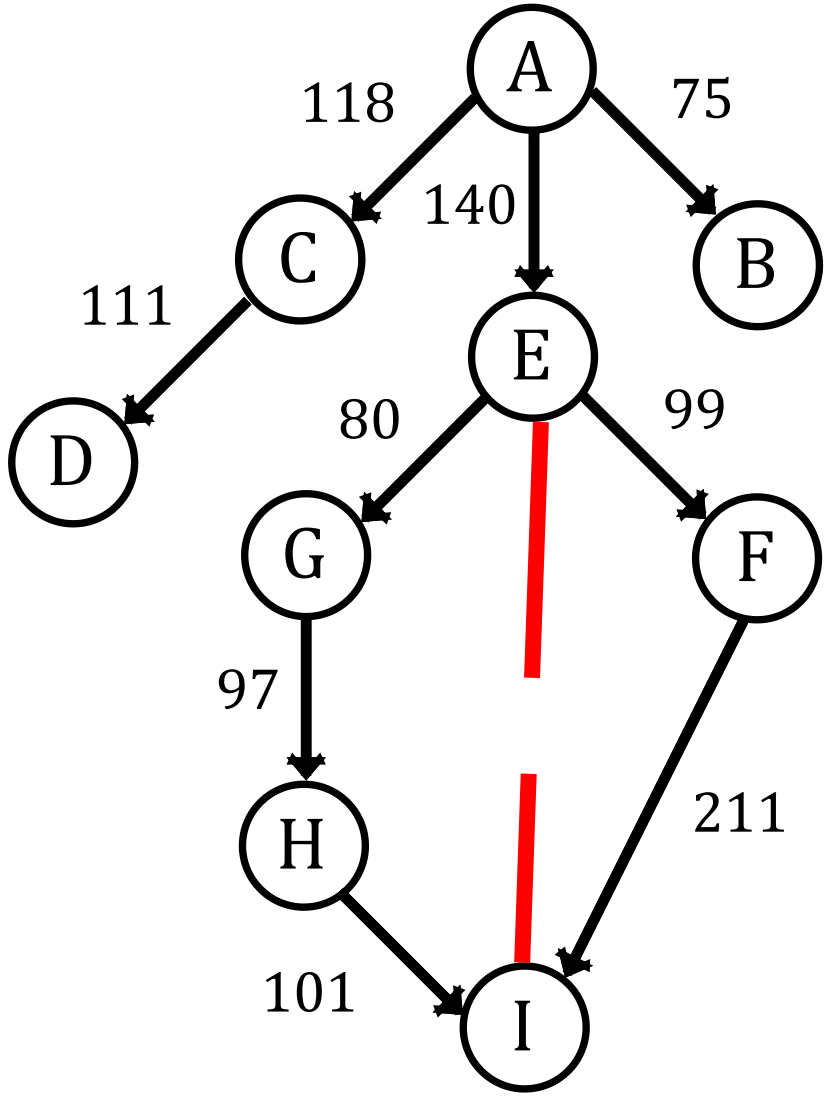
State	Heuristic: h(n)
A	366
B	374
C	329
D	**244**
E	253
F	178
G	193
H	98
I	0



Greedy Best First Search (Problem 01)

f(n) = h(n) = Straight Line Distance

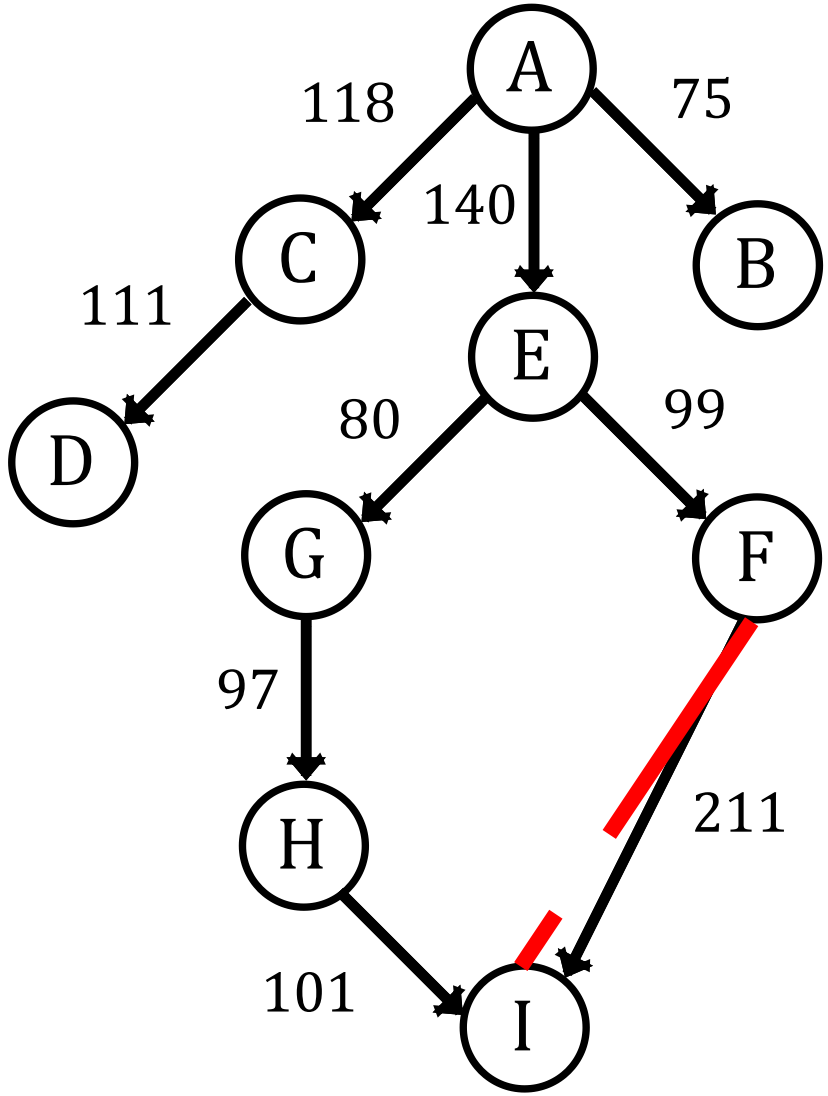
State	Heuristic: h(n)
A	366
B	374
C	329
D	244
E	**253**
F	178
G	193
H	98
I	0



Greedy Best First Search (Problem 01)

f(n) = h(n) = Straight Line Distance

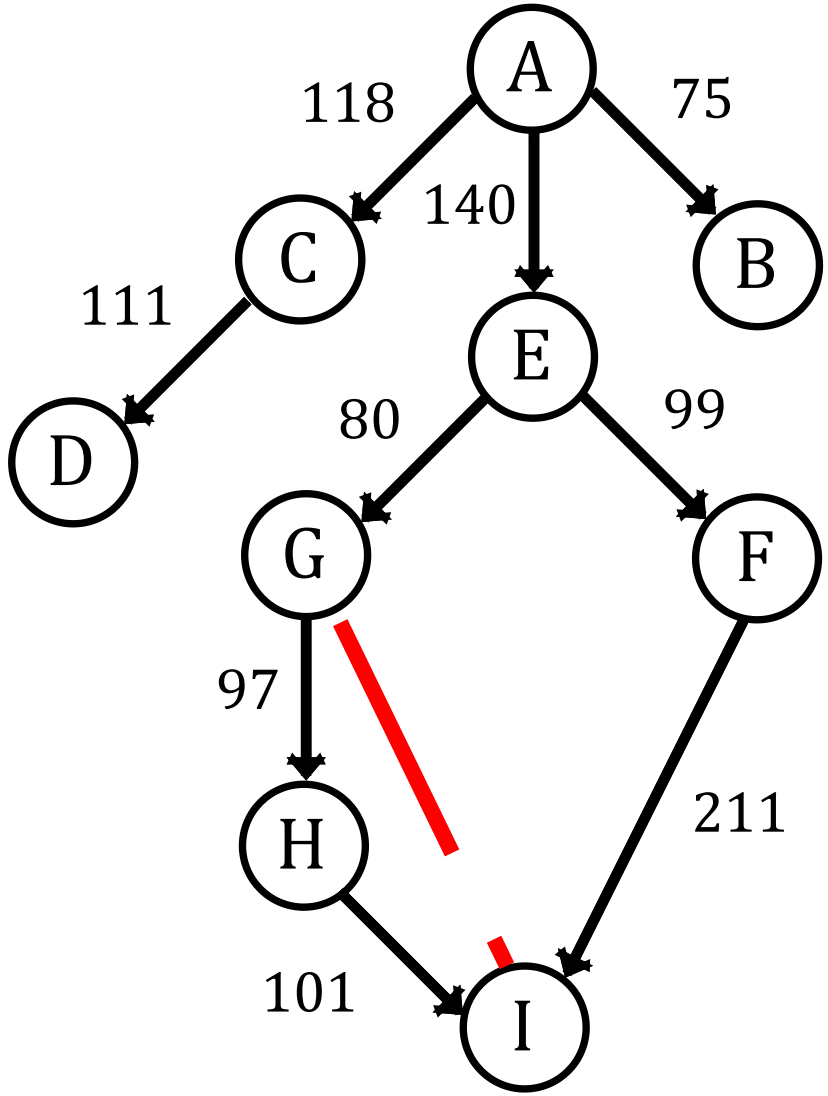
State	Heuristic: h(n)
A	366
B	374
C	329
D	244
E	253
F	**178**
G	193
H	98
I	0



Greedy Best First Search (Problem 01)

f(n) = h(n) = Straight Line Distance

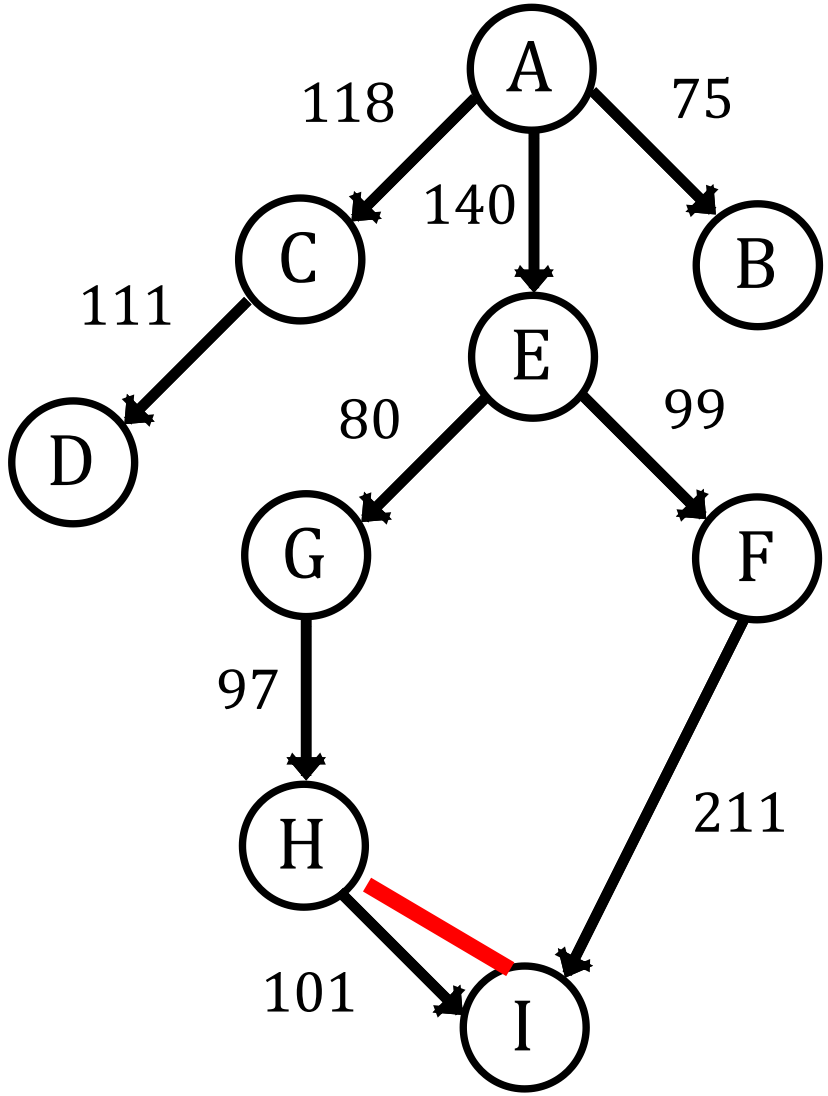
State	Heuristic: h(n)
A	366
B	374
C	329
D	244
E	253
F	178
G	**193**
H	98
I	0



Greedy Best First Search (Problem 01)

f(n) = h(n) = Straight Line Distance

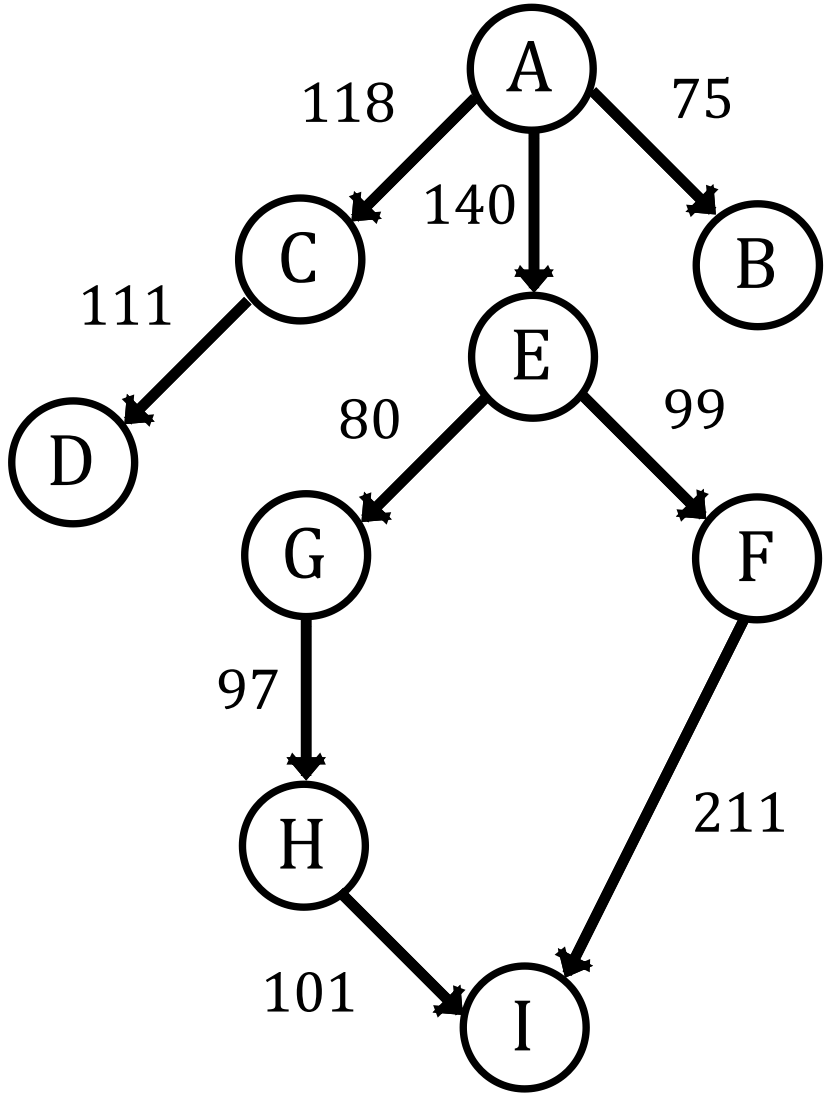
State	Heuristic: h(n)
A	366
B	374
C	329
D	244
E	253
F	178
G	193
H	**98**
I	0



Greedy Best First Search (Problem 01)

f(n) = h(n) = Straight Line Distance

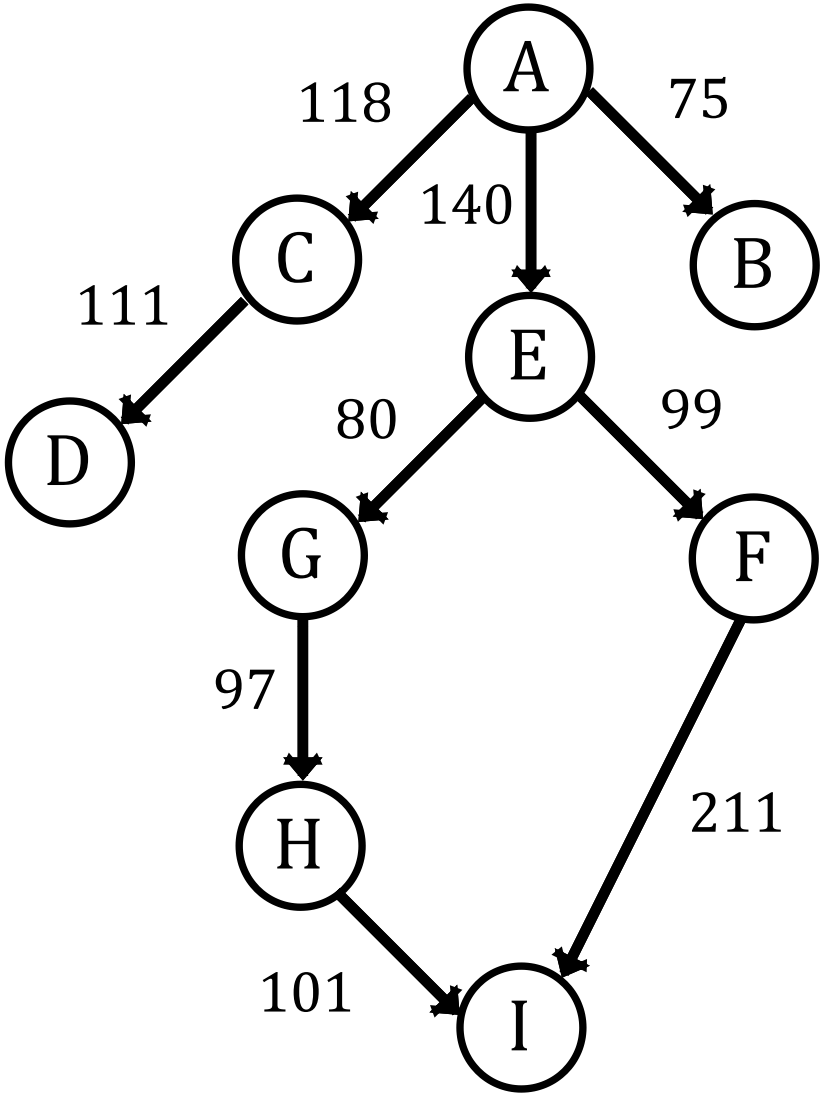
State	Heuristic: h(n)
A	366
B	374
C	329
D	244
E	253
F	178
G	193
H	98
I	**0**



Greedy Best First Search (Problem 01)

Queue		
Path	**Cost**	**f(n)**

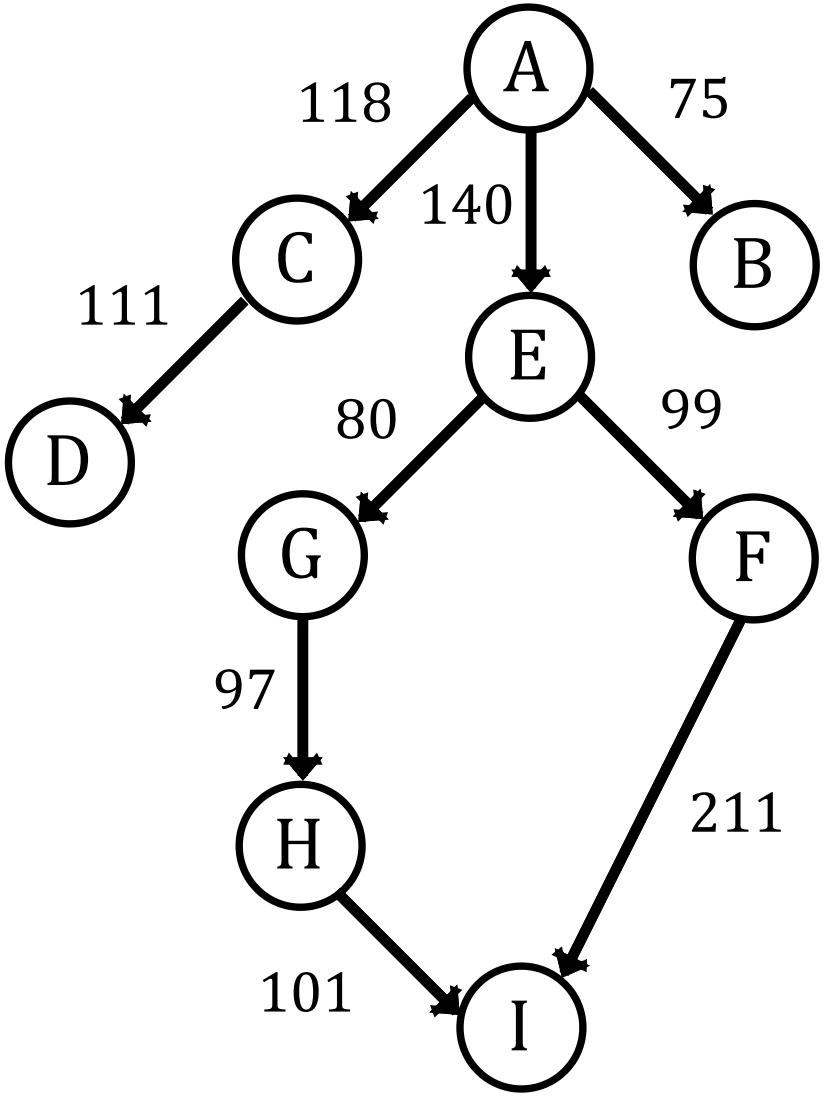
State	Heuristic: h(n)
A	366
B	374
C	329
D	244
E	253
F	178
G	193
H	98
I	0



Greedy Best First Search (Problem 01)

Queue		
Path	**Cost**	**f(n)**
<A>	0	366

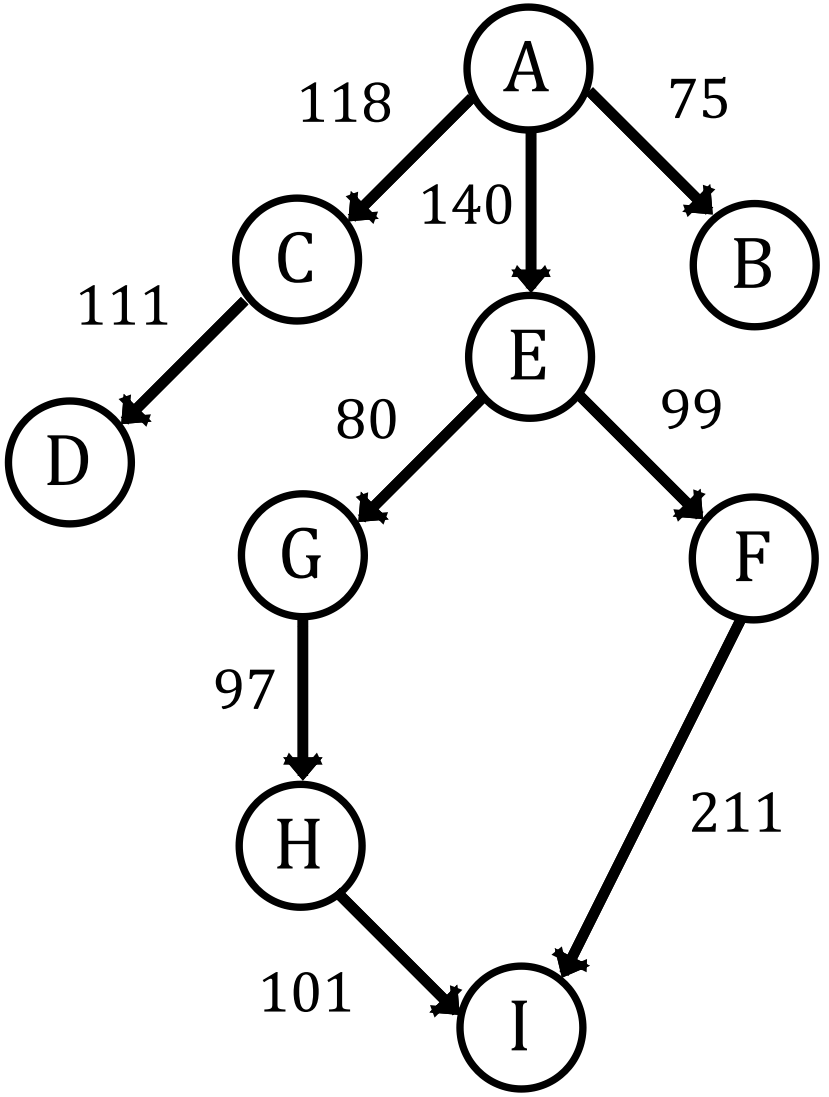
State	Heuristic: h(n)
A	366
B	374
C	329
D	244
E	253
F	178
G	193
H	98
I	0



Greedy Best First Search (Problem 01)

Queue		
Path	**Cost**	**f(n)**
<E,A>	140	253
<C,A>	118	329
<B,A>	75	374

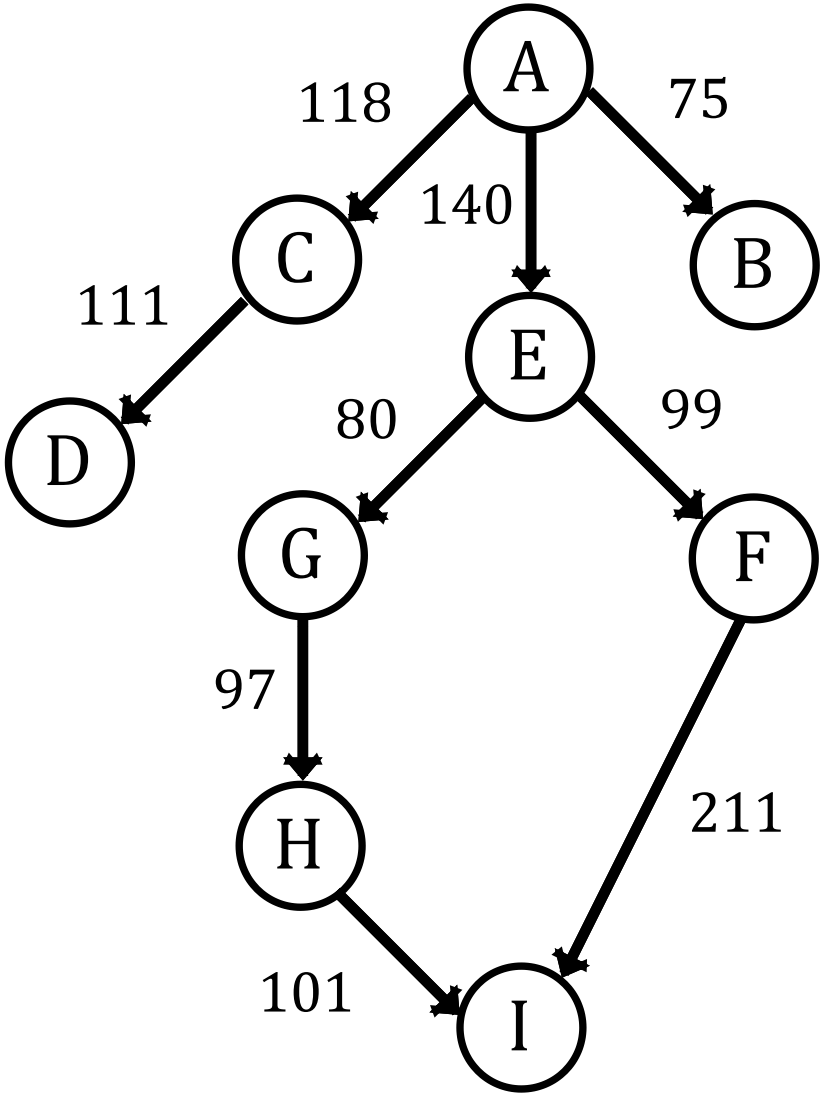
State	Heuristic: h(n)
A	366
B	374
C	329
D	244
E	253
F	178
G	193
H	98
I	0



Greedy Best First Search (Problem 01)

Queue		
Path	**Cost**	**f(n)**
<F,E,A>	239	178
<G,E,A>	220	193
<C,A>	118	329
<B,A>	75	374

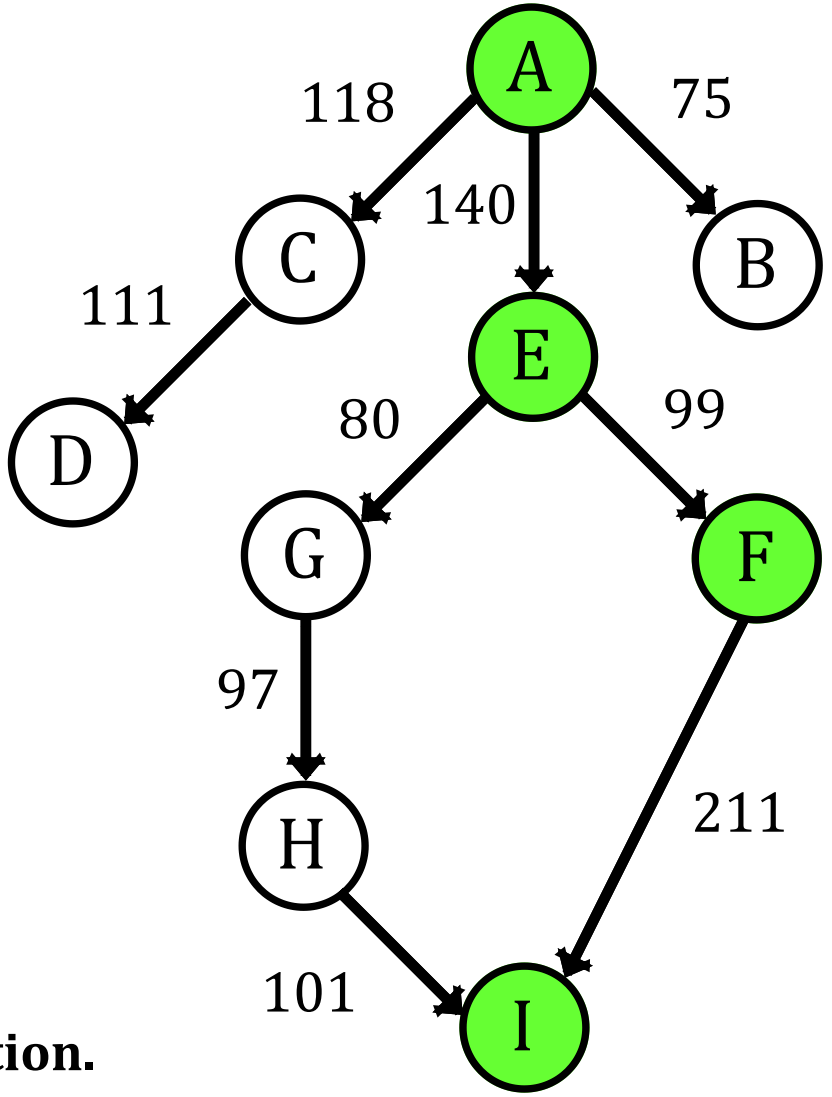
State	Heuristic: h(n)
A	366
B	374
C	329
D	244
E	253
F	178
G	193
H	98
I	0



Greedy Best First Search (Problem 01)

Queue		
Path	**Cost**	**f(n)**
<I,F,E,A>	450	0
<G,E,A>	220	193
<C,A>	118	329
<B,A>	75	374

State	Heuristic: h(n)
A	366
B	374
C	329
D	244
E	253
F	178
G	193
H	98
I	0



Solution = A → E → F → I
Path Cost = 140+99+211= 450

As minimum path is the goal path, so we have found a solution.

Greedy Best First Search

- Greedy First Search is not optimal.

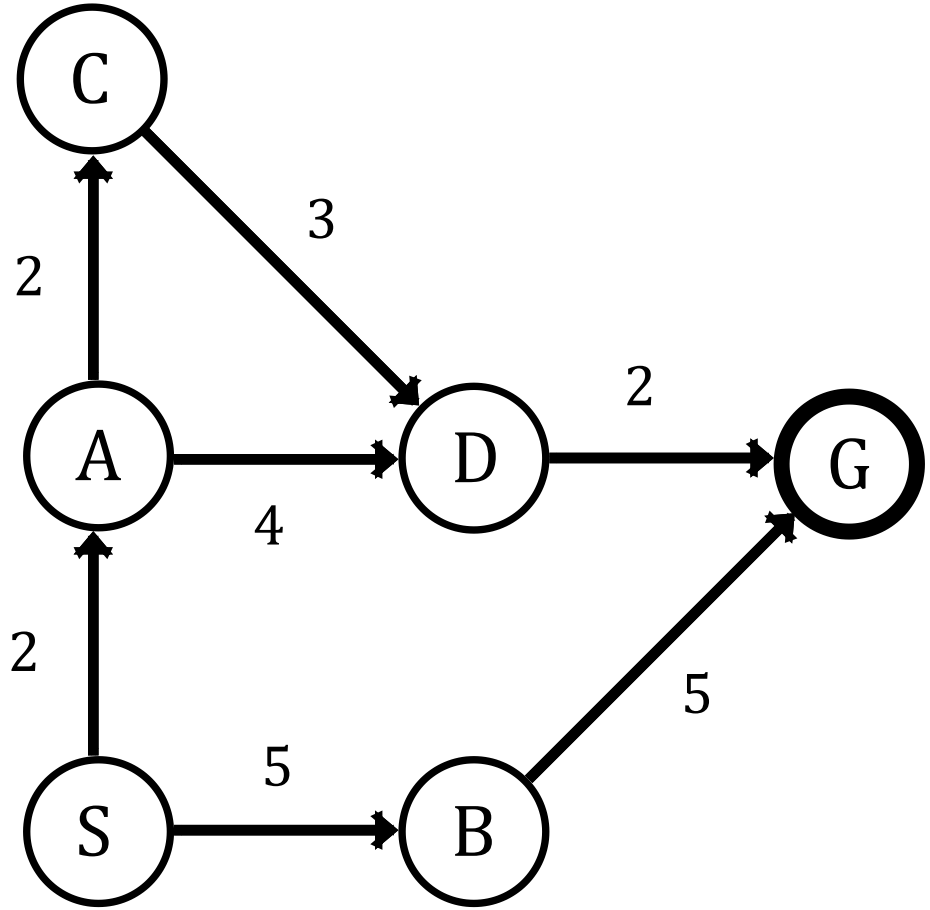
- **Path 1: A → E → F → I (Cost = 140+99+211= 450)**
- **Path 2: A → E → G → H → I (Cost = 140+80+97+101= 418)**

- There was path (i.e. path 2) more optimal than path 1.

- It is faster but not optimal.

Greedy Best First Search (Problem 02)

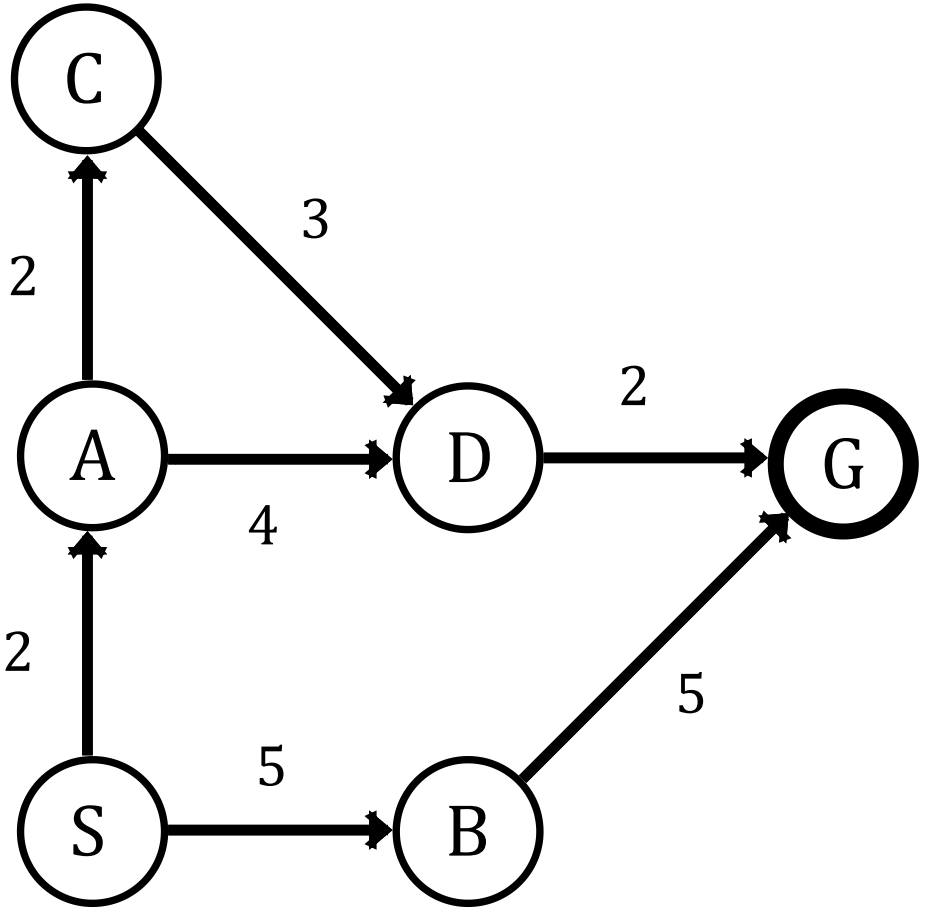
- **S = Initial State**
- **G = Goal State**



Greedy Best First Search (Problem 02)

f(n) = h(n) = Straight Line Distance

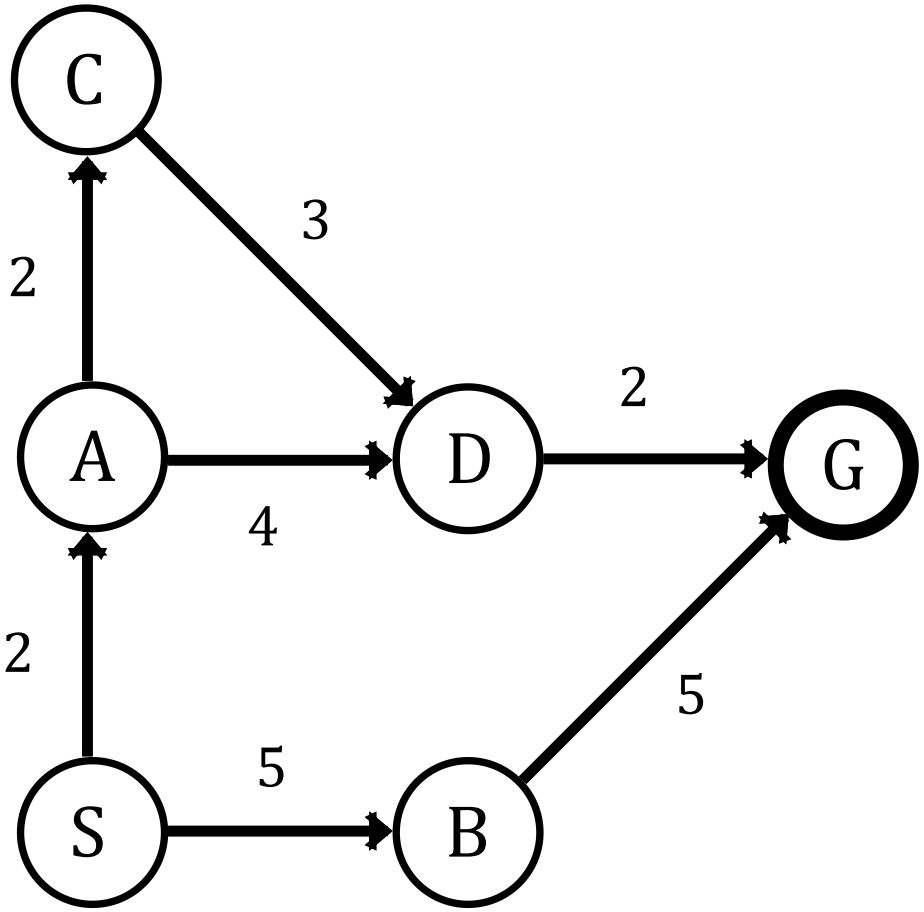
State	Heuristic: h(n)
S	10
A	2
B	3
C	1
D	4
G	0



Greedy Best First Search (Problem 02)

Queue		
Path	**Cost**	**f(n)**

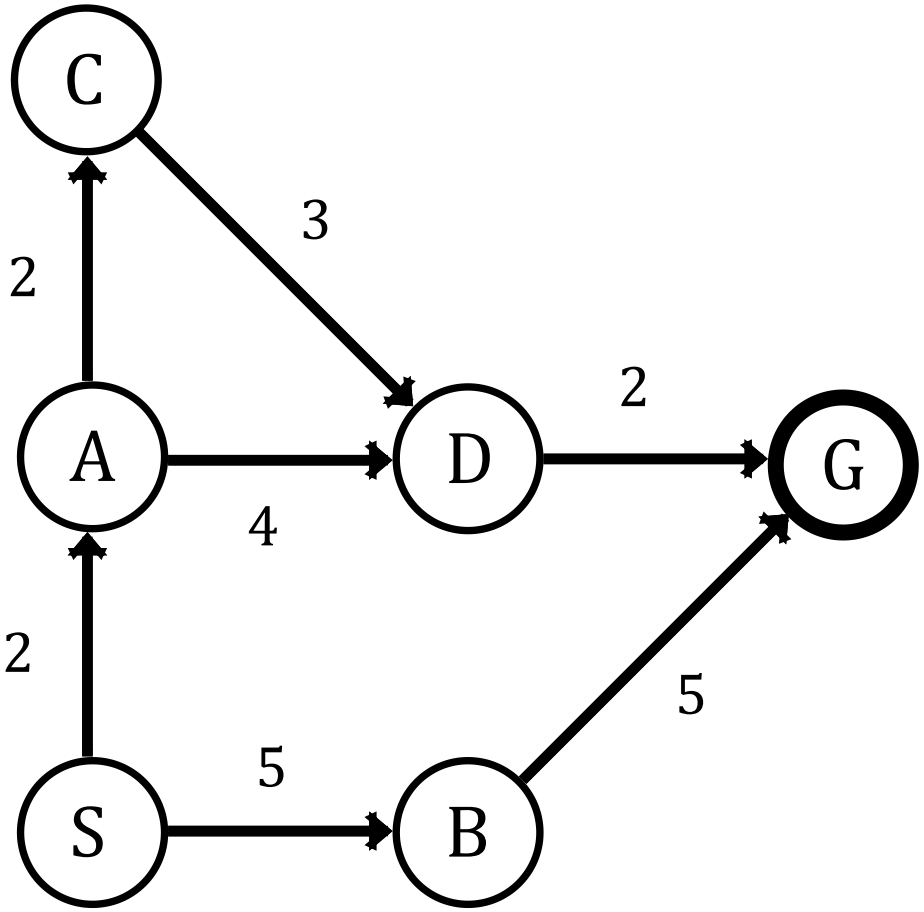
State	Heuristic: h(n)
S	10
A	2
B	3
C	1
D	4
G	0



Greedy Best First Search (Problem 02)

Queue		
Path	**Cost**	**f(n)**
<S>	0	10

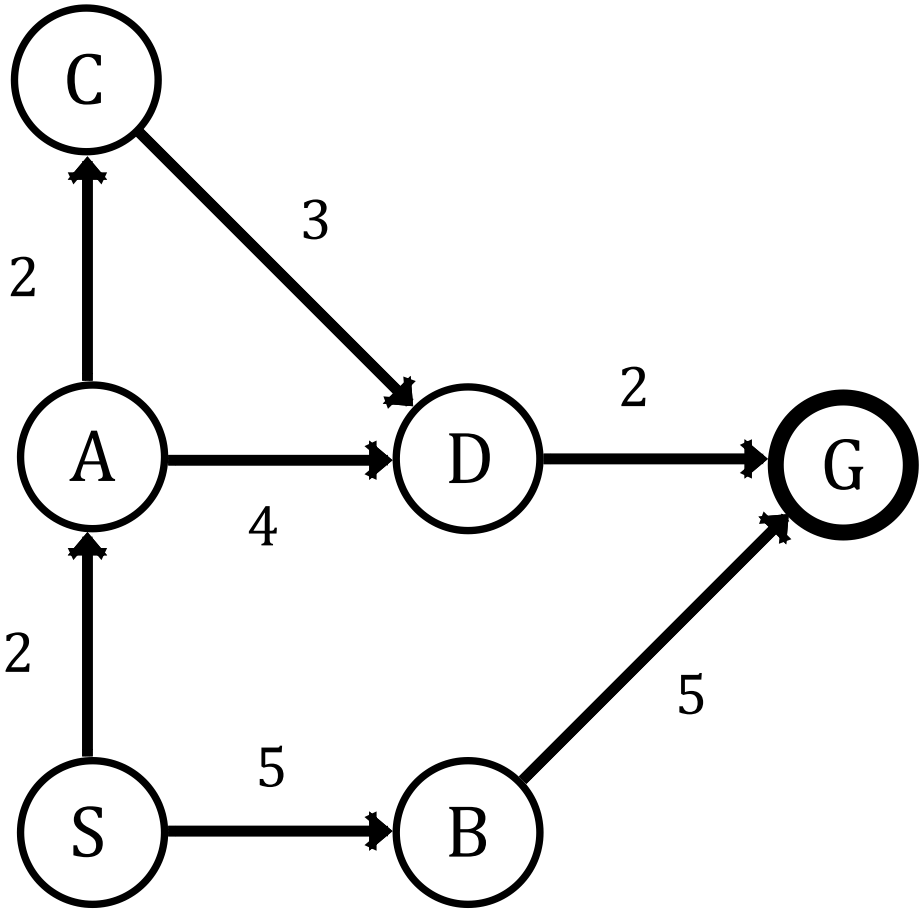
State	Heuristic: h(n)
S	10
A	2
B	3
C	1
D	4
G	0



Greedy Best First Search (Problem 02)

Queue		
Path	**Cost**	**f(n)**
<A,S>	2	2
<B,S>	5	3

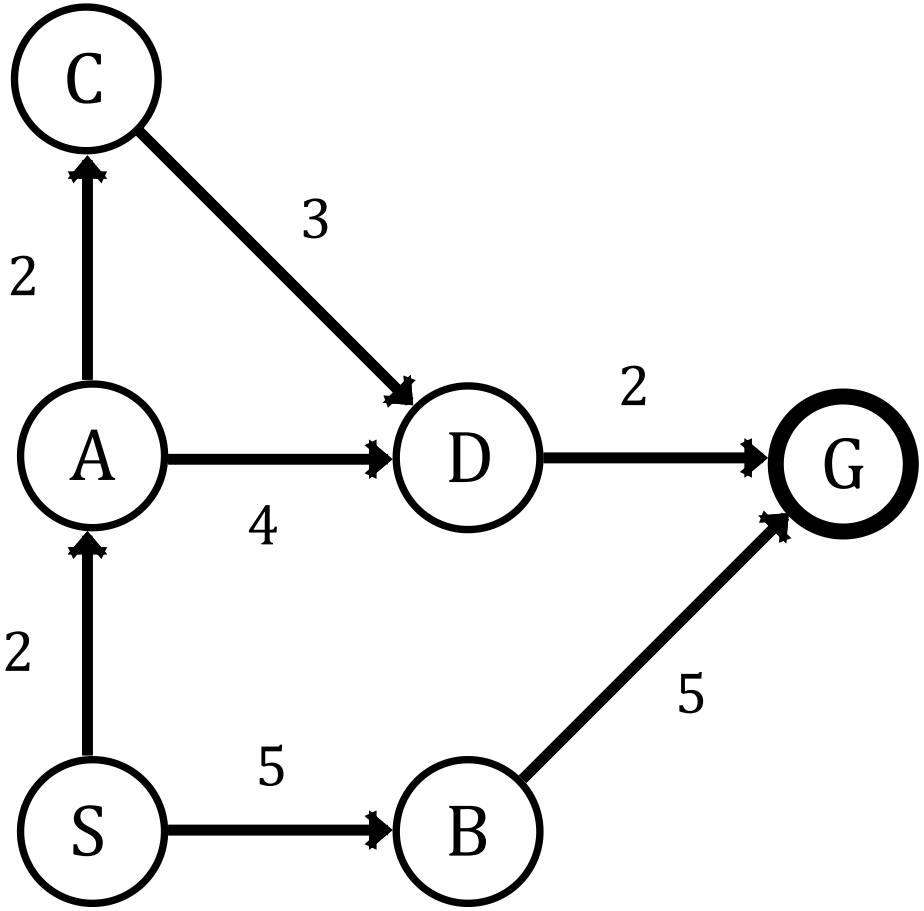
State	Heuristic: h(n)
S	10
A	2
B	3
C	1
D	4
G	0



Greedy Best First Search (Problem 02)

Queue		
Path	**Cost**	**f(n)**
<C,A,S>	4	1
<B,S>	5	3
<D,A,S>	6	4

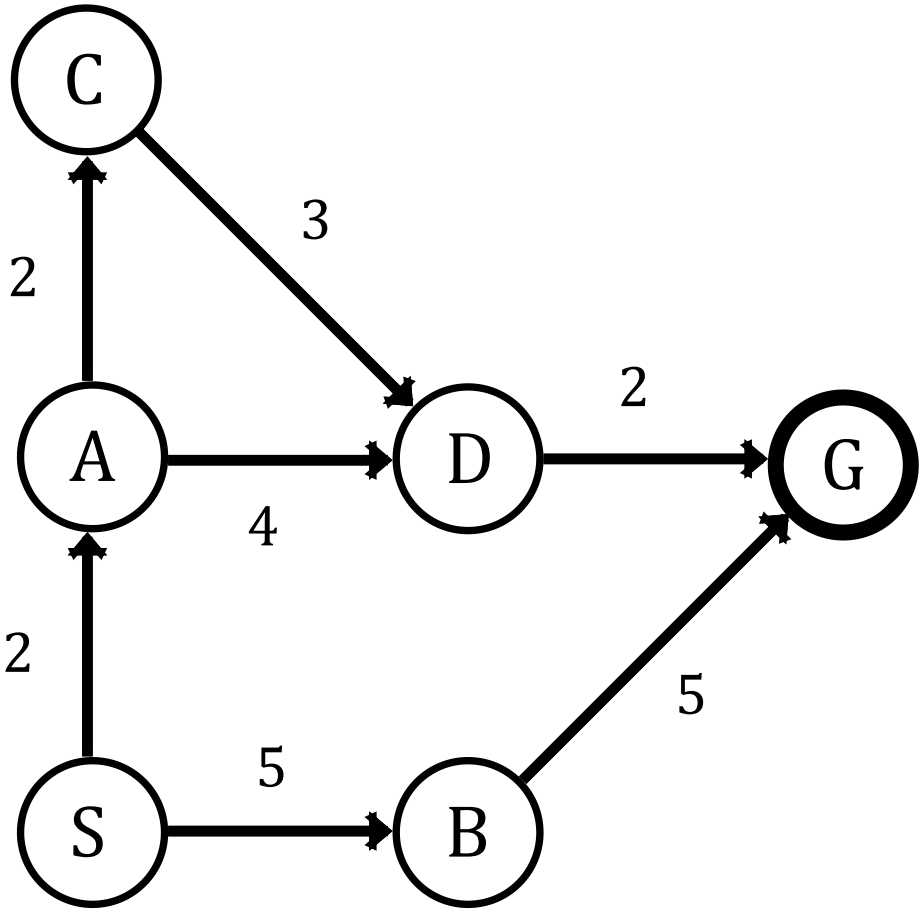
State	Heuristic: h(n)
S	10
A	2
B	3
C	1
D	4
G	0



Greedy Best First Search (Problem 02)

Queue		
Path	**Cost**	**f(n)**
<B,S>	5	3
<D,A,S>	6	4
<D,C,A,S>	7	4

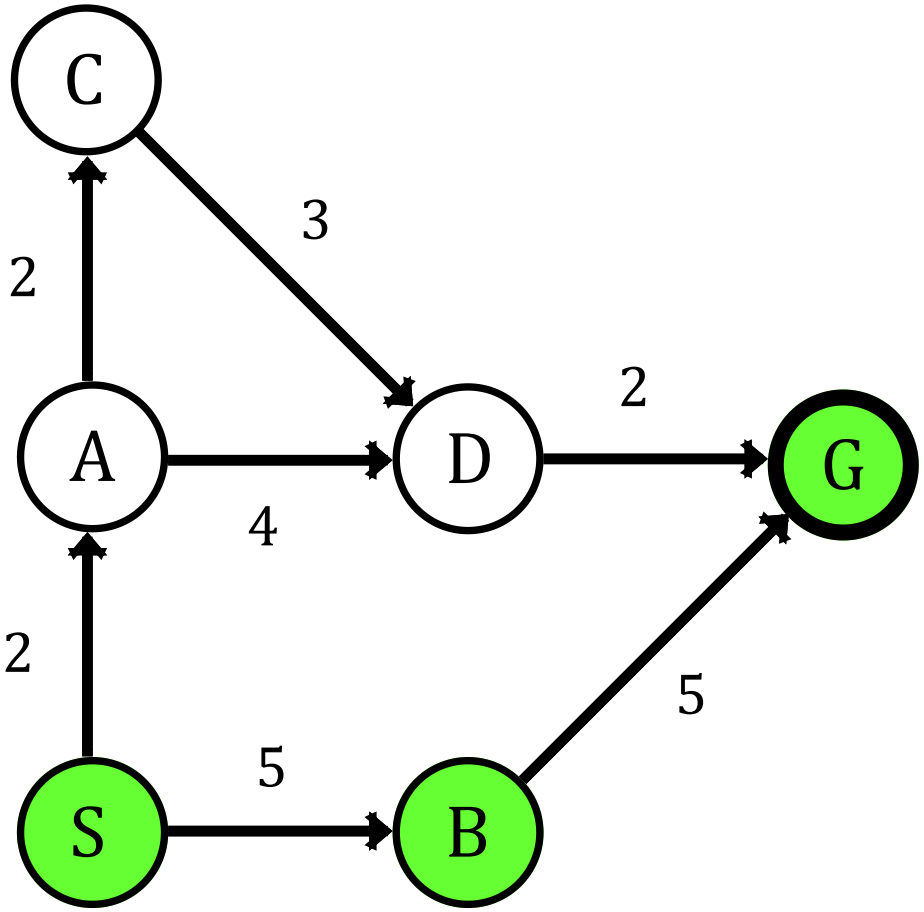
State	Heuristic: h(n)
S	10
A	2
B	3
C	1
D	4
G	0



Greedy Best First Search (Problem 02)

Queue		
Path	**Cost**	**f(n)**
<G,B,S>	10	0
<D,A,S>	6	4
<D,C,A,S>	7	4

State	Heuristic: h(n)
S	10
A	2
B	3
C	1
D	4
G	0



Solution = S → B → G
Path Cost = 5+5 = 10

As minimum path is the goal path, so we have found a solution.

A* Search

- Greedy Search minimizes a heuristic h(n) which is an estimated cost from a node n to the goal state. However, although greedy search can considerably cut the search time (efficient), it is neither optimal nor complete.
- Uniform Cost Search minimizes the cost g(n) from the initial state to n. UCS is optimal and complete but not efficient.
- New Strategy: Combine Greedy Search and UCS to get an efficient algorithm which is complete and optimal.

A* Search

- A search algorithm to find the shortest path through a search space to a goal state using a heuristic.

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

- f(n) = Function that gives an evaluation of the state.
- g(n) = The cost of getting from the initial state to the current state.
- h(n) = The cost of getting from the current state to a goal state.

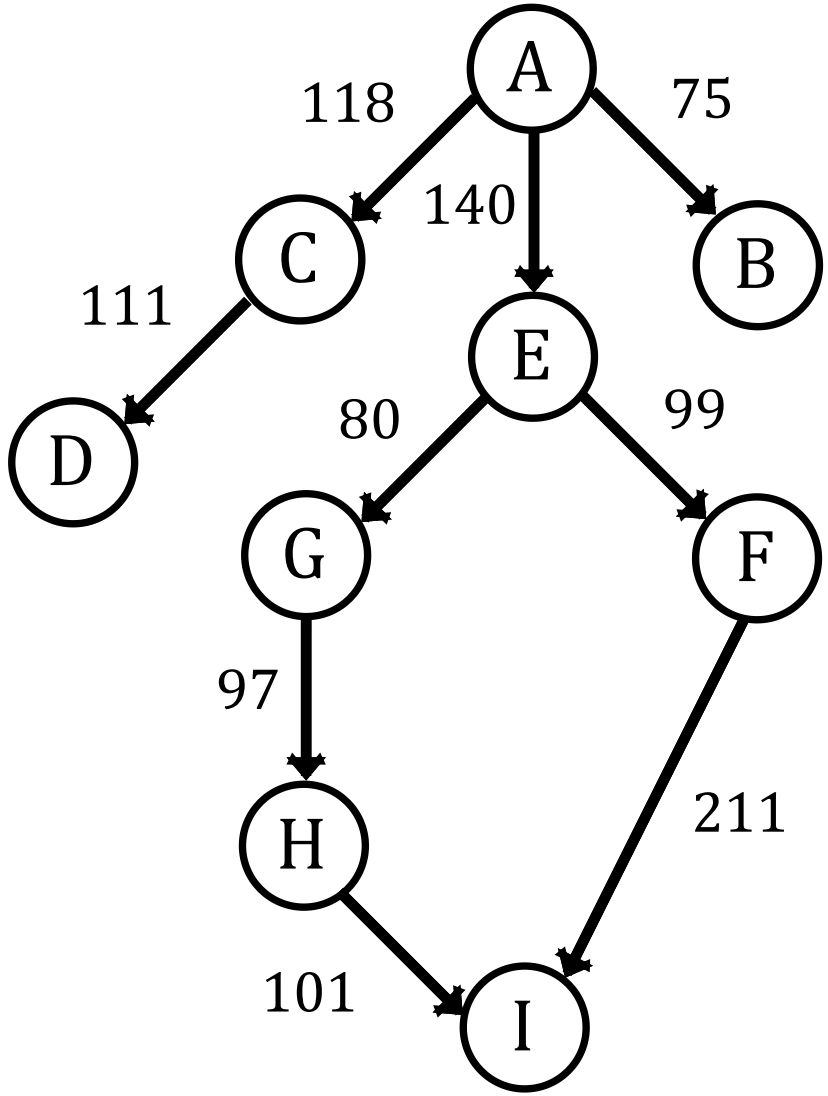
A* Search

Algorithm:

- Step 01: Initialize Queue with Starting Node.
- Step 02: Pick the path with minimum heuristic cost f(n)=g(n)+h(n) from Queue.
- Step 03: If the minimum path is goal, stop algorithm you found the solution.
- Step 04: For each neighbor node v add expanded path < v ,P > to Queue.
- Step 05: Repeat Step 02 to Step 04 until Queue is empty.

A* Search (Problem 01)

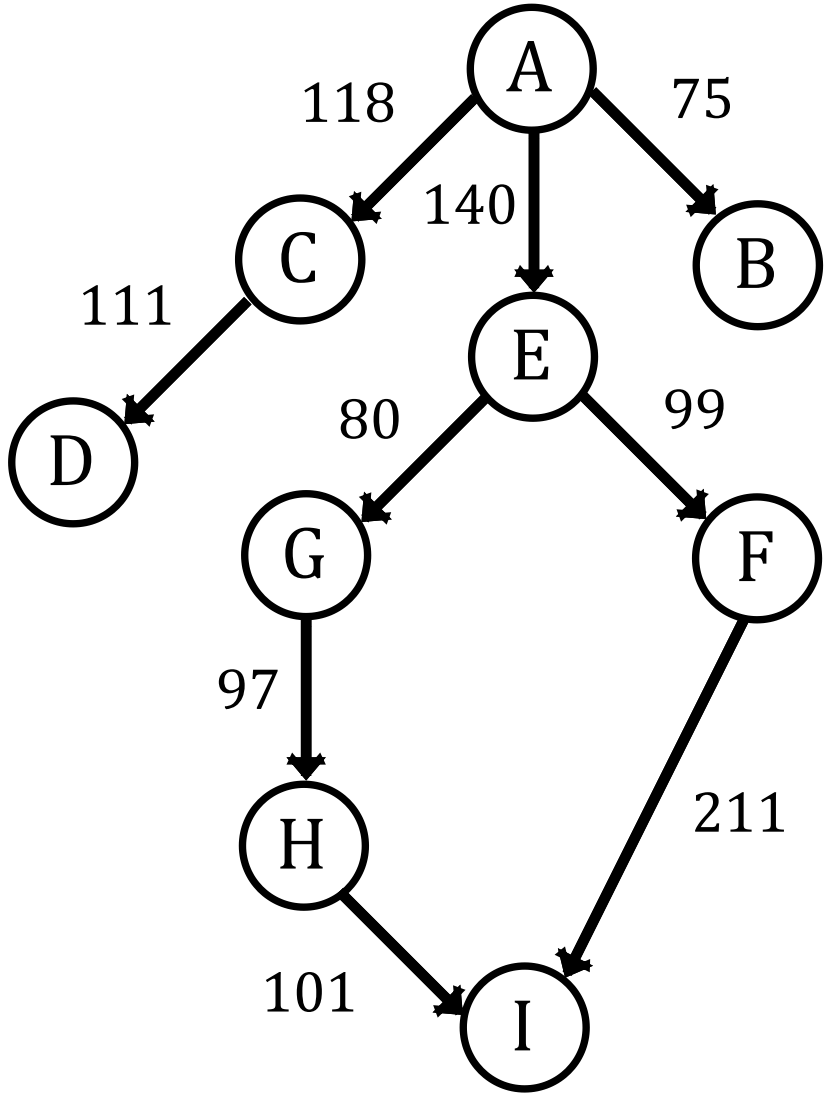
- **A = Initial State**
- **I = Goal State**



A* Search (Problem 01)

h(n) = Straight Line Distance

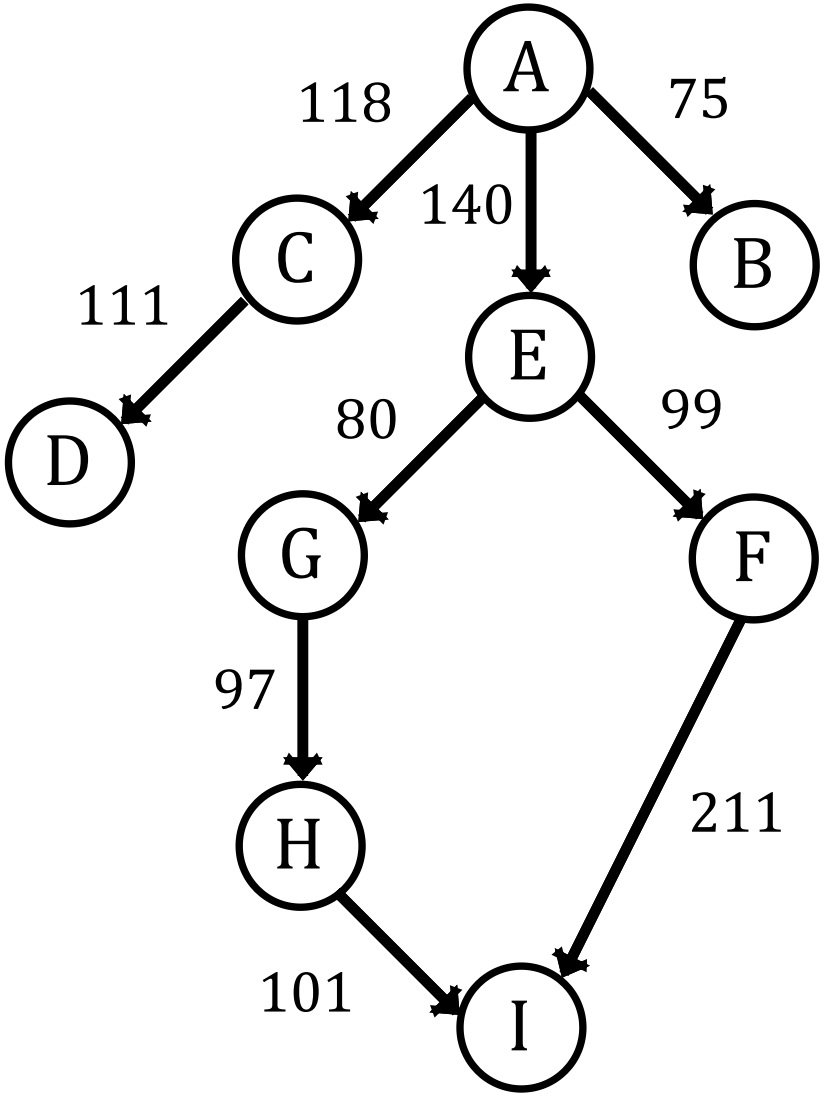
State	Heuristic: h(n)
A	366
B	374
C	329
D	244
E	253
F	178
G	193
H	98
I	0



A* Search (Problem 01)

Queue			
Path	**g(n)**	**h(n)**	**f(n)**

State	Heuristic: h(n)
A	366
B	374
C	329
D	244
E	253
F	178
G	193
H	98
I	0

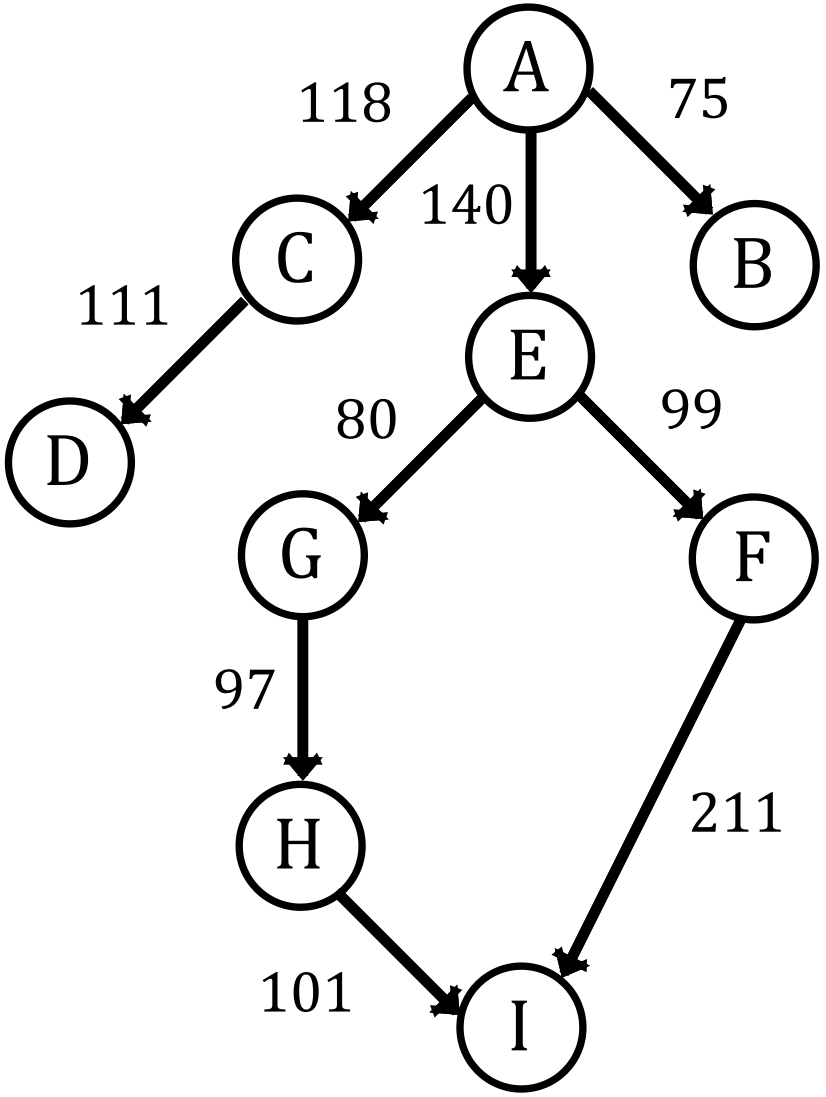


f(n) = g(n) + h(n)

A* Search (Problem 01)

Queue			
Path	**g(n)**	**h(n)**	**f(n)**
<A>	0	366	366

State	Heuristic: h(n)
A	366
B	374
C	329
D	244
E	253
F	178
G	193
H	98
I	0

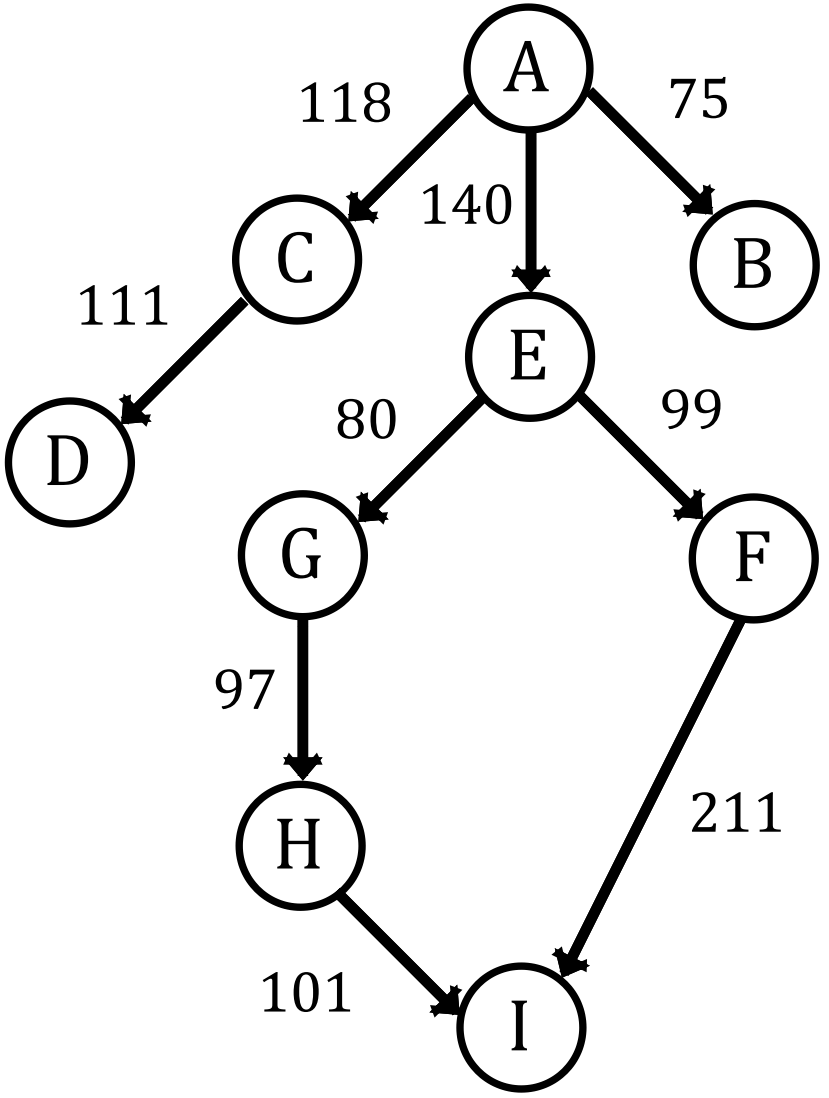


f(n) = g(n) + h(n)

A* Search (Problem 01)

Queue			
Path	**g(n)**	**h(n)**	**f(n)**
<E,A>	140	253	393
<C,A>	118	329	447
<B,A>	75	374	449

State	Heuristic: h(n)
A	366
B	374
C	329
D	244
E	253
F	178
G	193
H	98
I	0

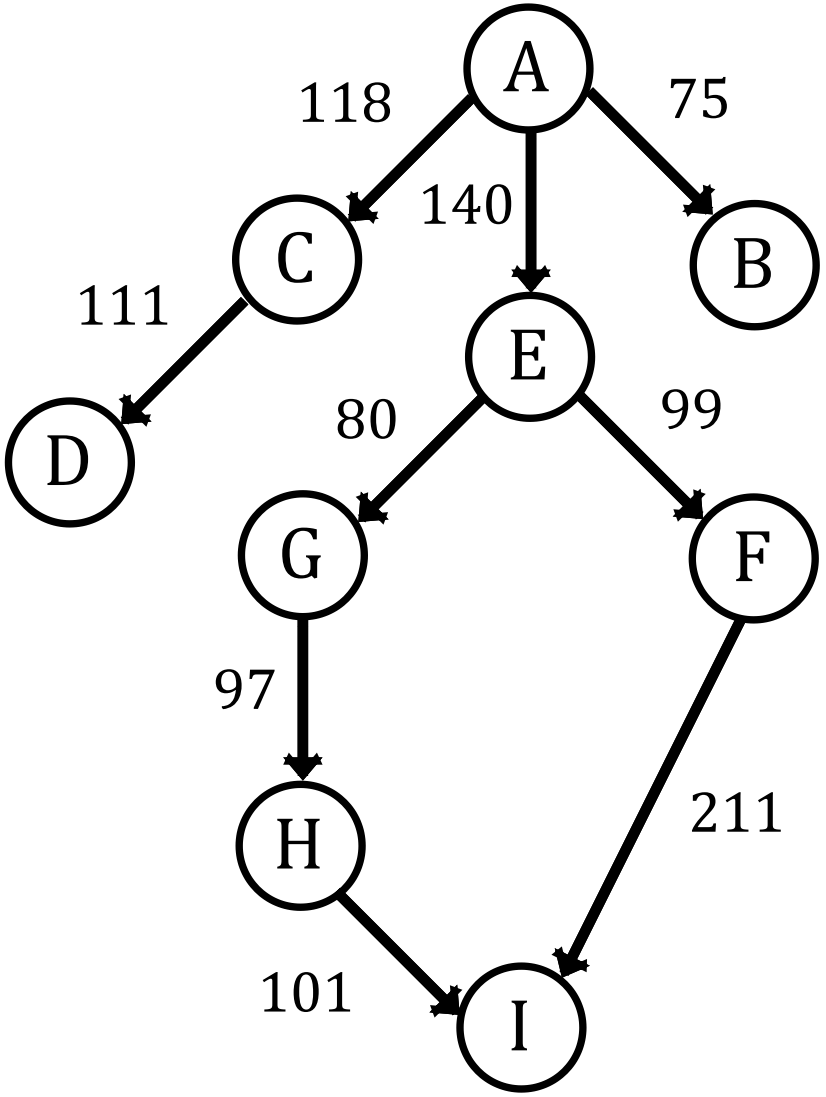


f(n) = g(n) + h(n)

A* Search (Problem 01)

Queue			
Path	**g(n)**	**h(n)**	**f(n)**
<G,E,A>	220	193	413
<F,E,A>	239	178	417
<C,A>	118	329	447
<B,A>	75	374	449

State	Heuristic: h(n)
A	366
B	374
C	329
D	244
E	253
F	178
G	193
H	98
I	0

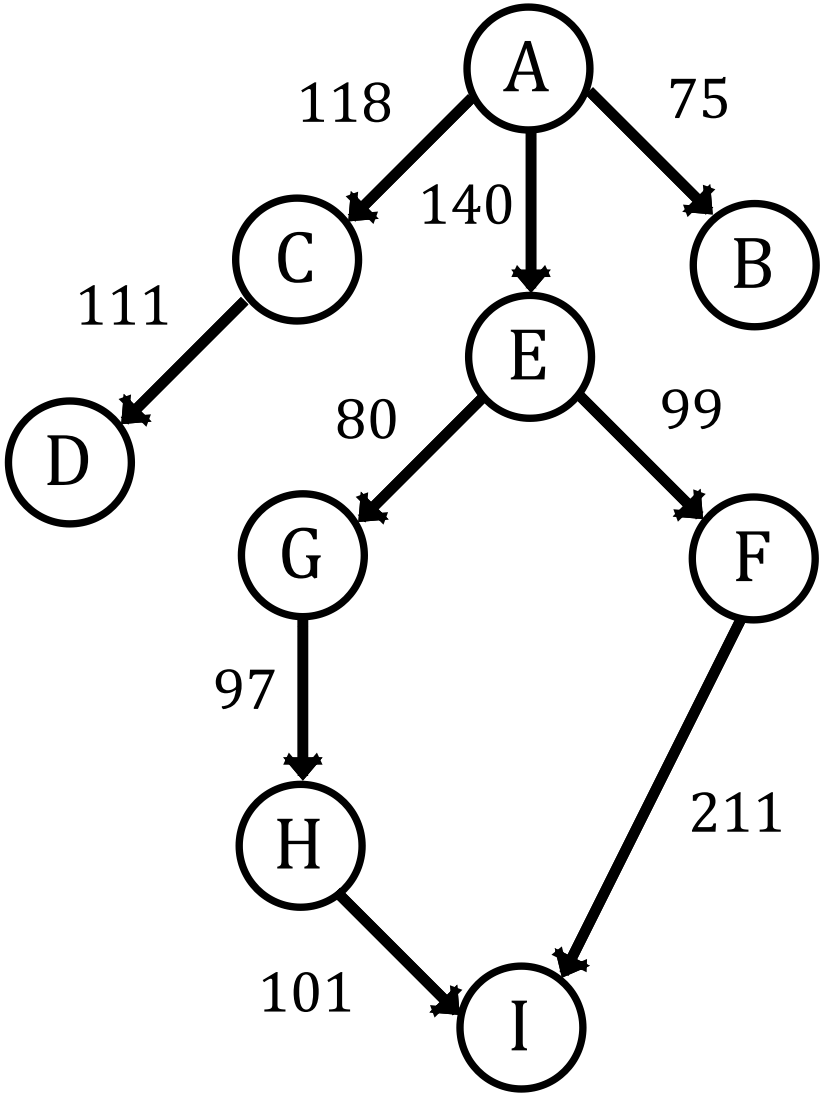


f(n) = g(n) + h(n)

A* Search (Problem 01)

Queue			
Path	**g(n)**	**h(n)**	**f(n)**
<H,G,E,A>	317	98	415
<F,E,A>	239	178	417
<C,A>	118	329	447
<B,A>	75	374	449

State	Heuristic: h(n)
A	366
B	374
C	329
D	244
E	253
F	178
G	193
H	98
I	0

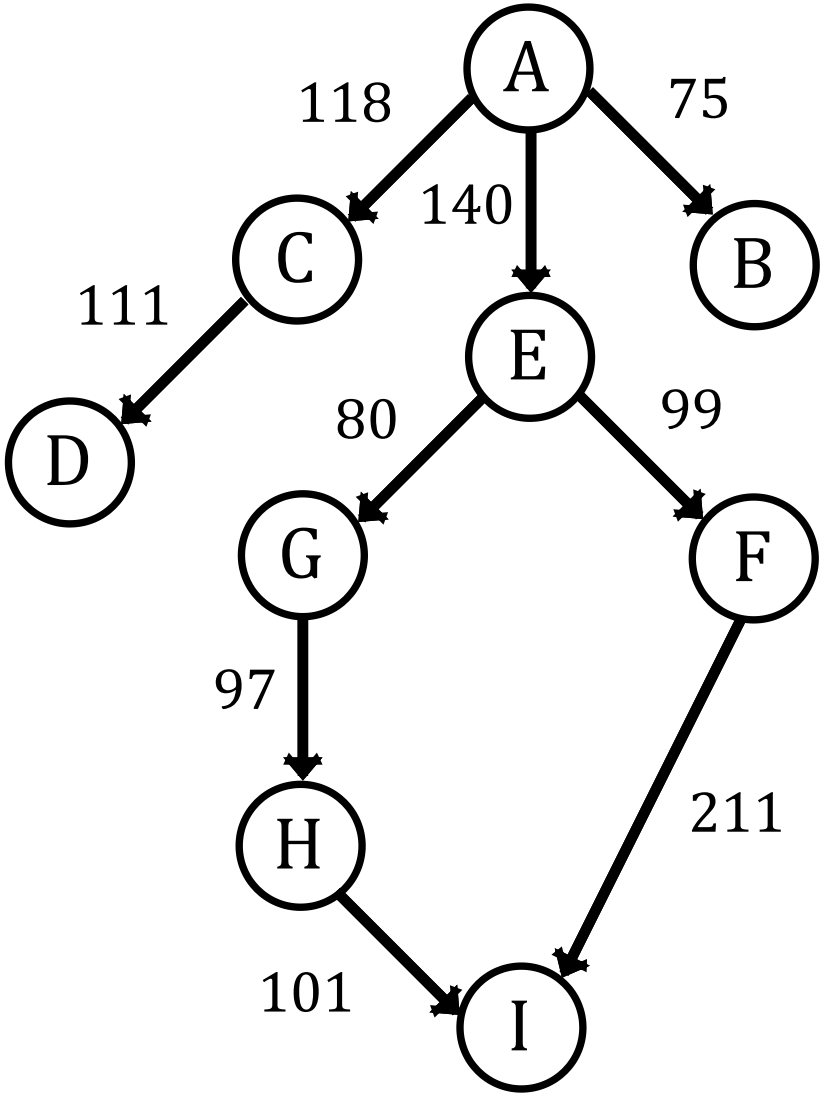


f(n) = g(n) + h(n)

A* Search (Problem 01)

Queue			
Path	**g(n)**	**h(n)**	**f(n)**
<F,E,A>	239	178	417
<I,H,G,E,A>	418	0	418
<C,A>	118	329	447
<B,A>	75	374	449

State	Heuristic: h(n)
A	366
B	374
C	329
D	244
E	253
F	178
G	193
H	98
I	0

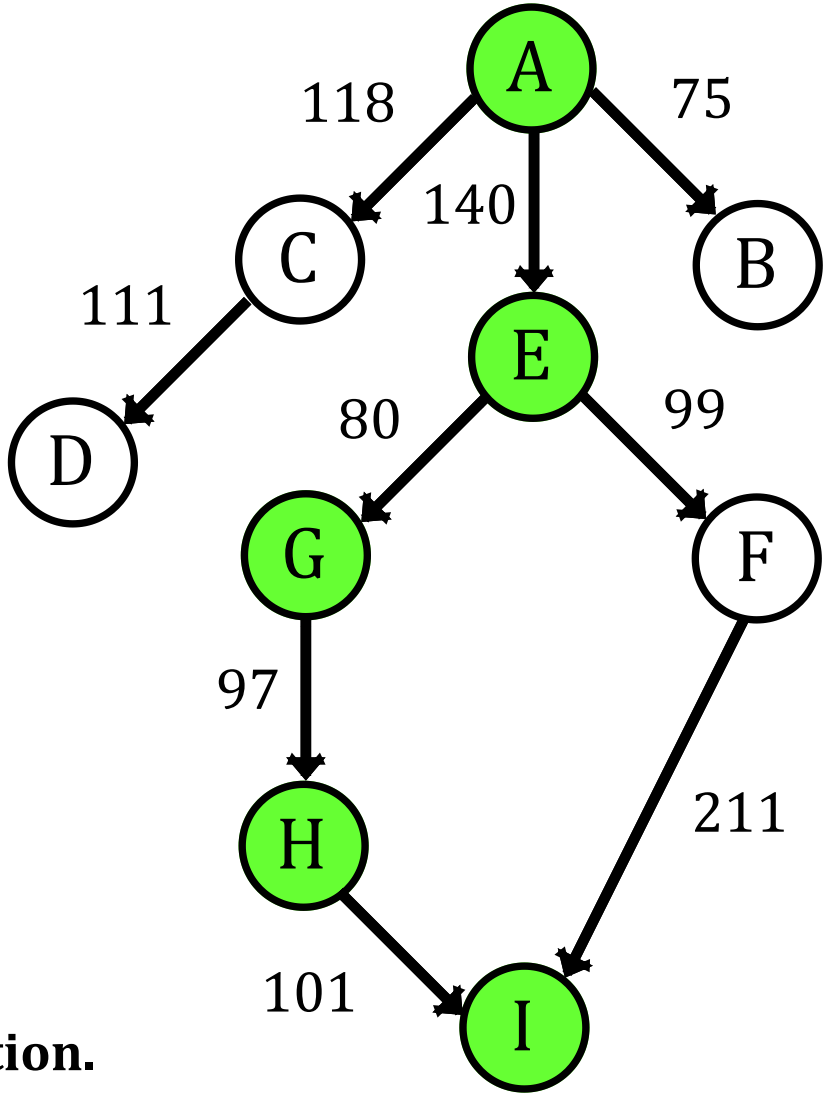


f(n) = g(n) + h(n)

A* Search (Problem 01)

Queue			
Path	**g(n)**	**h(n)**	**f(n)**
<I,H,G,E,A>	418	0	418
<C,A>	118	329	447
<B,A>	75	374	449
<I,F,E,A>	239	0	450

State	Heuristic: h(n)
A	366
B	374
C	329
D	244
E	253
F	178
G	193
H	98
I	0

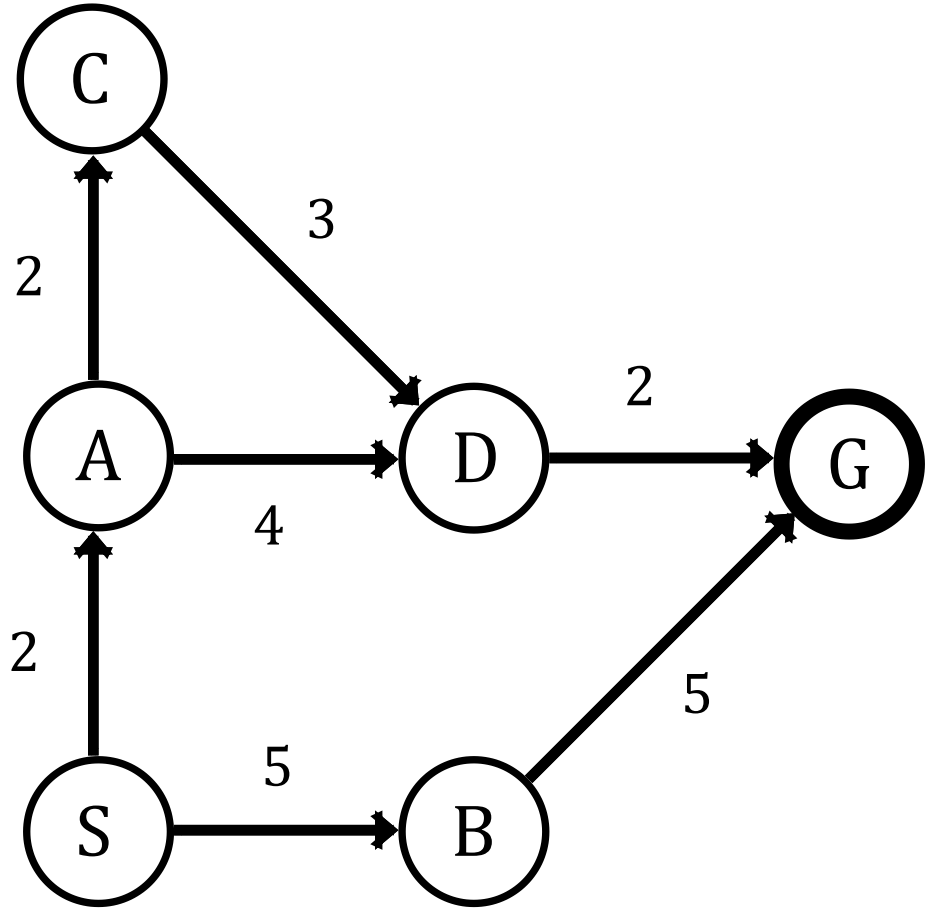


Solution = A → E → G → H → I
Path Cost = 140+80+97+101 = 418

As minimum path is the goal path, so we have found a solution.

A* Search (Problem 02)

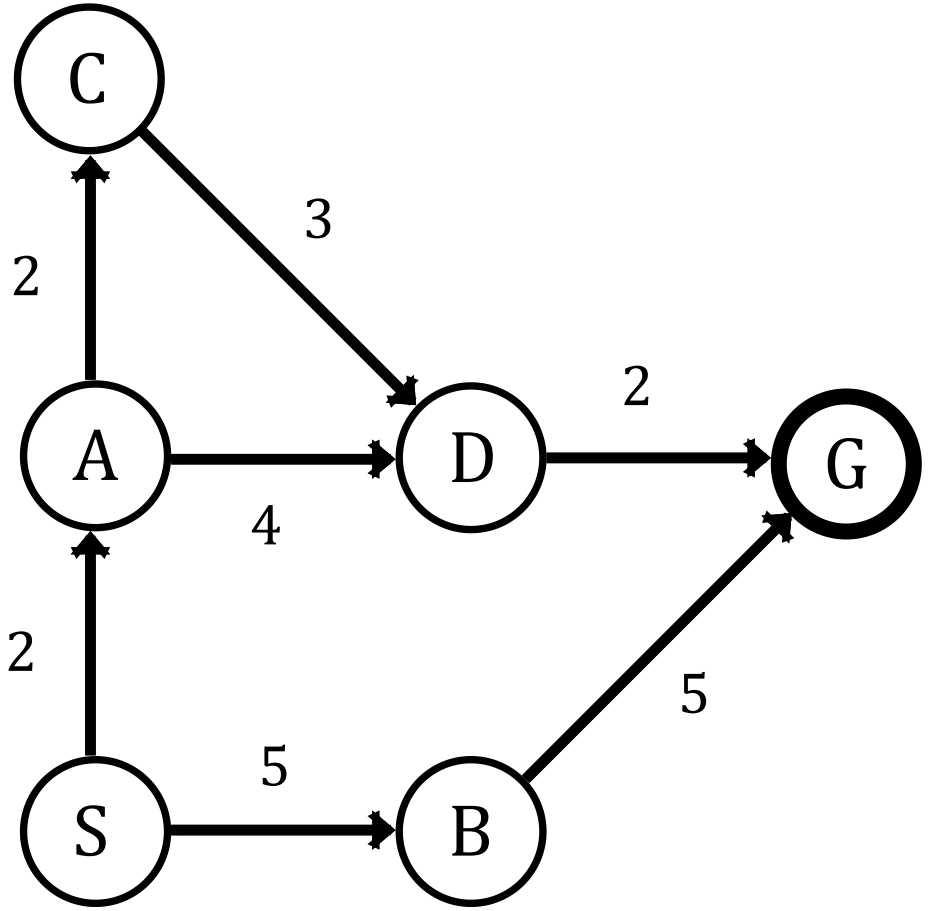
- **S = Initial State**
- **G = Goal State**



A* Search (Problem 02)

f(n) = h(n) = Straight Line Distance

State	Heuristic: h(n)
S	10
A	2
B	3
C	1
D	4
G	0

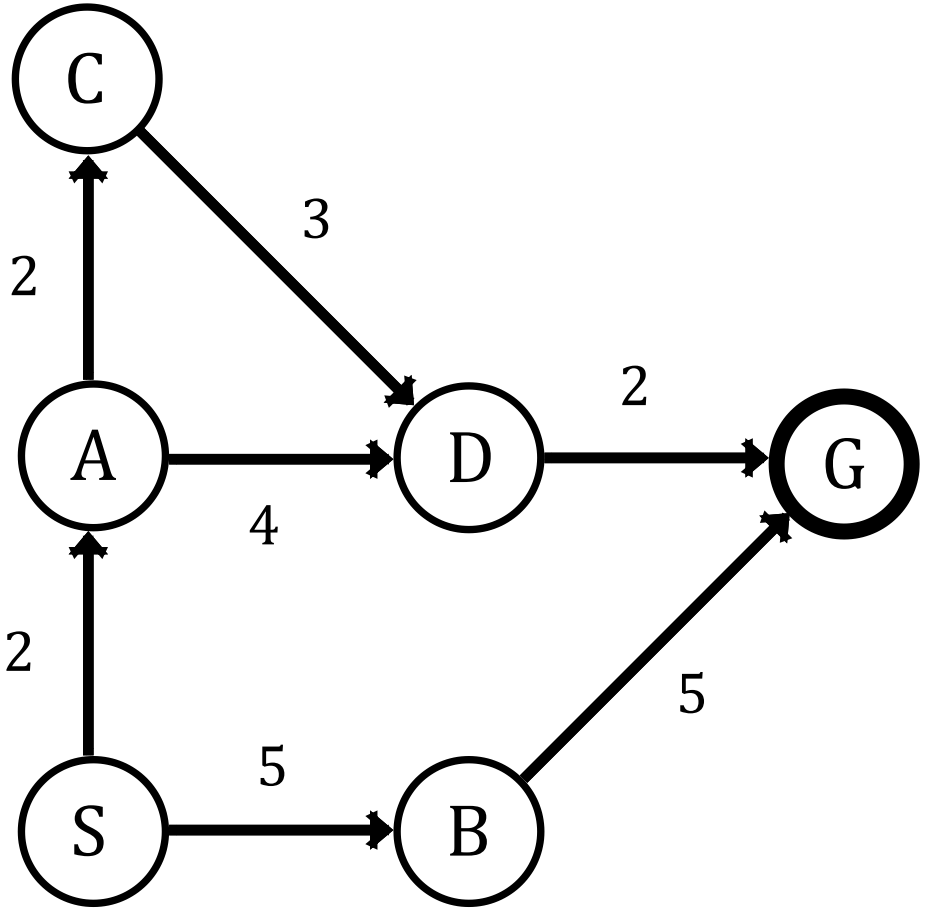


f(n) = g(n) + h(n)

A* Search (Problem 02)

Queue			
Path	**g(n)**	**h(n)**	**f(n)**

State	Heuristic: h(n)
S	10
A	2
B	3
C	1
D	4
G	0

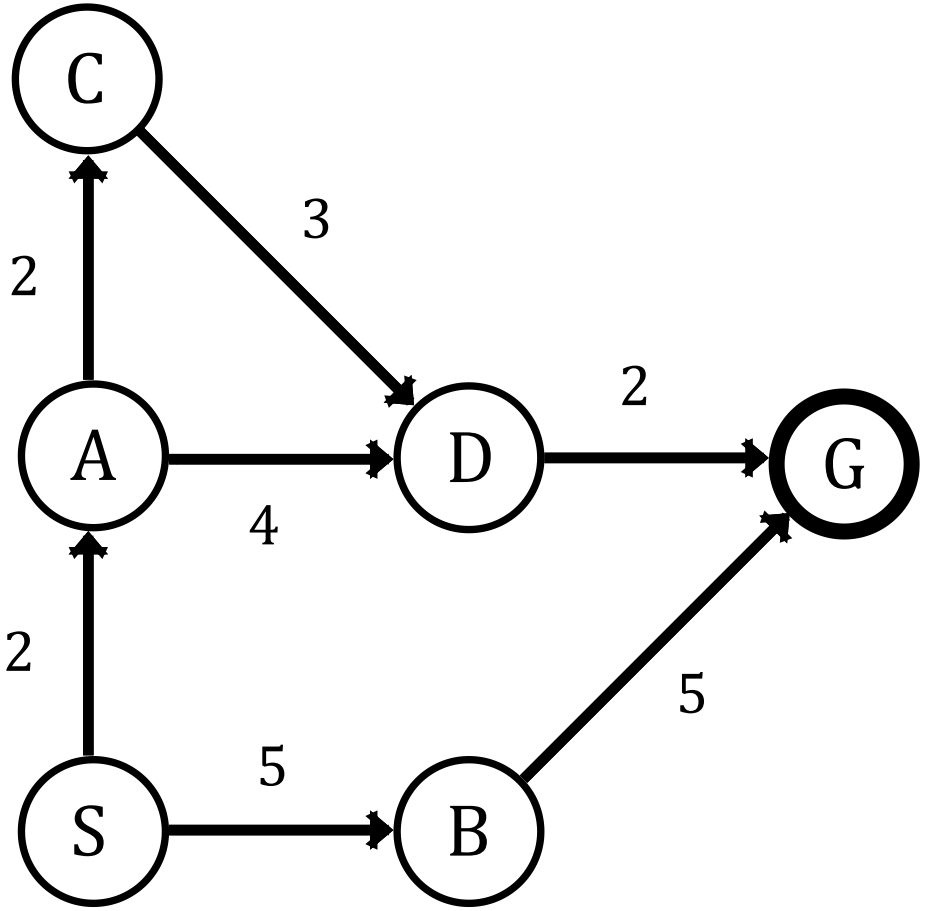


f(n) = g(n) + h(n)

A* Search (Problem 02)

Queue			
Path	**g(n)**	**h(n)**	**f(n)**
<S>	0	10	10

State	Heuristic: h(n)
S	10
A	2
B	3
C	1
D	4
G	0

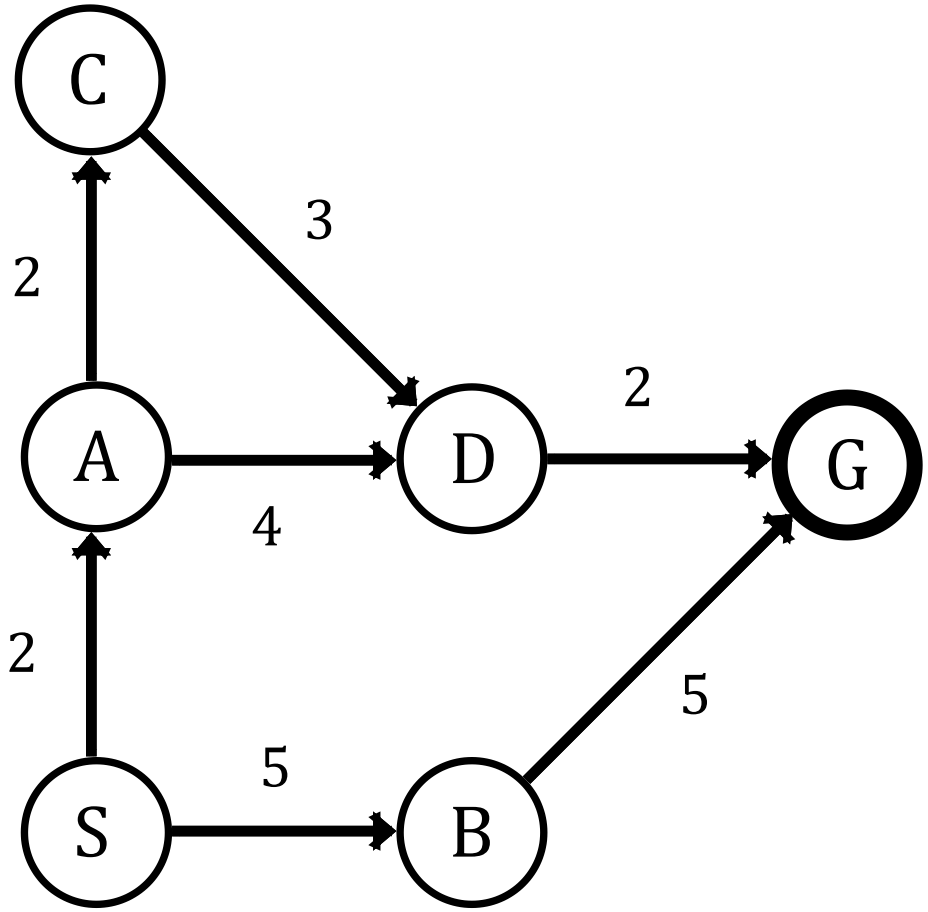


f(n) = g(n) + h(n)

A* Search (Problem 02)

Queue			
Path	**g(n)**	**h(n)**	**f(n)**
<A,S>	2	2	4
<B,S>	5	3	8

State	Heuristic: h(n)
S	10
A	2
B	3
C	1
D	4
G	0

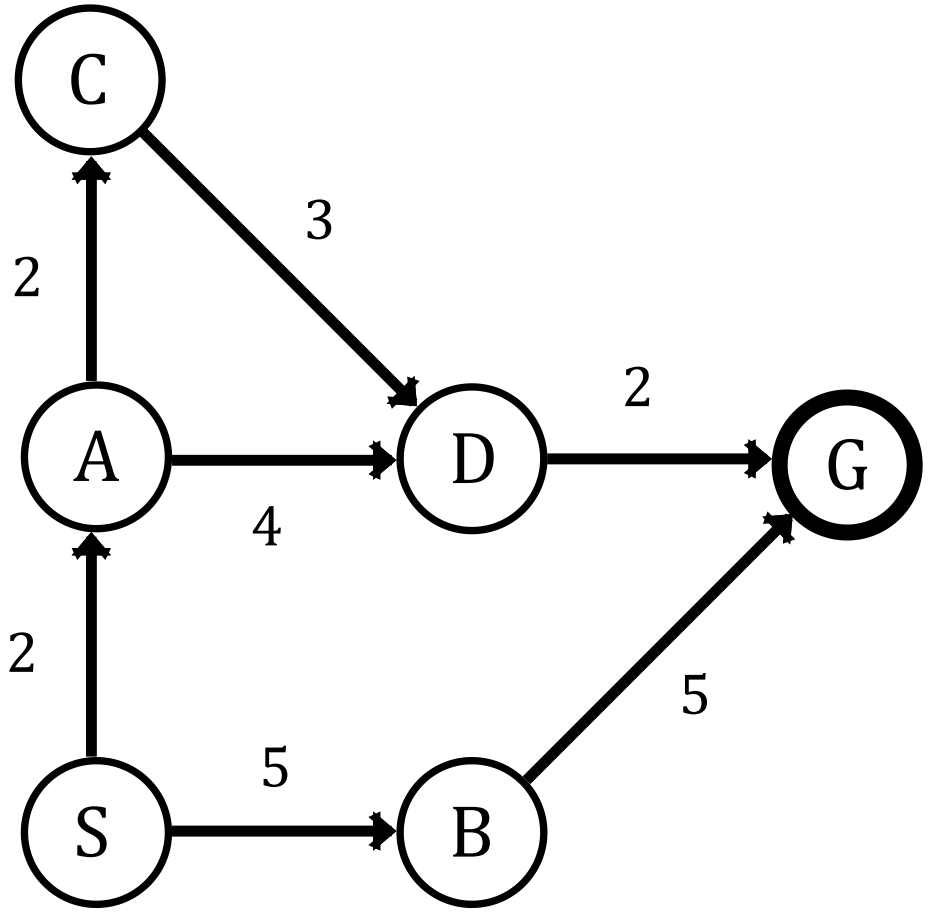


f(n) = g(n) + h(n)

A* Search (Problem 02)

Queue			
Path	**g(n)**	**h(n)**	**f(n)**
<C,A,S>	4	1	5
<B,S>	5	3	8
<D,A,S>	6	4	10

State	**Heuristic: h(n)**
S	10
A	2
B	3
C	1
D	4
G	0

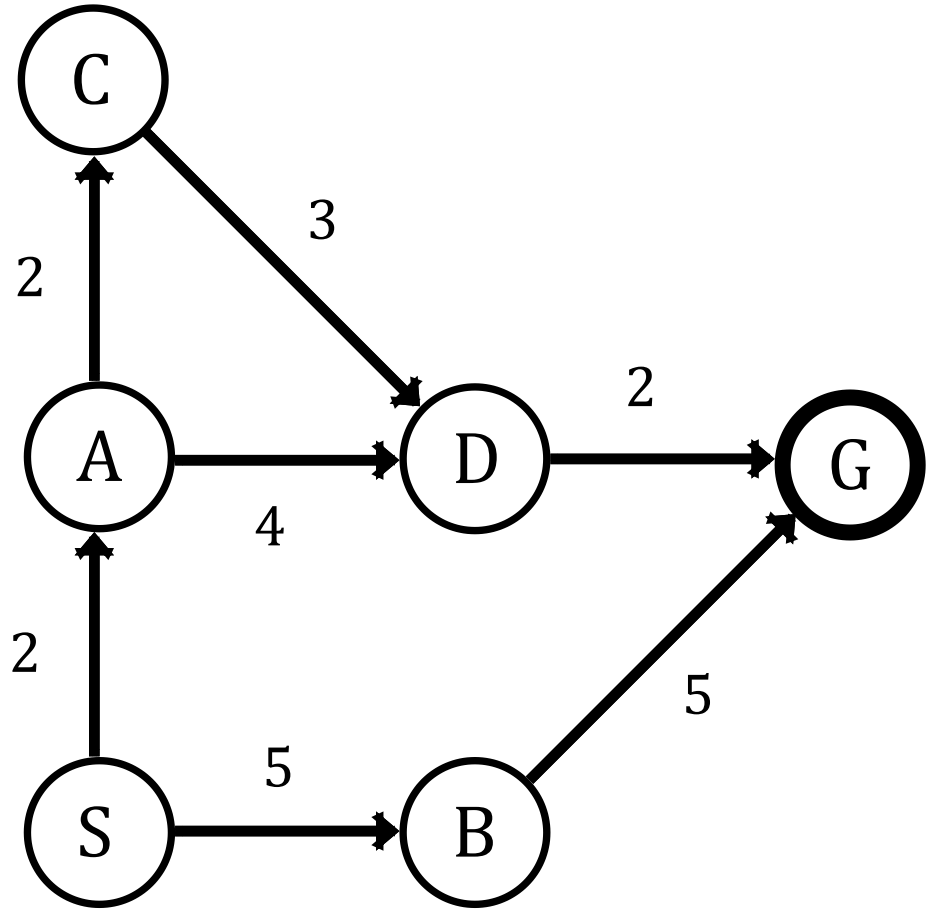


f(n) = g(n) + h(n)

A* Search (Problem 02)

Queue			
Path	**g(n)**	**h(n)**	**f(n)**
<B,S>	5	3	8
<D,A,S>	6	4	10
<D,C,A,S>	7	4	11

State	**Heuristic: h(n)**
S	10
A	2
B	3
C	1
D	4
G	0

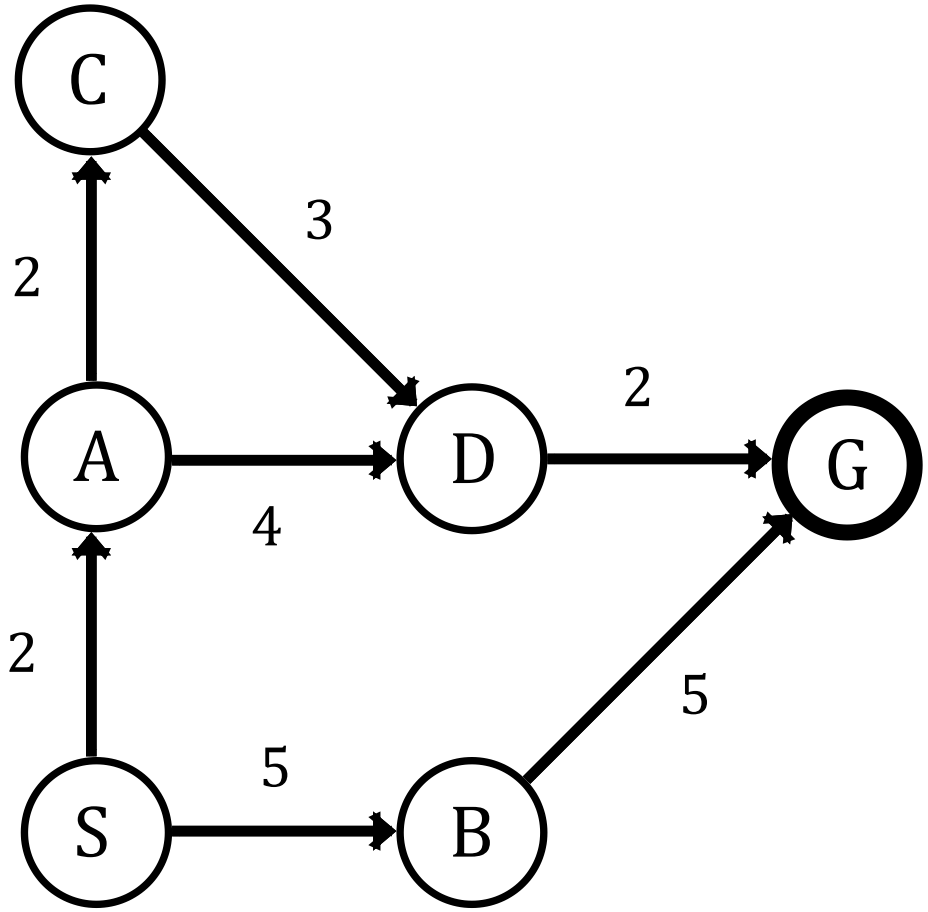


f(n) = g(n) + h(n)

A* Search (Problem 02)

Queue			
Path	**g(n)**	**h(n)**	**f(n)**
<D,A,S>	6	4	10
<G,B,S>	10	0	10
<D,C,A,S>	7	4	11

State	**Heuristic: h(n)**
S	10
A	2
B	3
C	1
D	4
G	0

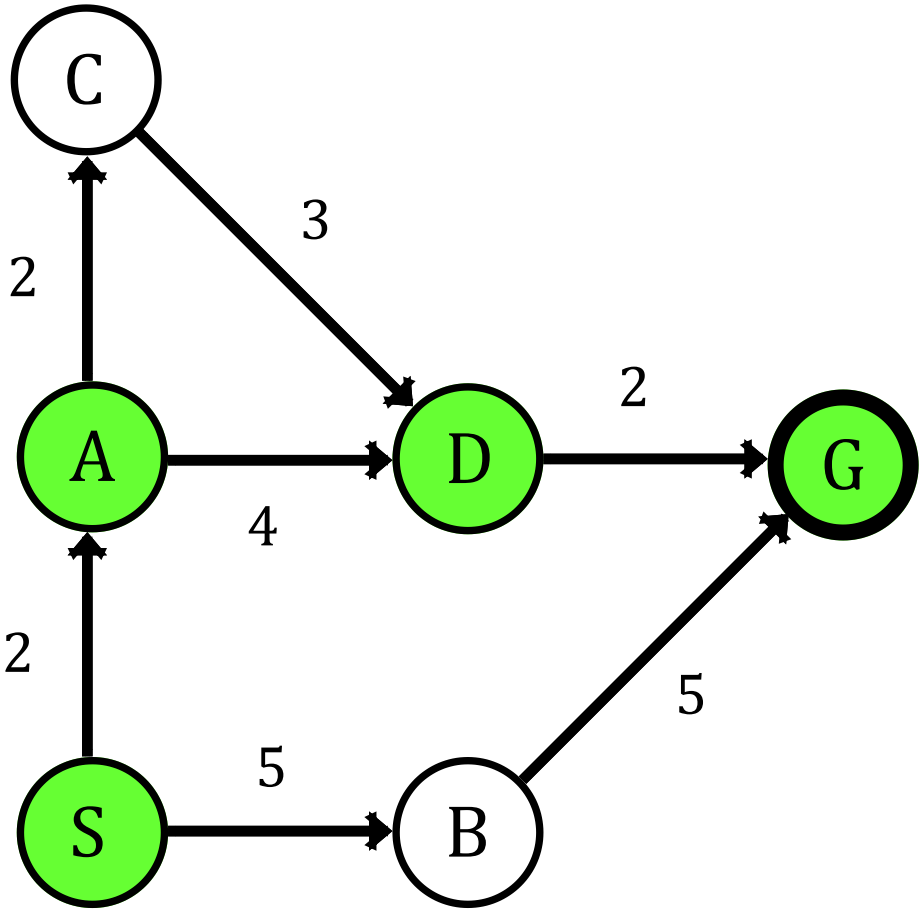


f(n) = g(n) + h(n)

A* Search (Problem 02)

Queue			
Path	**g(n)**	**h(n)**	**f(n)**
<G,D,A,S>	8	0	8
<G,B,S>	10	0	10
<D,C,A,S>	7	4	11

State	**Heuristic: h(n)**
S	10
A	2
B	3
C	1
D	4
G	0



Solution = S → A → D → G
Path Cost = 2+4+2 = 8

As minimum path is the goal path, so we have found a solution.