DS & AI ENGINEERING

Artificial Intelligence

Informed search

Lecture No.- 03

By- Aditya sir

Recap of Previous Lecture

- Topic
- Topic
- Topic
- Topic

Properties of Heuristics
- Admissible
- Consistent

A* Search
- Tree Search
- Graph Search

Topics to be Covered

- Topic
- Topic
- Topic

A* Search Continued...

About Aditya Jain sir

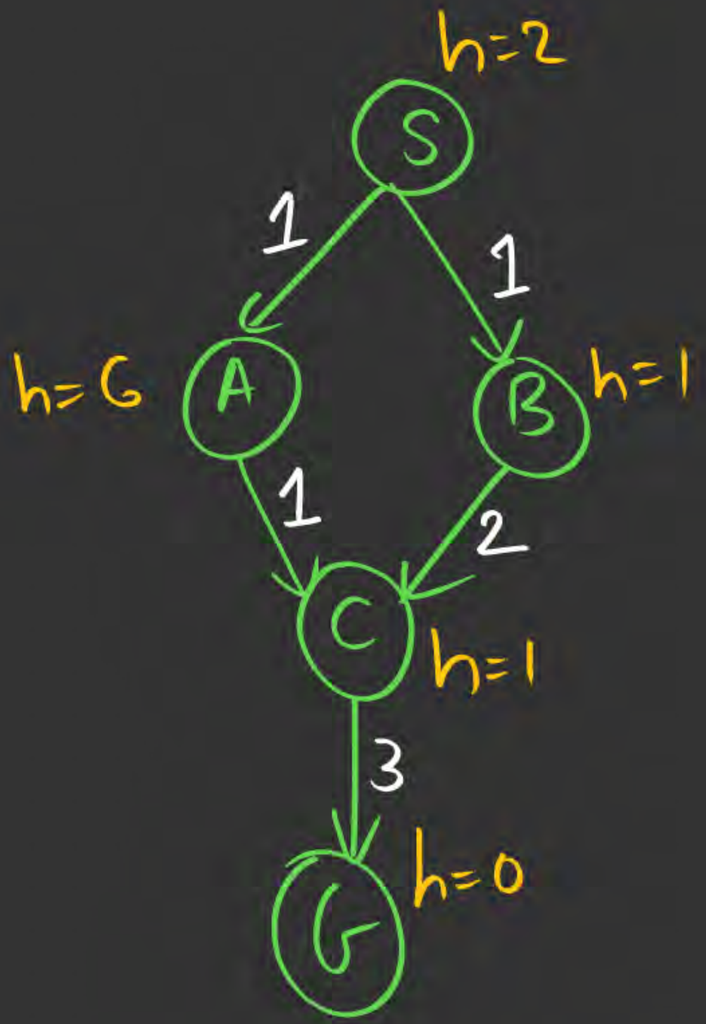
1. Appeared for GATE during BTech and secured AIR 60 in GATE in very first attempt - City topper
2. Represented college as the first Google DSC Ambassador.
3. The only student from the batch to secure an internship at Amazon. (9+ CGPA)
4. Had offer from IIT Bombay and IISc Bangalore to join the Masters program
5. Joined IIT Bombay for my 2 year Masters program, specialization in Data Science
6. Published multiple research papers in well known conferences along with the team
7. Received the prestigious excellence in Research award from IIT Bombay for my Masters thesis in ML
8. Completed my Masters with an overall GPA of 9.36/10
9. Joined Dream11 as a Data Scientist
10. Have mentored 15,000+ students & working professions in field of Data Science and Analytics
11. Have been mentoring & teaching GATE aspirants to secure a great rank in limited time
12. Have got around 27.5K followers on Linkedin where I share my insights and guide students and professionals.

Revision : Graph and Tree Search

- **Graph Search** → We make a closed list and never visit closed node.
- **Tree Search** → We do not make closed list.

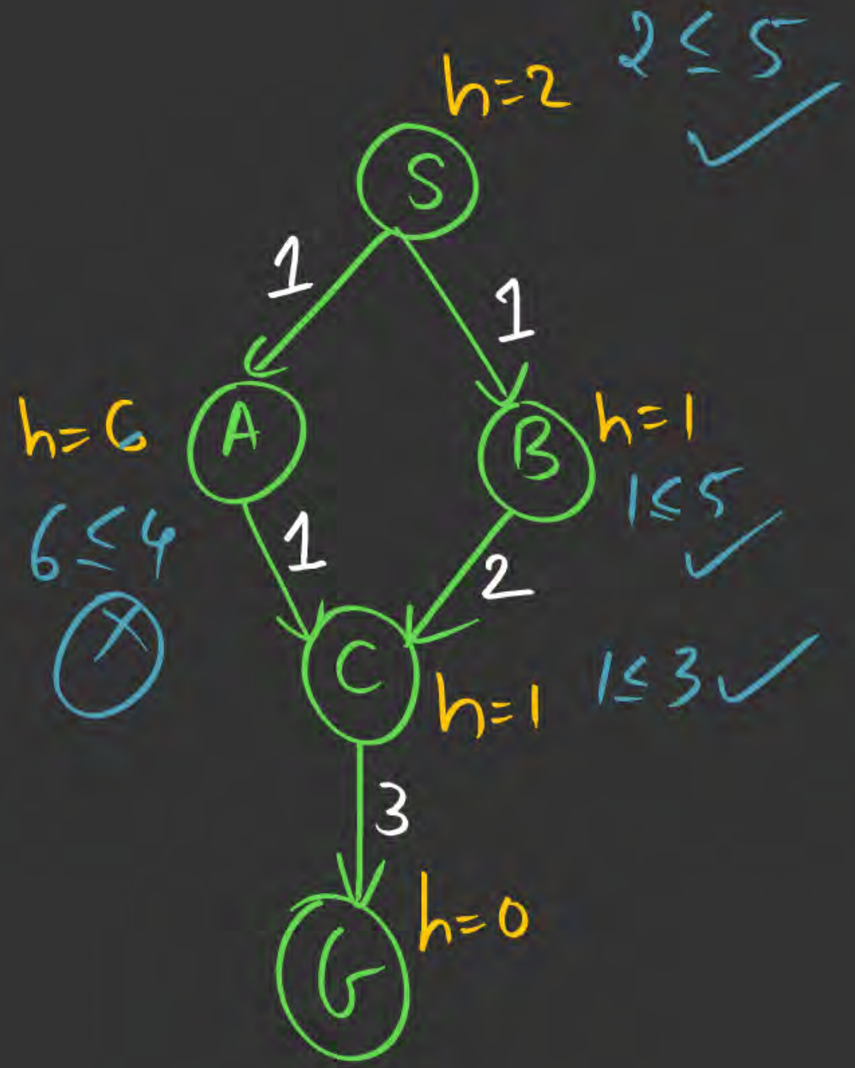
Revision : Graph and Tree Search

- **Graph Search** → Consist h optimal solution.
- **Tree Search** → Admissible h optimal solution.



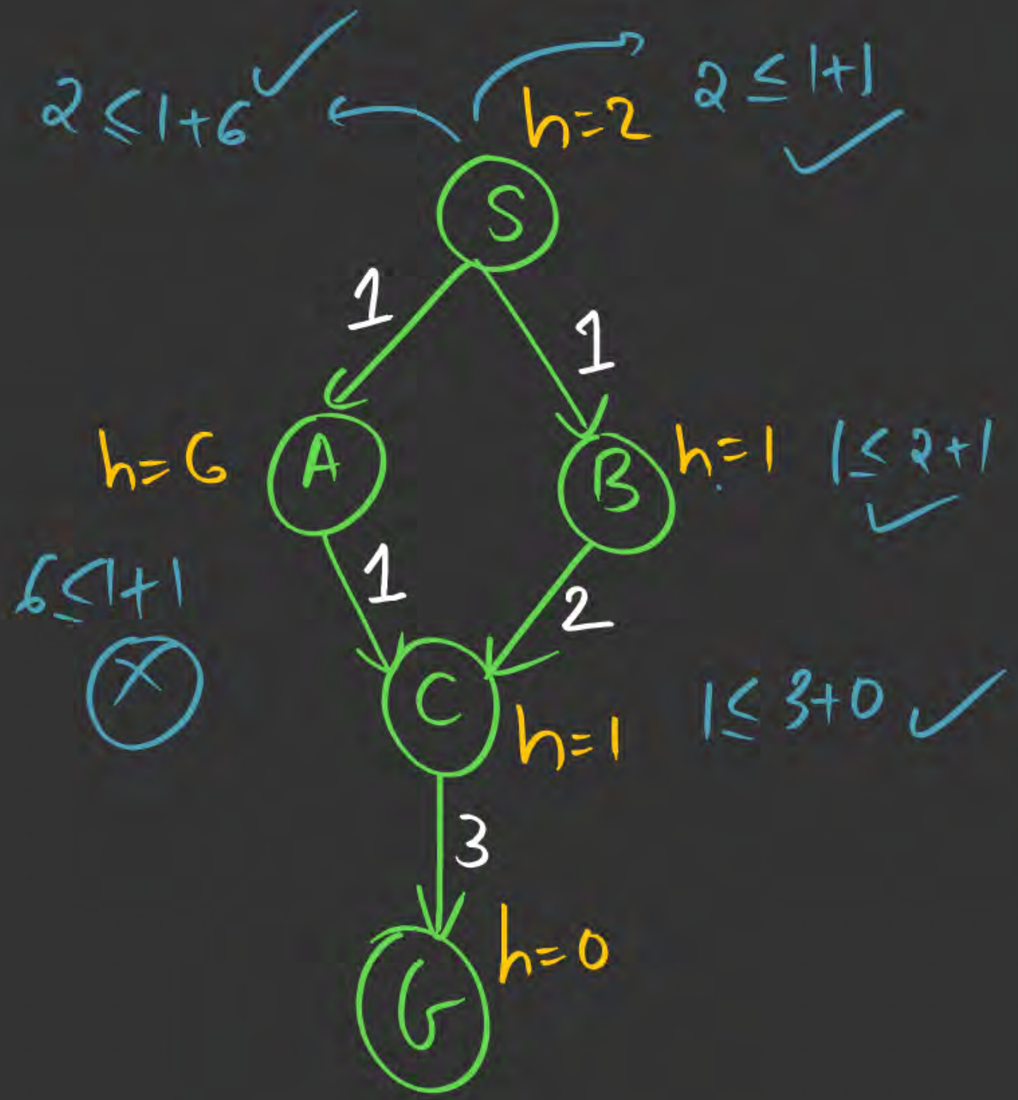
1) Admissible ?

2) Consistent ?



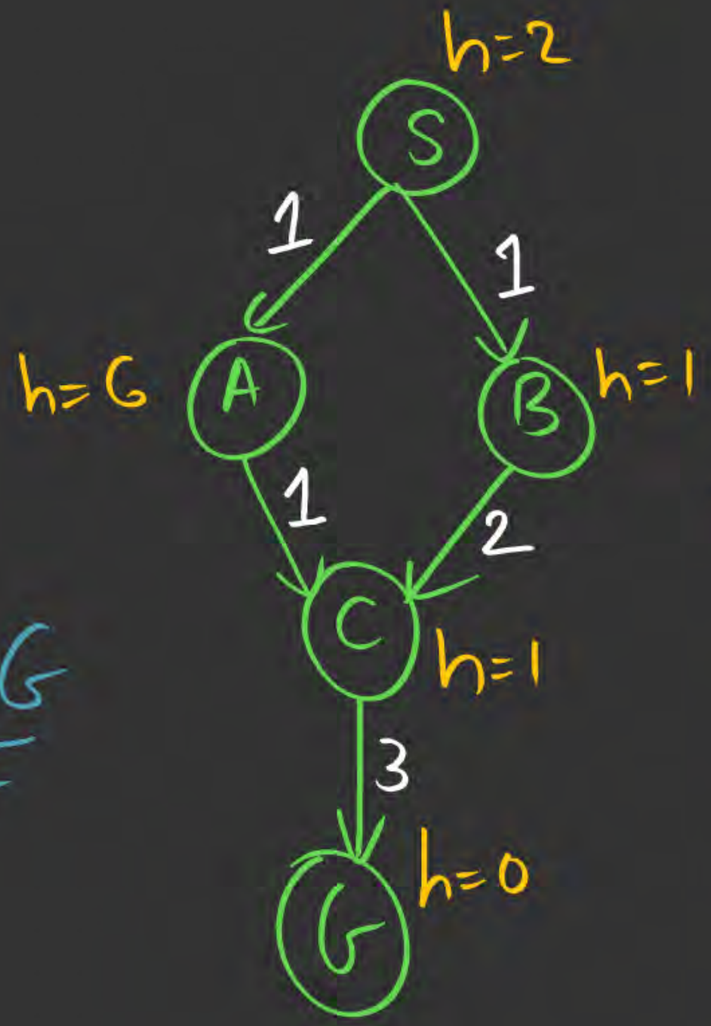
1) Admissible? ⟶ No

$h(n) \leq h^*(n)$



2) Consistent? ⟶ No

$h(n) \leq c(n, n') + h(n')$



S → G

1) A* Graph Search:

Open	closed
S x	S B C G
A	7 7 7
B	(2) 2 2
C	x (4) 4
G	x x (6)

↳ Stop

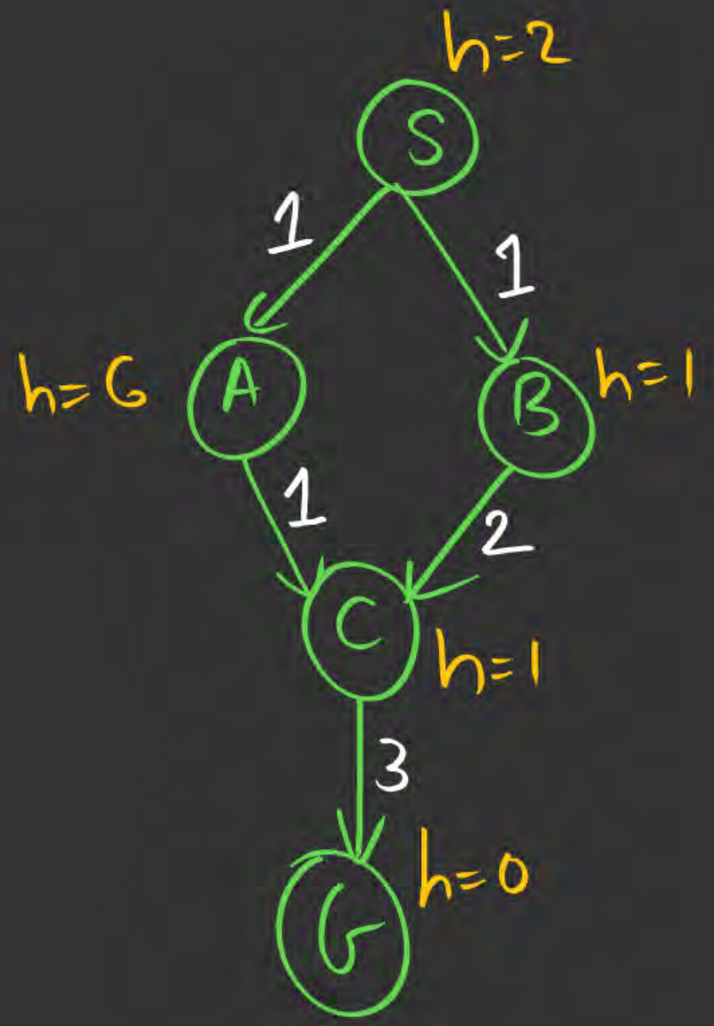
SBCG

Cost = 6

Bad Heuristics

A x
C x

not optimal



② A* Tree Search:

h → Ax
Cx

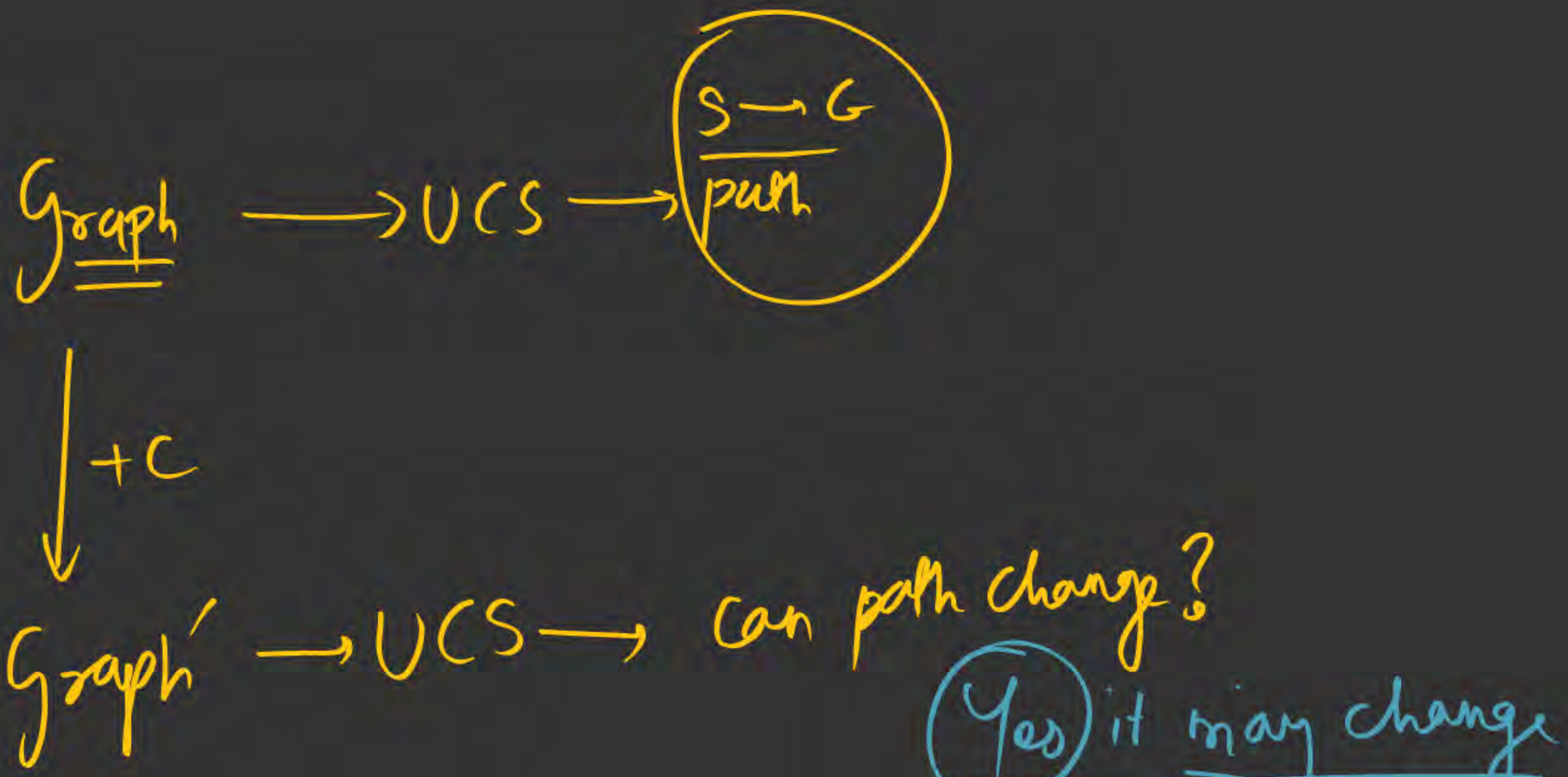
Open	Closed			
SX	S	B	C	G
A	7	7	7	↳ stop
B	(2)	2	2	
C	X	(4)	4	
G	X	X	(6)	

not optimal

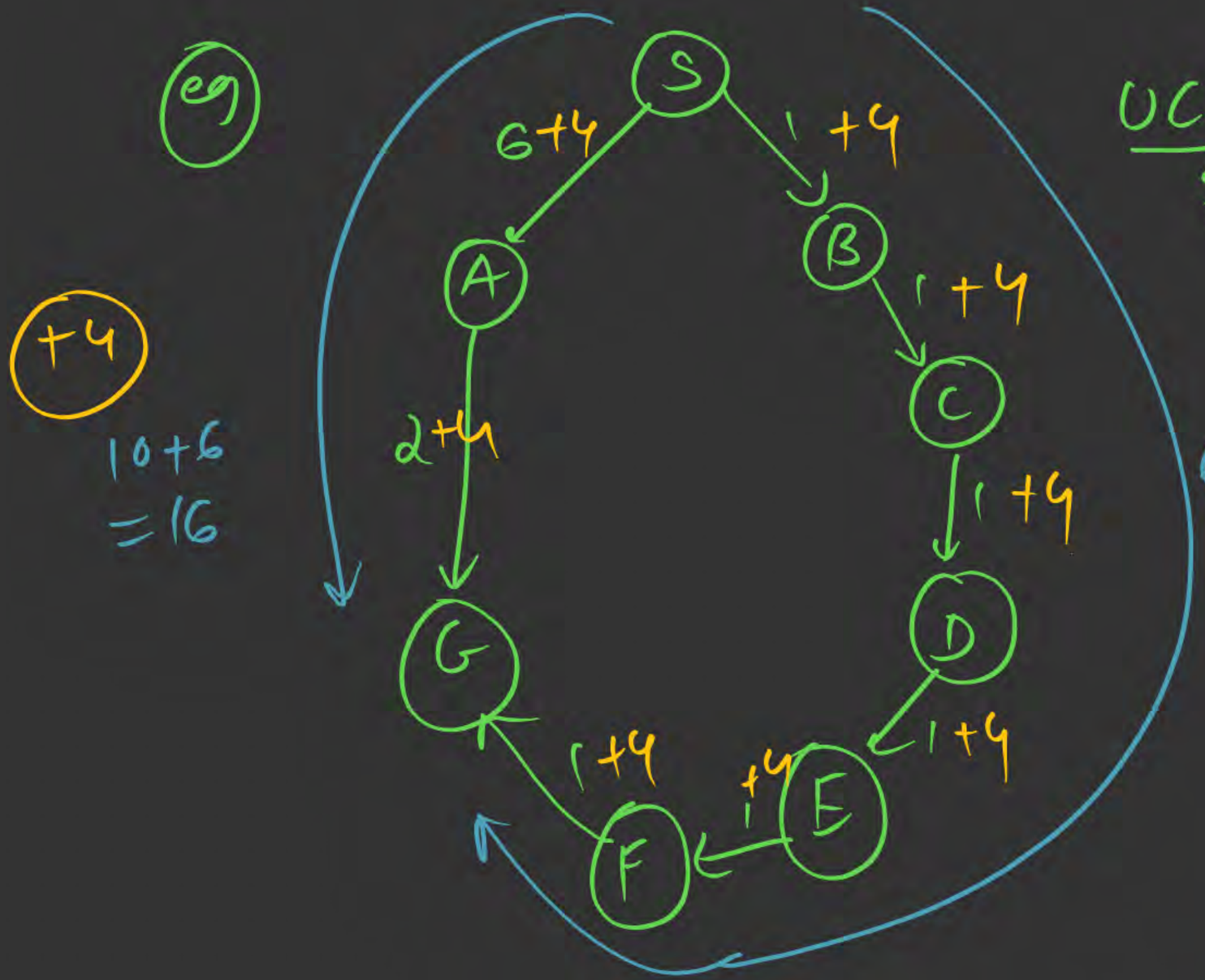
SBCG

Cost = 6

(Q.1)


Graph
UCS
S → G
path
+c
Graph'
UCS
can path change?
Yes it may change

eg



+4

10+6
=16

UCS

SBCDEFG

Cost = 6

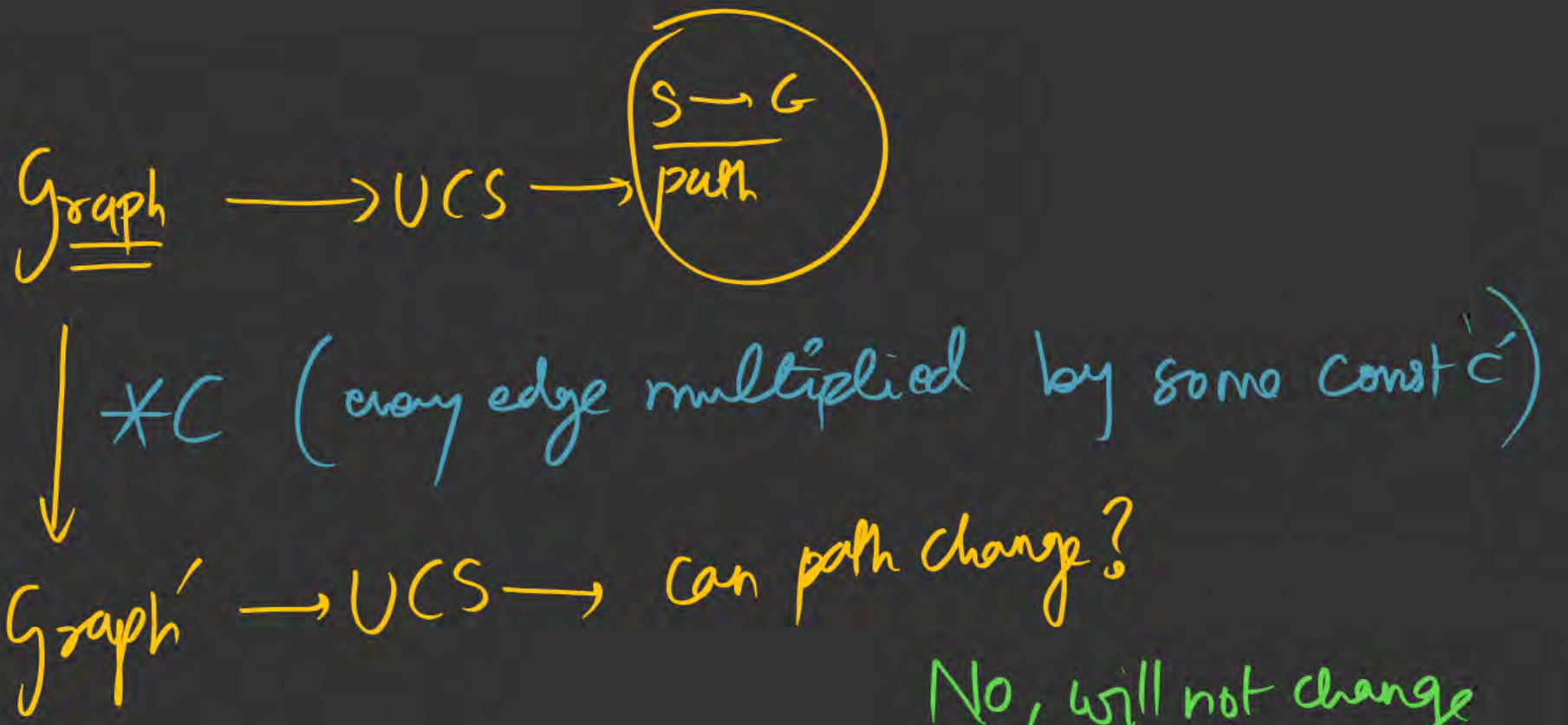
6+6x4 = 30

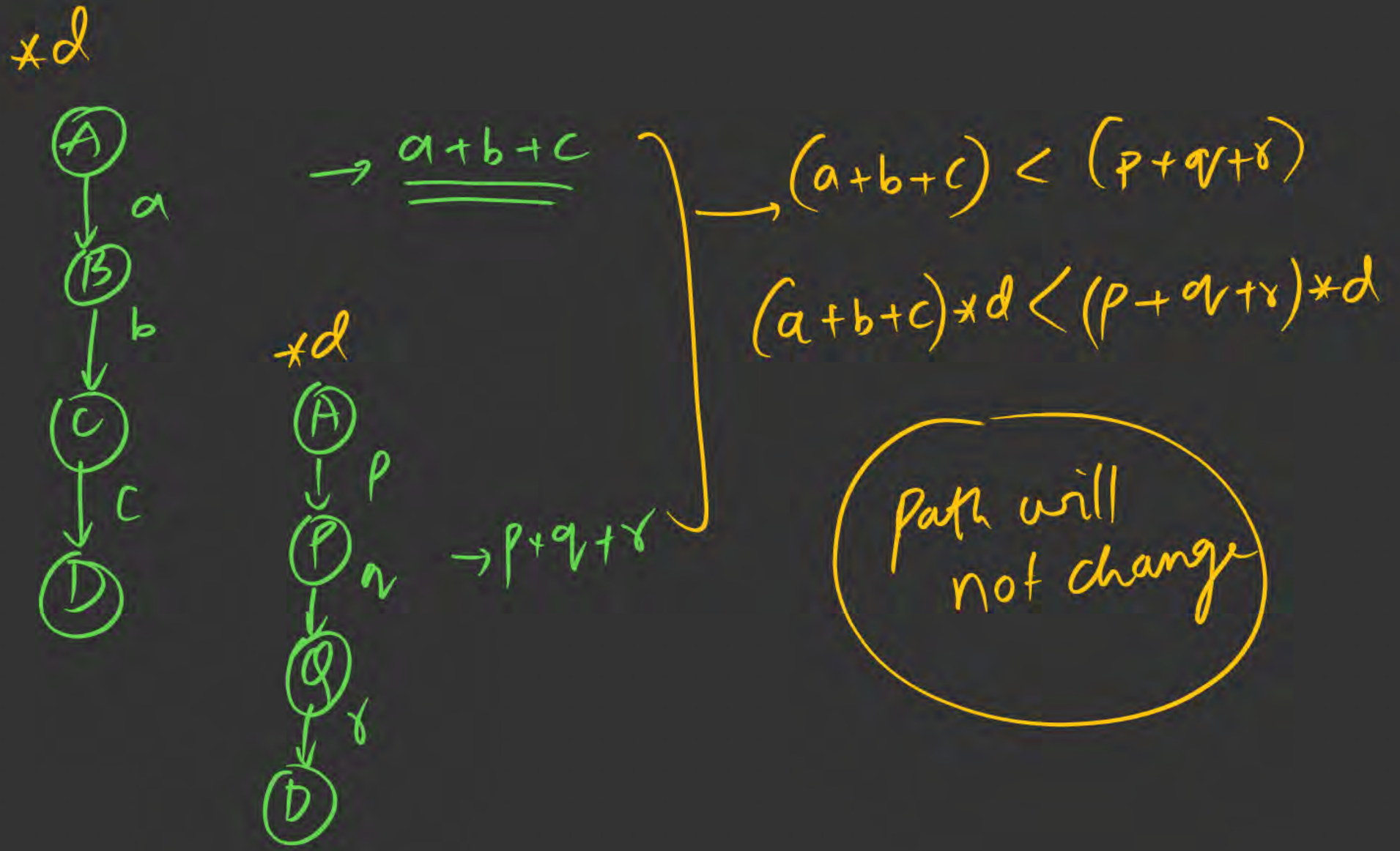
updated graph

Shortest path

SAG
Cost = 16

(q.2)


Graph
UCS
S → G
path
*C (every edge multiplied by some const 'c')
Graph'
UCS
can path change?
No, will not change


*d
A
a
B
b
C
c
D
→ a+b+c
(a+b+c) < (p+q+r)
(a+b+c)*d < (p+q+r)*d
*d
A
p
P
q
Q
r
D
→ p+q+r
Path will not change

Revision : Time and Space Complexity

- **$A^* \Rightarrow O(b^d)$** → We might store all nodes (Visit all nodes)

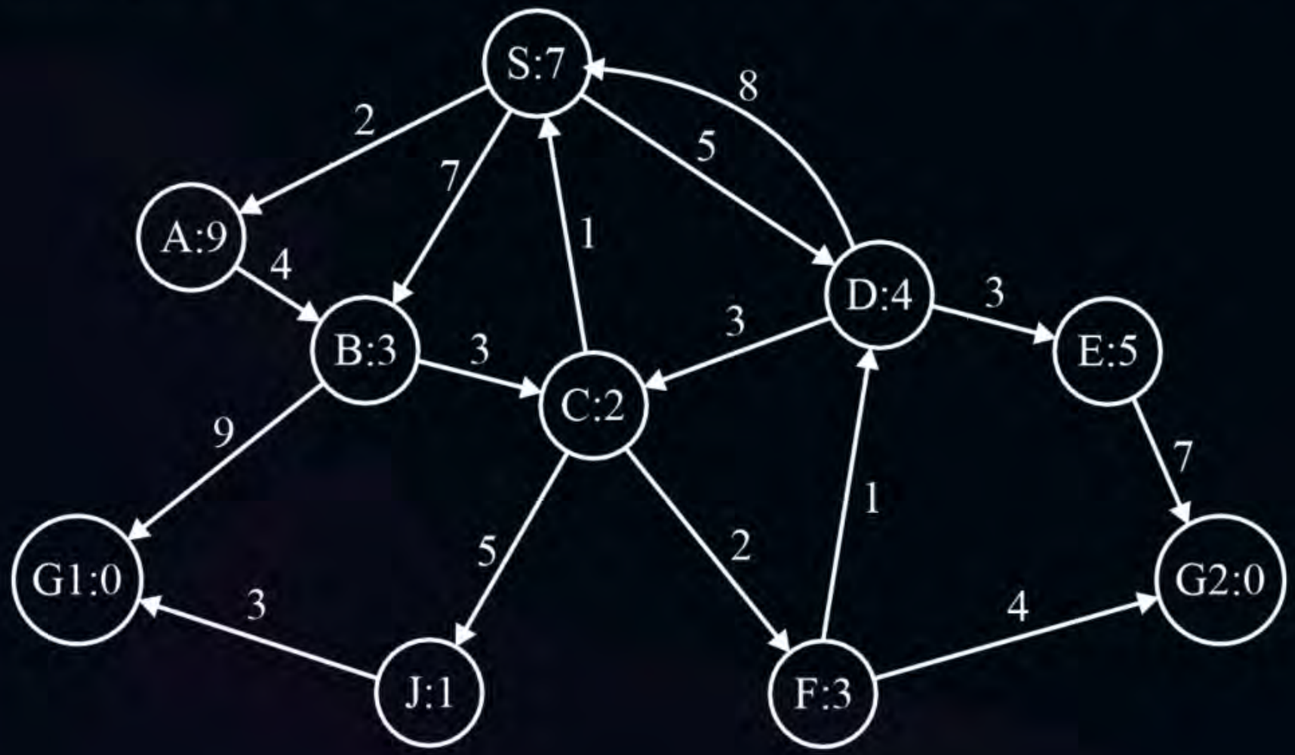
Revision : Heuristic – consistent and admissible

Summary:

	A* tree	A* graph
If C heuristic	✓	✓
If A heuristic	✓	×
If neither C nor A heuristics	×	×

Topic : Artificial Intelligence

#Q. Consider the search space below, where S is the start node and Gl and G2 satisfy the goal test. Arcs are labeled with the cost of traversing them and the heuristic cost to a goal is reported inside nodes (so lower scores are better).



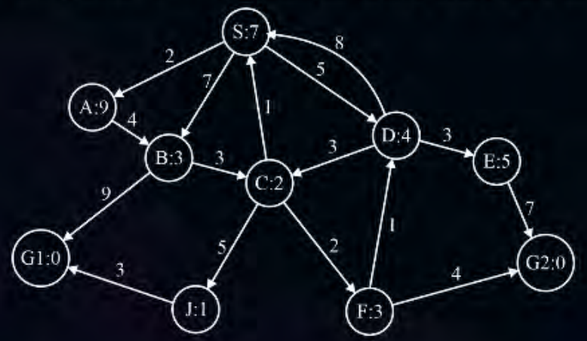
Topic : Artificial Intelligence

For A* search, indicate which goal state is reached at what cost and list, in order, all the states popped off of the OPEN list. You use a search graph to show your work.

Note: When all else is equal, nodes should be removed from OPEN in alphabetical order.

Topic : Artificial Intelligence

#Q. Consider the search space below, where S is the start node and Gl and G2 satisfy the goal test. Arcs are labeled with the cost of traversing them and the heuristic cost to a goal is reported inside nodes (so lower scores are better).



OPEN	CLOSED	S	D	B	C	A	E	F	G2
S	X	11	11	11	(11)	11	11	11	
A	X	10	(10)	10	10	10	10	10	
B	X	(9)	9	9	9	9	9	9	
D	X	x	10	(10)	16	10	10	10	
C	X	x	13	13	13	(13)	13	13	
E	X	x	x	16	16	16	16	16	
G1		x	x	x	14	14	14	14	
J		x	x	x	(13)	13	(13)	13	
F		x	x	x	x	x	15	(14)	
G2		x	x	x	x	x			

G2 → stop

G2 : 14

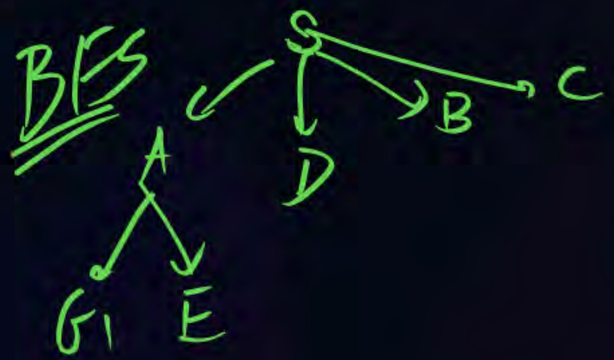
SDCFG2

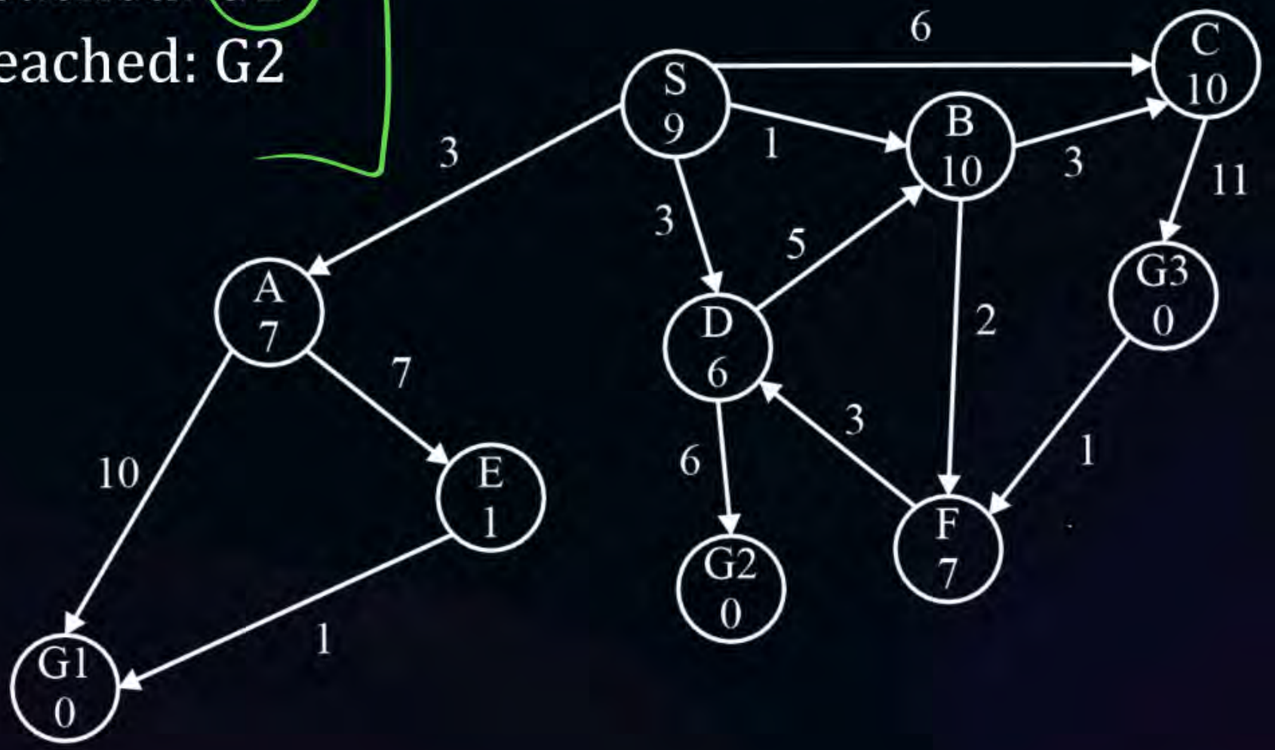
Topic : Artificial Intelligence

#Q. Consider the search graph below, where S is the start node and G1, G2, and G3 are goal states. Arcs are labeled with the cost of traversing them and the heuristic cost to a goal is shown inside the nodes. For each of the three search strategies below, indicate which of the goal states is reached:

Ans

(a) Breadth-first search. Goal reached: G1
(b) Uniform cost search. Goal reached: G2
(c) A* search. Goal reached: G2





Topic : Artificial Intelligence

Advantages

- **Optimality:** A* is guaranteed to find the least-cost path to the goal if the heuristic used is admissible (never overestimates the true cost) and consistent (satisfies the triangle inequality).
- **Efficiency:** A* is generally more efficient than uninformed search algorithms like Dijkstra's algorithm because it uses heuristics to guide the search. This often results in exploring fewer nodes, reducing the overall computation.
- **Flexibility:** A* can be adapted to various problems by changing the heuristic function. This flexibility makes it applicable to a wide range of scenarios, from simple grid navigation to complex graph-based pathfinding.
- **Combines Advantages of Greedy and Uniform-Cost Search:** A* combines the strengths of greedy best-first search (which is fast but not always optimal) and uniform-cost search (which is optimal but can be slow) by balancing exploration and exploitation.

Topic : Artificial Intelligence

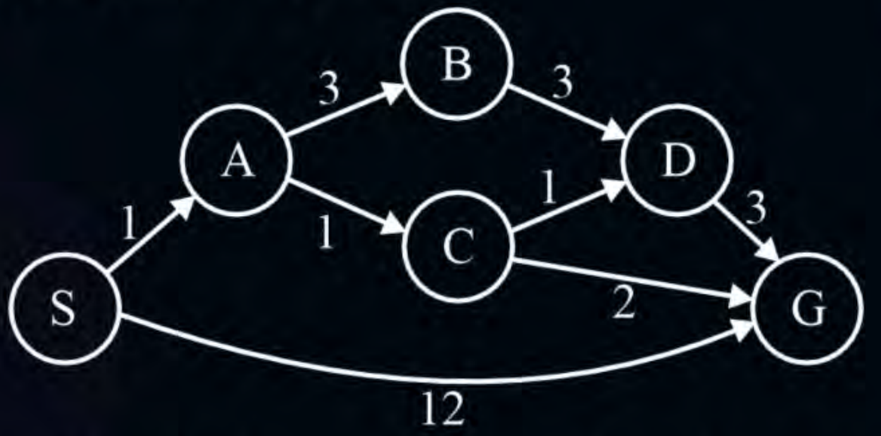
Disadvantages

- High Memory Usage: A* requires storing all generated nodes in memory, which can lead to high memory consumption, especially in large search spaces. This makes it less suitable for problems with very large state spaces.
- Computationally Intensive: In the worst case, A* can be computationally expensive, especially if the branching factor is high or the heuristic is not well-optimized. The algorithm's time complexity is exponential in the worst case.
- Heuristic Dependency: The performance of A* heavily depends on the quality of the heuristic. If the heuristic is poorly chosen, the algorithm can degrade to a less efficient search, potentially exploring many unnecessary nodes.
- Not Always Practical for Real-Time Applications: Due to its potential high time and space complexity, A* may not be suitable for real-time applications where quick decisions are necessary.

Topic : Artificial Intelligence

#Q. Answer the following questions about the search problem shown above. Break any ties alphabetically. For the questions that ask for a path, please give your answers in the form 'S – A – D – G'.

HW



(a) What path would breadth-first graph search return for this search problem?
(b) What path would uniform cost graph search return for this search problem?
(c) What path would depth-first graph search return for this search problem?
(d) What path would A* graph search, using a consistent heuristic, return for this search problem?

Topic : Artificial Intelligence

#Q. Consider the heuristics for this problem shown in the table below.

(i) Is h_1 admissible? Yes **No**
(ii) Is h_1 consistent? Yes **No**
(iii) Is h_2 admissible? **Yes** No
(iv) Is h_2 consistent? Yes **No**

State	h_1	h_2
S	5	4
A	3	2
BH	6	6
C	2	1
D	3	3
G	0	0

Topic : Artificial Intelligence

#Q. Let h_1 and h_2 be two admissible heuristics used in A* search.
Which ONE of the following expressions is always an admissible heuristic?

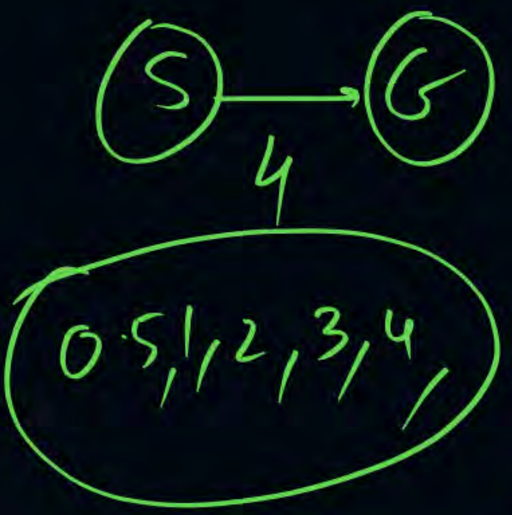
A $h_1 + h_2$

B $h_1 \times h_2$

C $h_1 / h_2, (h_2 \neq 0)$

D $|h_1 - h_2|$

D



35%

$\frac{4}{0.5} = 8$

4/1/2 → 9

Topic : Artificial Intelligence

#Q. If h1 and h2 are two admissible heuristics, then which of the following are guaranteed to be admissible?

A min(h1, h2)

B max(h1, h2)

C (h1 + h2) / 2

D (h1 . h2)^0.5

$h \leq h^*$

$h \leq 15$

$h = 1, 2 \ldots 14, 15$

$\frac{14+15}{2} \leq 15$

$\sqrt{14 \times 15} \leq 15$

Topic : Artificial Intelligence

Weighted A* search Algorithm:

- Weighted A* is a variant of the classic A* algorithm used for faster pathfinding by introducing a weight factor to the heuristic. It trades optimality for speed, making it useful in time-critical applications.
- A* guarantees optimal path (if heuristic is admissible and consistent) but can be slow. Weighted A* speeds up the search by being more aggressive in exploring nodes closer to the goal, even if it sacrifices optimality.

1) GBFS → fast (f = h)

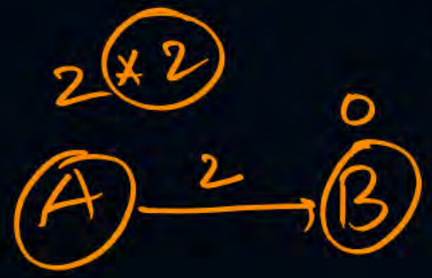
2) A^* → slow (f = h + g) ⟹

3) weighted A^* → fast ⟹ f = g + w * h(n)

but may not be optimal

w: weight

Topic : Artificial Intelligence



admissible

graph
h()
A*

$h \leq h^*()$

Simple A* search
↳ optimal Soln.

wt = 2

graph
h' = (2*h)
A*

h' = no more admissible
So, no more optimal

Same as applying weighted A* on original graph with weight = 2

Topic : Artificial Intelligence

Weighted A* search Algorithm:

- Classic A* uses:

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

- Weighted A* modifies this to:

$$f(n) = g(n) + w \cdot h(n)$$

Where:

- $g(n)$ = cost from start to current node n
- $h(n)$ = heuristic estimate from n to goal
- $w \geq 1$ = weight factor

Topic : Artificial Intelligence

Weighted A* search Algorithm:

- If w = 1 : behaves like classic A*, optimal and complete
- If w > 1 : heuristic is over-emphasized, search is faster but not guaranteed to be optimal
- Higher the w the more greedy the algorithm becomes (closer to Greedy Best-First Search)

$f = g + 1 \cdot h = g + h$

Topic : Artificial Intelligence

Weighted A* search Algorithm:

Summary

Algorithm	Modification	Notes
A*	$f(n) = g(n) + h(n)$	Optimal
Weight A*	$f(n) = g(n) + w \cdot h(n)$	Faster, suboptimal
Greedy BFS	$f(n) = h(n)$	Fastest, no optimality
Uniform Cost	$f(n) = g(n)$	Ignores heuristic, but optimal

THANK - YOU