Best first Search

* in BFS & DFS when we are at a node, we can consider any of the adjacent as next node.
  So both BFS & DFS blindly explore paths without considering any cost function.

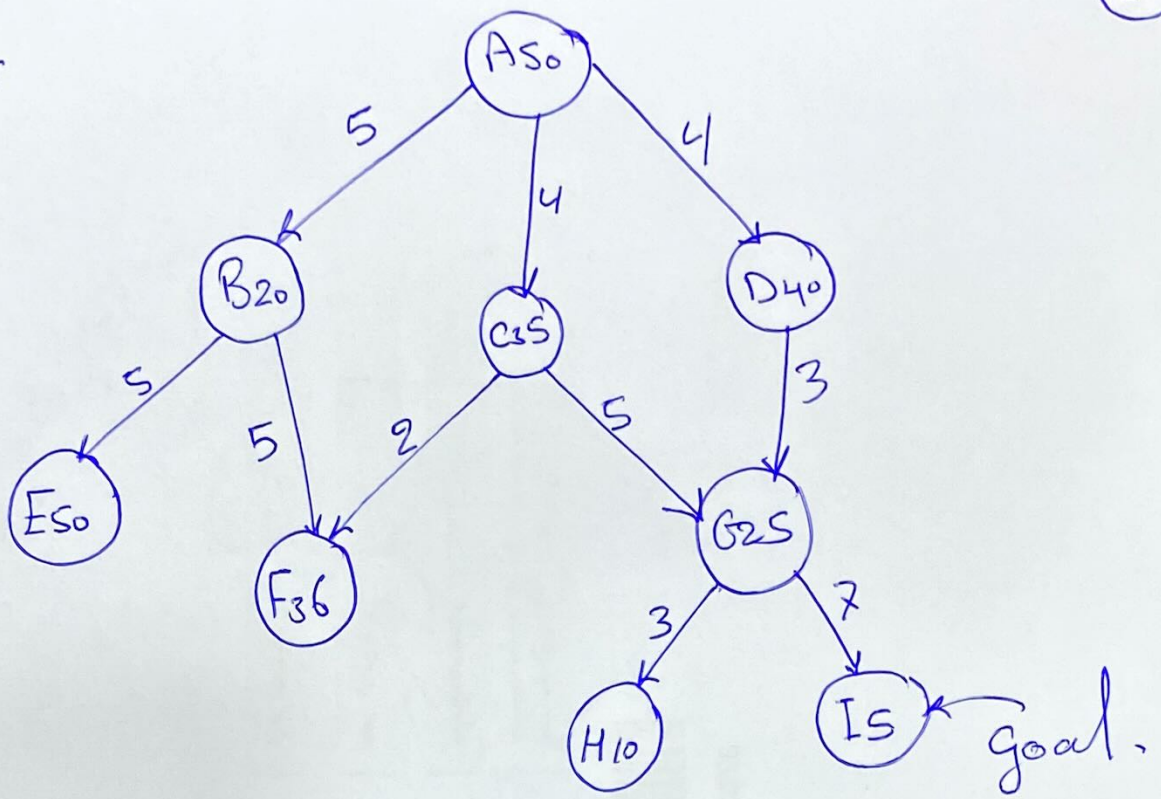
* in Best first Search, we will use an evaluation function to decide which adjacent is most promising and then explore.

* We use a priority queue to store the cost of node.
  So the implementation is a variation of Breadth first search we just need to change the queue to priority queue.

* Advantage of Best first Search:-
  - Can switch between BFS & DFS, thus gaining the advantages of the both.
  - more efficient when compared to DFS.

* Disadvantage of Best first Search:-
  - The chance of getting stuck in a loop are higher.

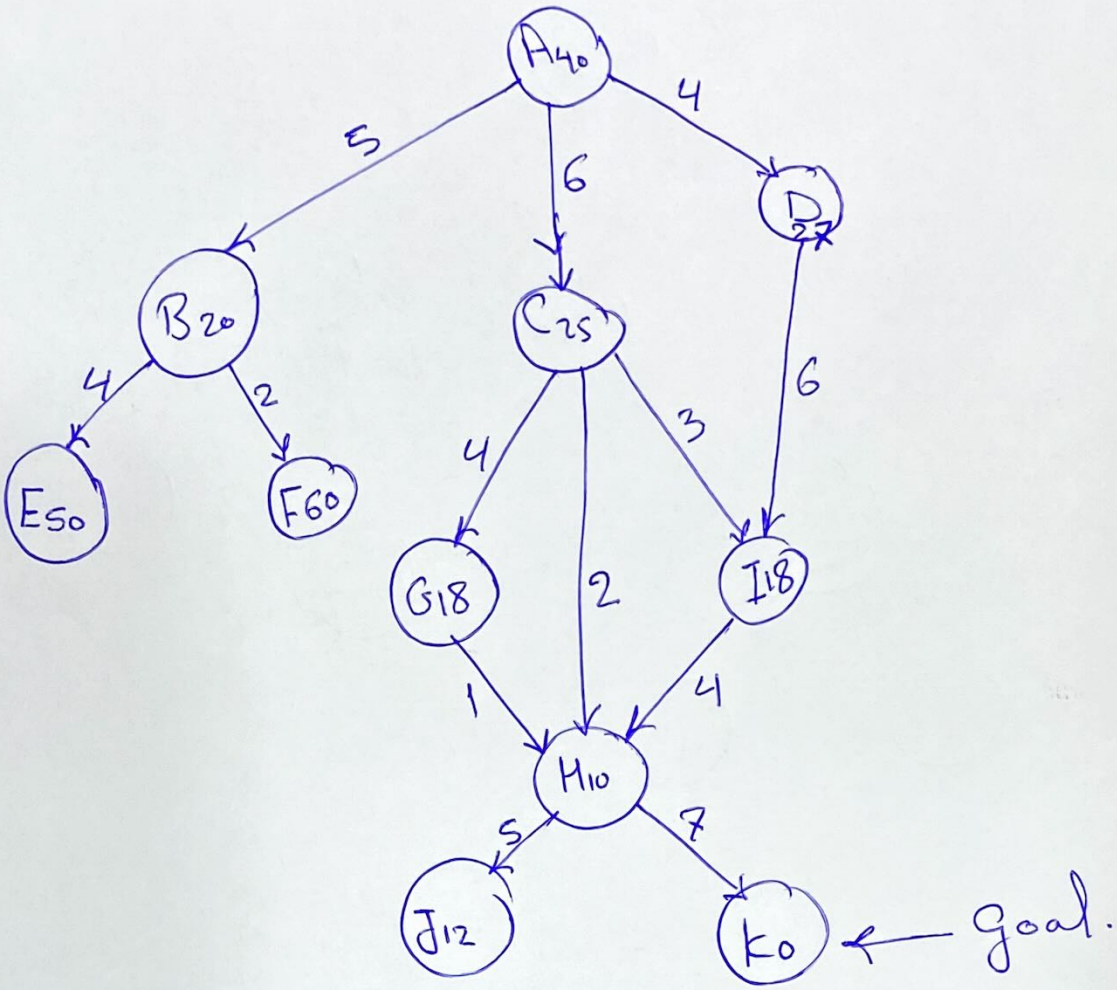
ex:-



CS	open	close
		[ ]
A50	A50	A50
B20	B20 C35 D40	A50 B20
C35	C35 F36 D40 E50	A50 B20 C35
G25	G25 E36 D40 E50	A50 B20 C35 G25
I5	I5 H10 E36 D40 E50	A50 B20 C35 G25 I5
stop		

Solution path:-

$A_0 \rightarrow B_5 \rightarrow C_9 \rightarrow G_5 \rightarrow I_7$



Cs	open	close
		[ ]
A_{40}	A_{40}	
B_{20}	B_{20} C_{25} D_{27}	A_{40}
C_{25}	C_{25} D_{27} E_{50} F_{60}	A_{40} B_{20}
H_{10}	H_{10} G_{18} I_{18} D_{27} E_{50} F_{60}	A_{40} B_{20} C_{25}
K_0	K_0 J_{12} G_{18} I_{18} D_{27} E_{50} F_{60}	A_{40} B_{20} C_{25} H_{10}
	Stop.	A_{40} B_{20} C_{25} H_{10} K_0

$A_0 \rightarrow B_5 \rightarrow C_{11} \rightarrow H_2 \rightarrow K_7$

Solution path.