DS & AI ENGINEERING

Artificial Intelligence

Informed search

Lecture No.- 07

By- Aditya sir

Recap of Previous Lecture

Topic

Questions

Topics to be Covered

Topic

Topic

Hill climbing

Practice Questions

①

Have to do → 95%

②

like/ want to do

①

love what yo do

Do what you love

4 - 7 yrs/old

Topic: Beam Search

GBFS

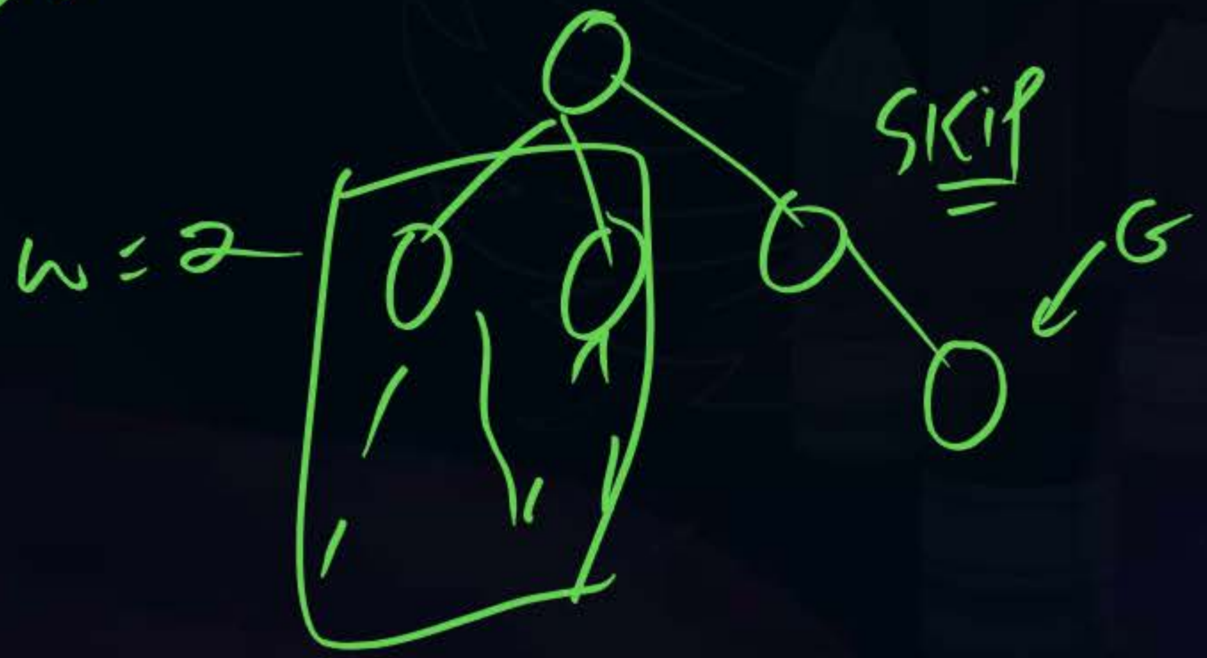
Beam Search ⇒Complete × Optimal

⇓ ✗ ✗

Concept of Beam width

- Whenever any node is Expanded, then we do not bring all nodes into the open list
- We define a Beam width (W) and then whenever any node is visited, best "W" neighbours (neighbours with min f value) are brought to open list.

  In GBFS ⇒ f =Heuristic Value



Topic: Beam Search

So, Steps in Beam Search ⇒ (Beam Width "w")

1. Start node visit.

   Find f = h value for all neighbours.

2. Bring all the best "w" neighbours in open list.
3. Visit node in open list with min f value.
4. Keep repeating step 1, 2, 3.

   (neither Complete nor ~~Optimum~~ Optimal).

5th class

cricket match

[Fear & Excitement are same emotions]

-ve

+ve

Topic: Hill Climbing Search

GBFS

V.V.imp

- Use an evaluation function $f(n)=h(n)$, but the maximum size of the nodes list is w, a fixed constant.
- Only keeps w best nodes as candidates for expansion and throws the rest away. More space-efficient than greedy search but may throw away a node that is on a solution path.

w = 1 ⇒ Hill climbing search

w = define ⇒ Beam search

Here f value of each node ⇒ $h(n)$

w = ∞ ⇒ GBFS

- Space complexity ⇒ $O(b^d)$
- Time complexity ⇒ $O(b^d)$

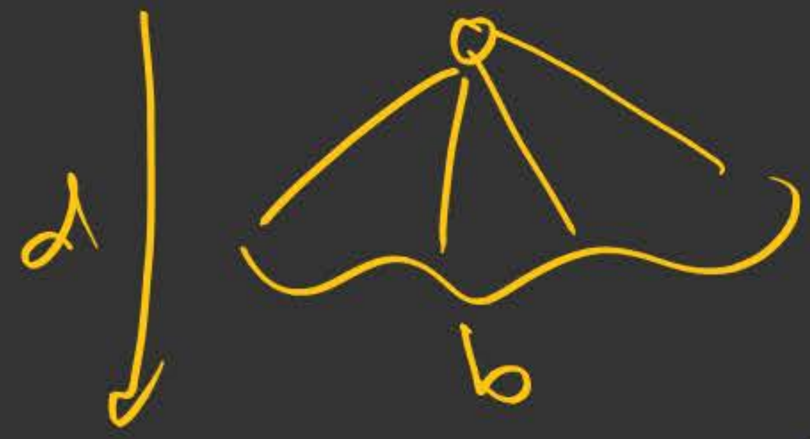
- No beam width defined, we observe all the neighbours.

- If we define 'w' then
- Space complexity = $O(w^d)$
- Time complexity = $O(w^d)$

GBFS

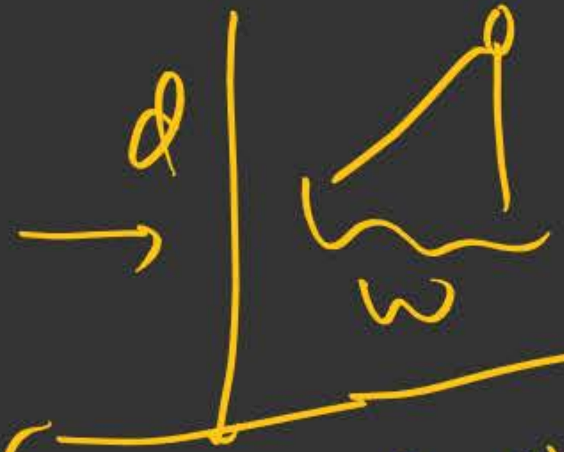
TC : $O(b^d)$

SC : $O(b^d)$



$w \leq b$

Beam Search : $\underline{\underline{w}}$ →



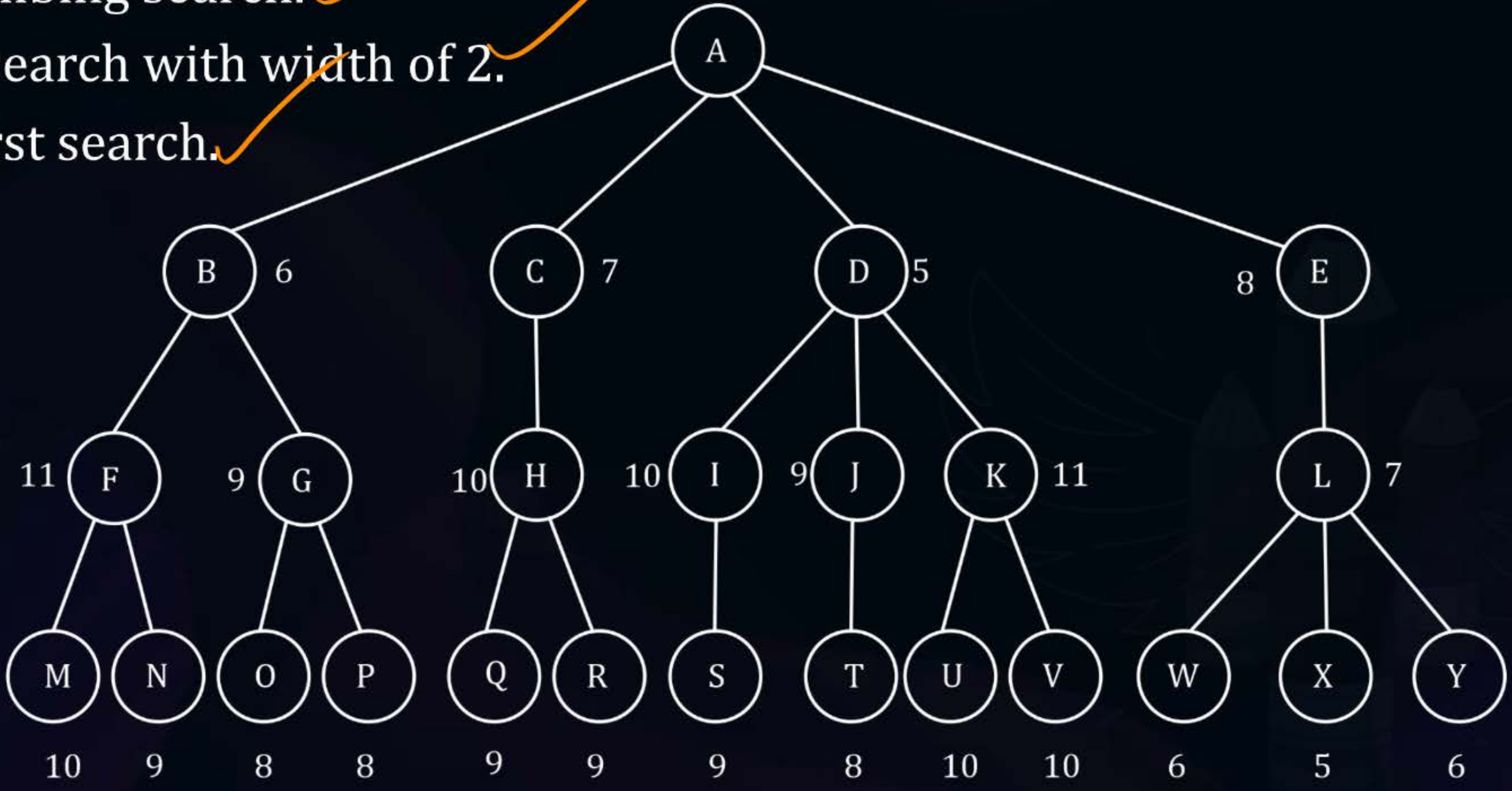
$$\boxed{\begin{array}{l} TC : O(w^d) \\ SC : O(\underline{w^d}) \end{array}}$$

Topic: Hill Climbing Search

Draw the search tree and write the order of expansion of the nodes when applying:

- Hill climbing search.
- Beam search with width of 2.
- Best first search.

GBFS



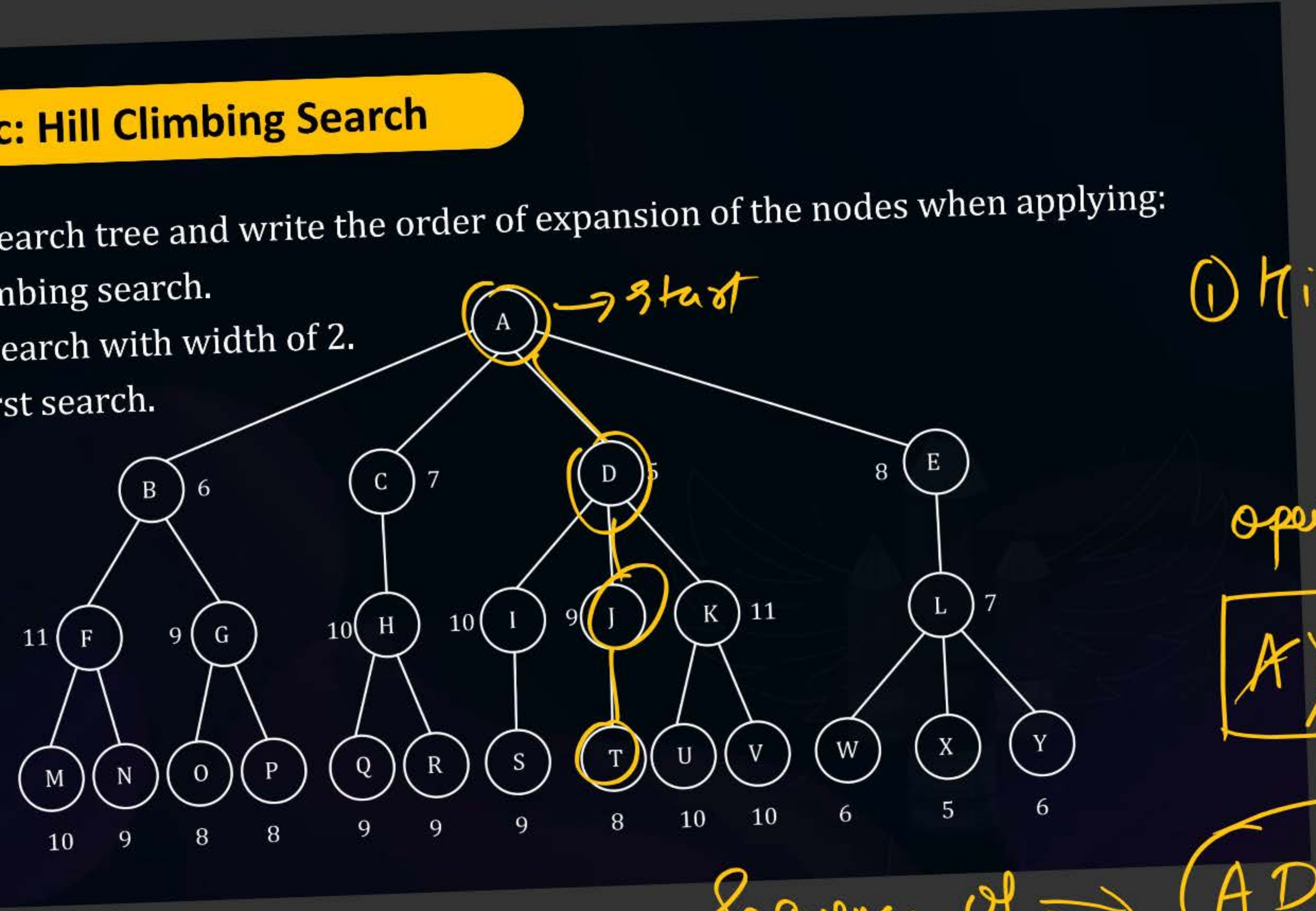
c: Hill Climbing Search

earch tree and write the order of expansion of the nodes when applying:

nbing search.

earch with width of 2.

rst search.



① Hill climbing. w = 1

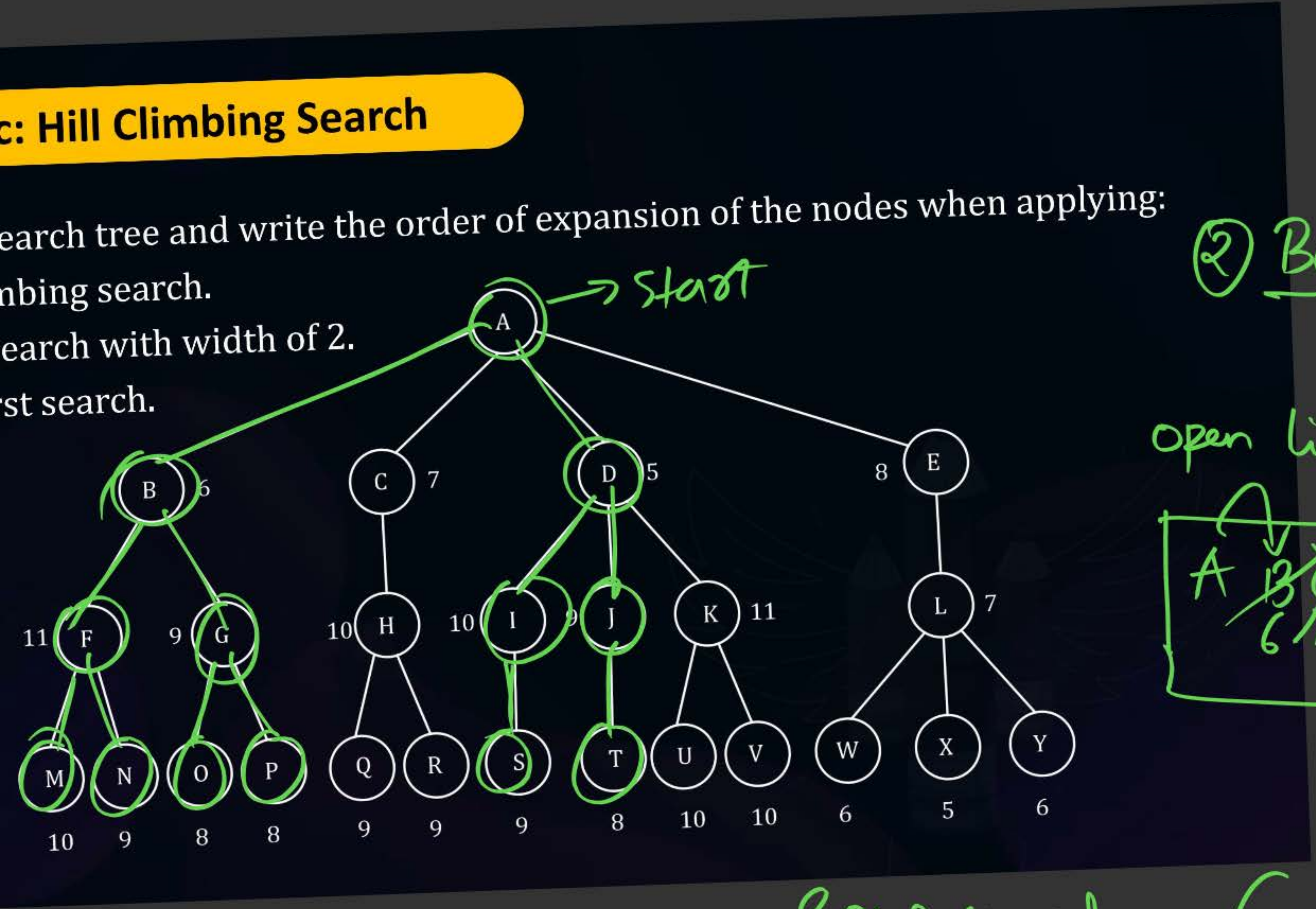
open list

A ~~B~~ ~~C~~ D ~~E~~ ~~I~~ J ~~K~~ ~~T~~

6 7 5 8 10 9 11 8

Stop

Sequence of node visited ⇒ A D J T



c: Hill Climbing Search

earch tree and write the order of expansion of the nodes when applying:

nbing search.

earch with width of 2.

rst search.

② Beam Search, W=2

open list

A ~~B~~ ~~C~~ ~~D~~ ~~E~~ ~~I~~ ~~J~~ ~~K~~ ~~F~~ ~~G~~ ~~O~~ ~~P~~ ~~T~~ ~~S~~

6 7 5 8 10 9 11 11 9 8 8 8 9

~~M N~~

~~10 9~~

Sequence of visited nodes : (A D B G O P J T I S F N M)

Stop

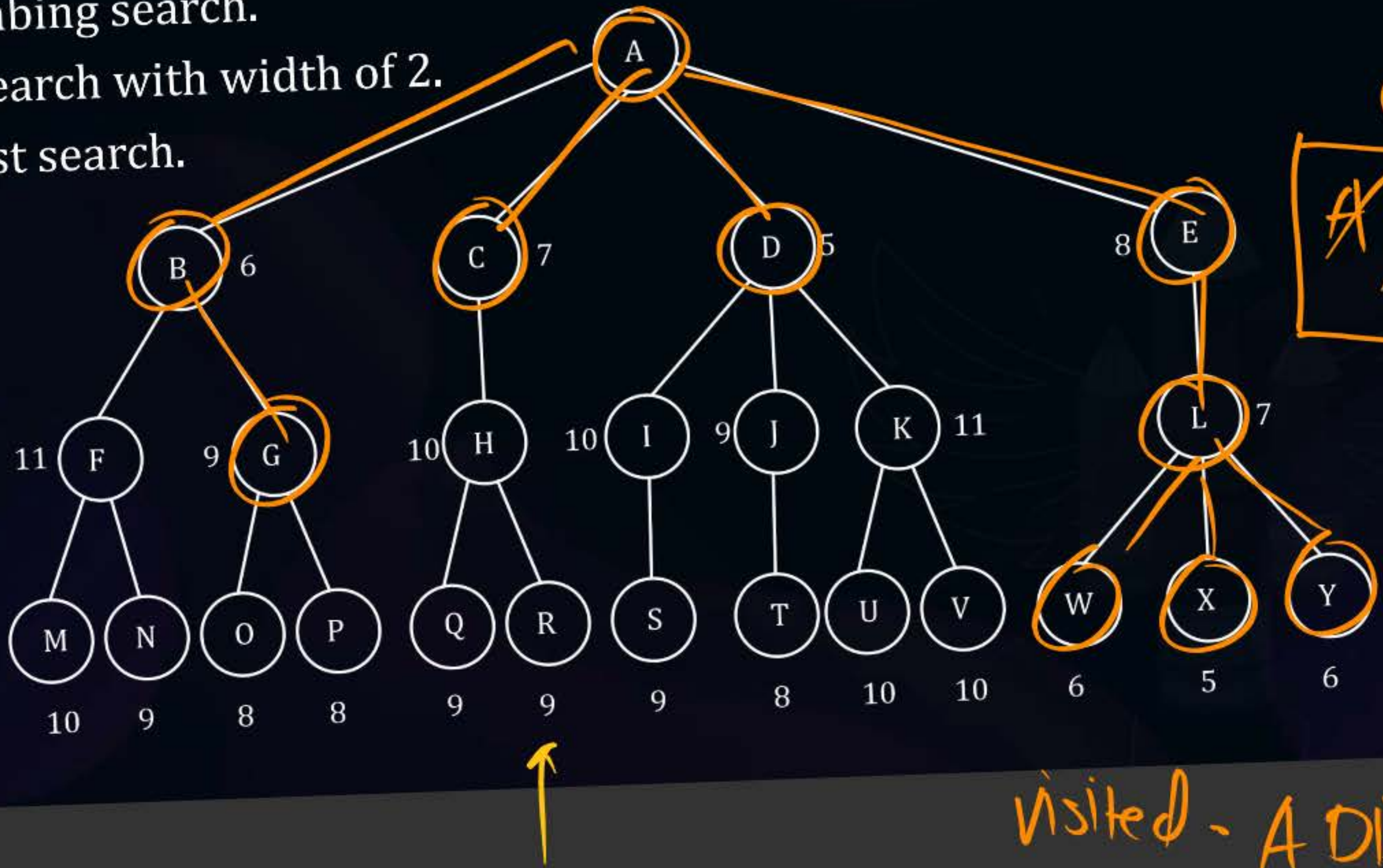
c: Hill Climbing Search

earch tree and write the order of expansion of the nodes when applying:

nbing search.

earch with width of 2.

rst search.



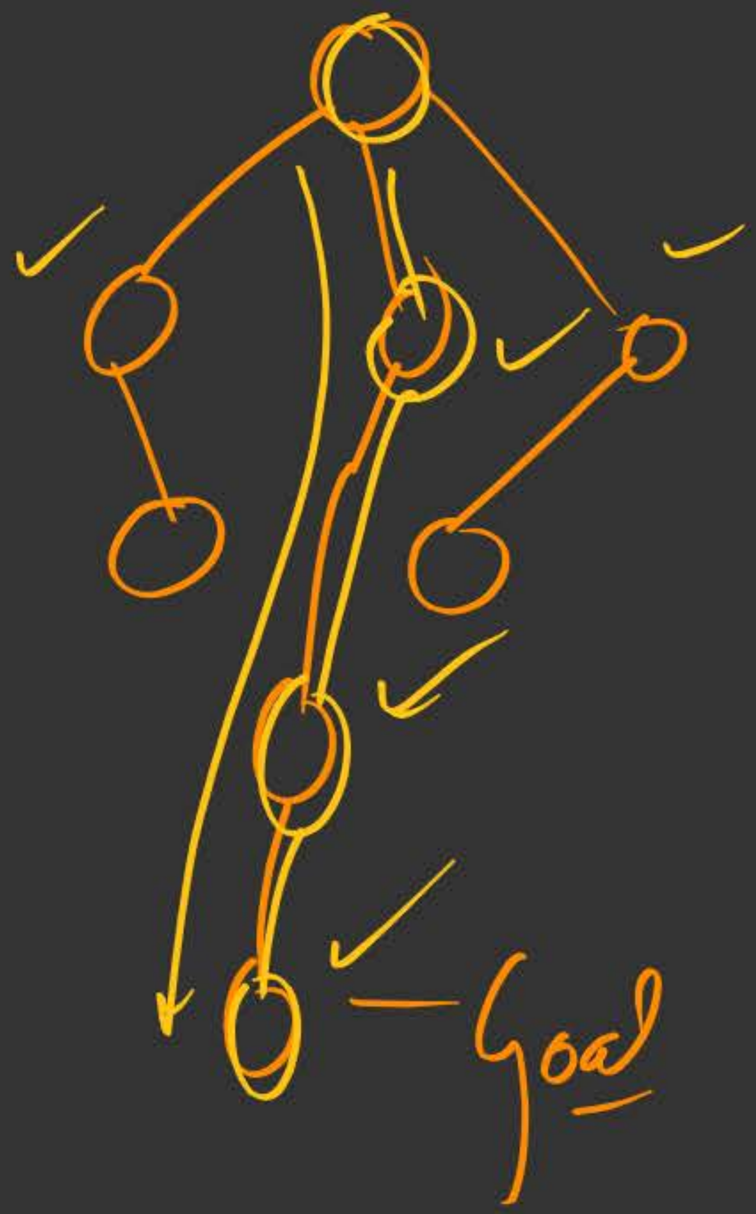
③ GBFS

Open list

~~A~~ ~~B~~ ~~C~~ ~~D~~ ~~E~~ I J K F ~~G~~ H ~~L~~ ~~W~~ ~~X~~ ~~Y~~ O P

6 7 5 8 10 9 11 11 9 10 7 6 5 6 8 8 . . . -

visited Seq: A D B C E L X W Y G


Goal

Topic: Hill Climbing Search

Optimal X , Complete X

Hill climbing beam search:

- because work on ~~at~~ heuristics only

because it delete on throw away some of nodes, because of this detection it may throw nodes that lead to goal.

$\Downarrow$

$\boxed{f=h}$

Not complete:

because does not search whole graph

Topic: Hill Climbing Search

GBFS

GBFS

- If we have a Graph Search i.e. we create the visited node list i.e. once a node is visited it will not be visited again.

complete ✓ (slow)

BFS: w=∞

- If we have a ~~trace~~ tree search no visited list

BFS: w=∞

Not Complete

[MCQ]

#Q. We will investigate various search algorithms for the following graph. Edges are labeled with their costs, and heuristic values h for states are labeled next to the states. S is the start state, and G is the goal state. In all search algorithms, assume ties are broken in alphabetical order.
What path is returned by greedy graph search?

A S → A → C → D → G

[✓ B] S → A → C → G

C S → A → C → D → G

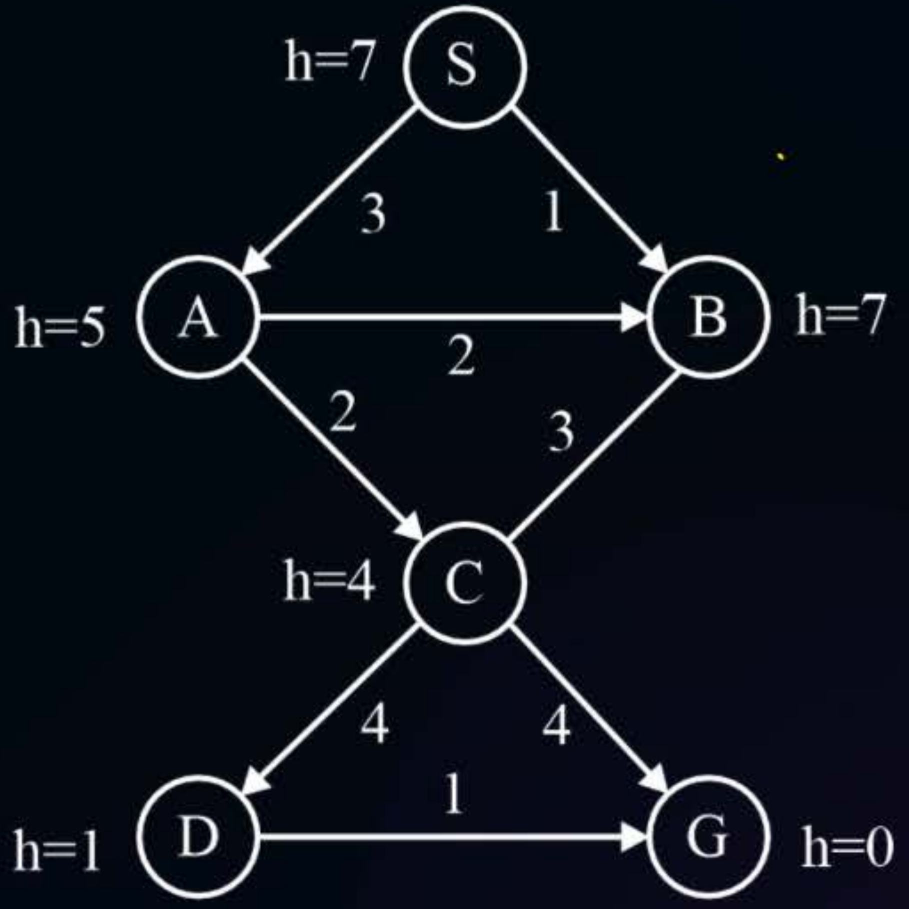
D None of the above

open

~~S~~ ~~A~~ B ~~C~~ D ~~G~~
5 7 4 1 0

closed

SACG → stop



[MCQ]

#Q. Consider the following graph. For the following sub-questions, ties are broken in alphabetical order. from $\boxed{S \xrightarrow{to} G}$
Order of node visit is:

① DFS

A SABDCG

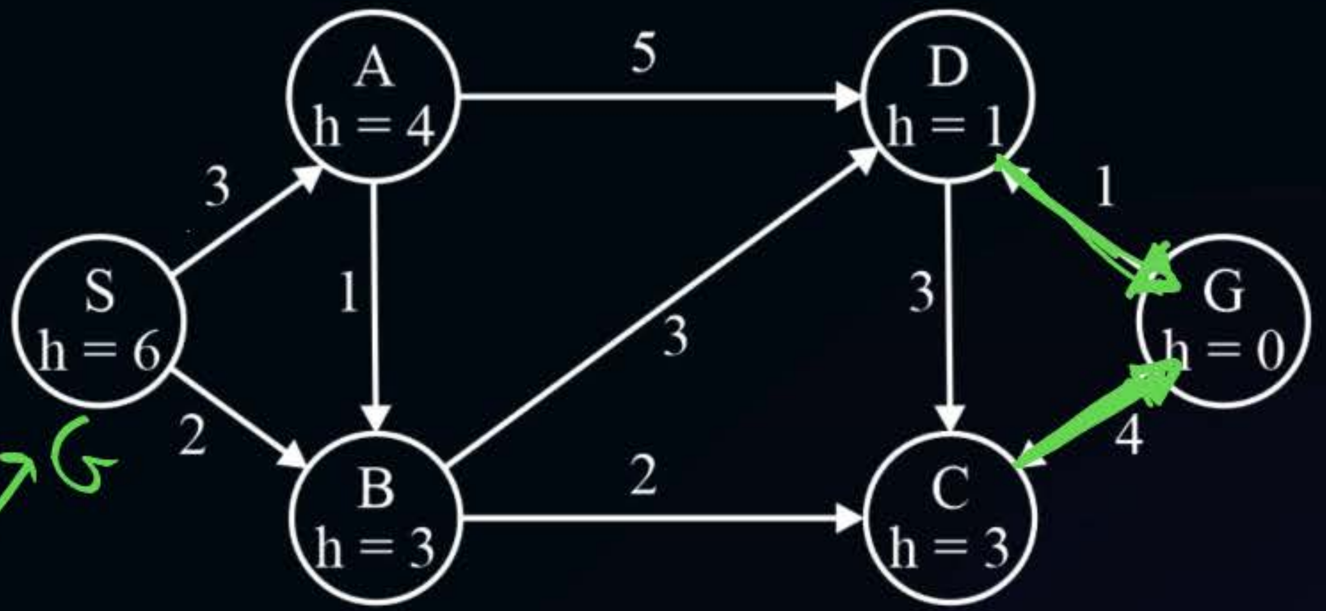
B SABCDG

C SADG

D ✓ None of the above

$$\begin{tikzcd}
S \arrow[r] & A \arrow[d] & & \\
& B \arrow[r] & C \arrow[r] & G
\end{tikzcd}$$

SABCG



[NAT]

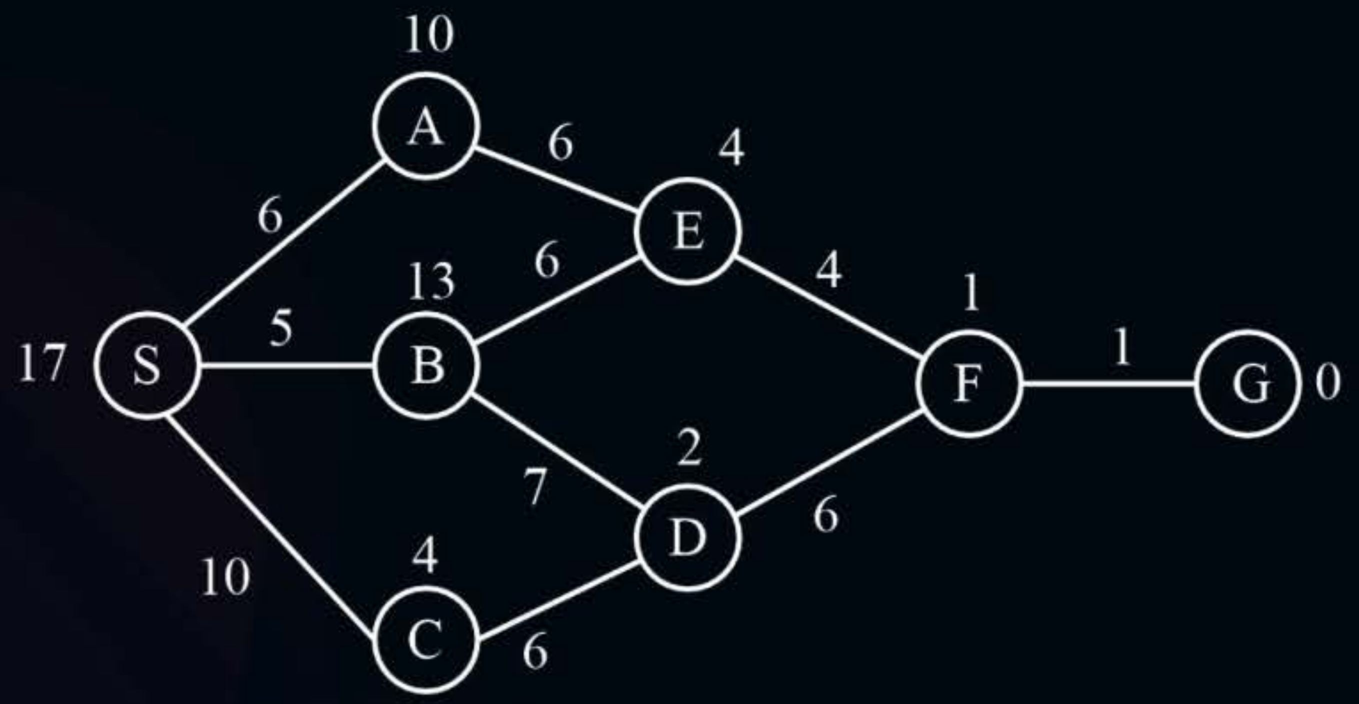
graph.

#Q. Perform the A* algorithm on the following figure. Explicitly write down the queue at each step.

$S \rightarrow G$

Cost = ?

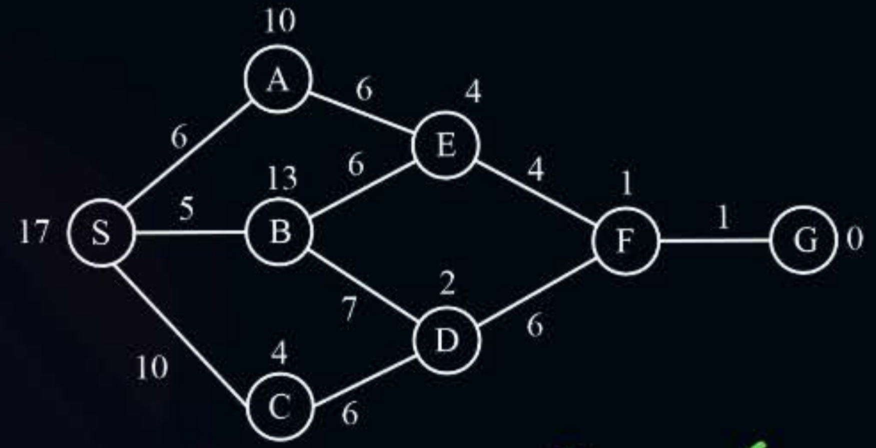
58%



[NAT]

graph

#Q. Perform the A* algorithm on the following figure. Explicitly write down the queue at each step.



S → G

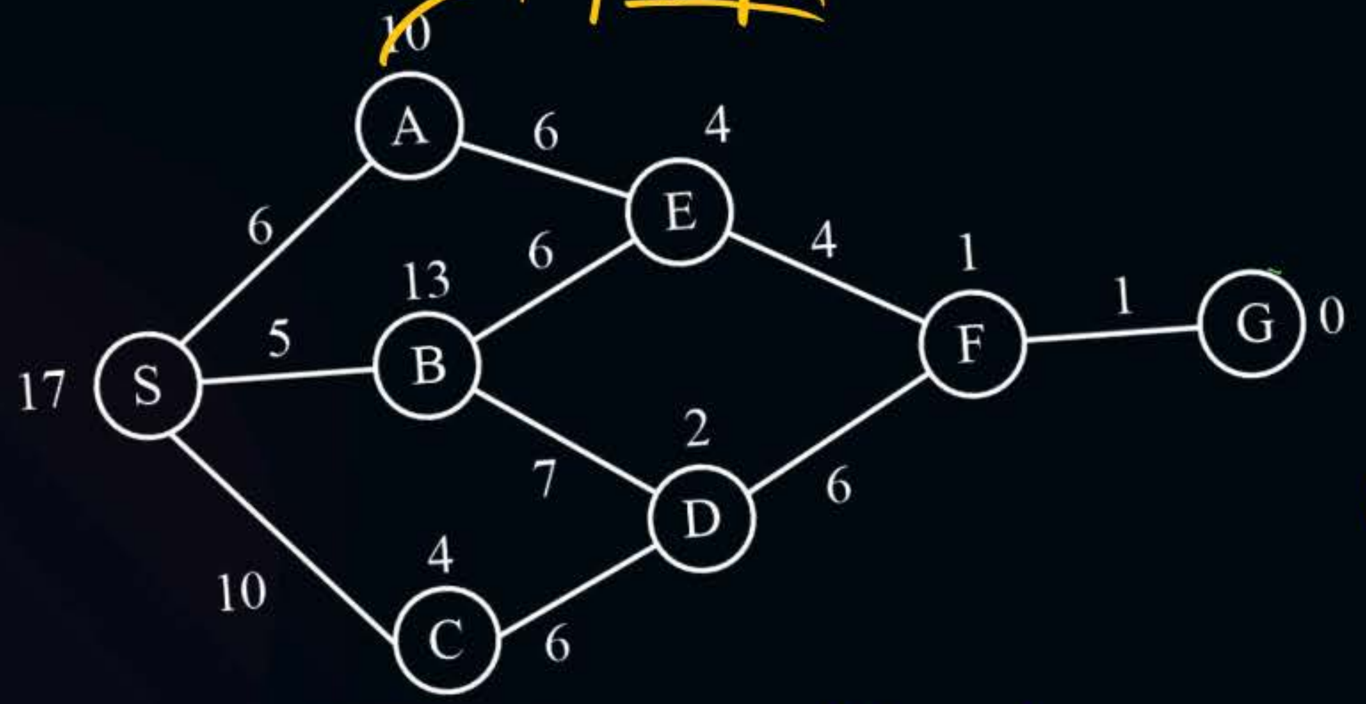
Cost = 17

path $\boxed{SAEFG}$ ✓

open	closed					
S×	S	C	A	E	F	G
A×	16	16	16	16	16	
B ·	18	18	18	18	18	
C×	14	14	14	14	14	
D ·	×	18	18	18	18	
E×	×	×	16	16	16	
F×	×	×	×	17	17	
G×	×	×	×	×	17	

stop

graph



S → G

cost = ?
6

Apply UCS:

path

SBEFG

cost = 16

90%

Open	Closed							
	S	B	A	C	E	D	F	G
S×								
A×	6	6	6	6	6	6	6	
B×	5	5	5	5	5	5	5	
C×	10	10	10	10	10	10	10	
E×	×	11	11	11	11	11	11	
D×	×	12	12	12	12	12	12	
F×	×	×	×	×	15	15	15	
G	×	×	×	×	×	×	16	

Stop

THANK - YOU