

SEARCH METHODS IN AI

HEURISTIC SEARCH METHODS



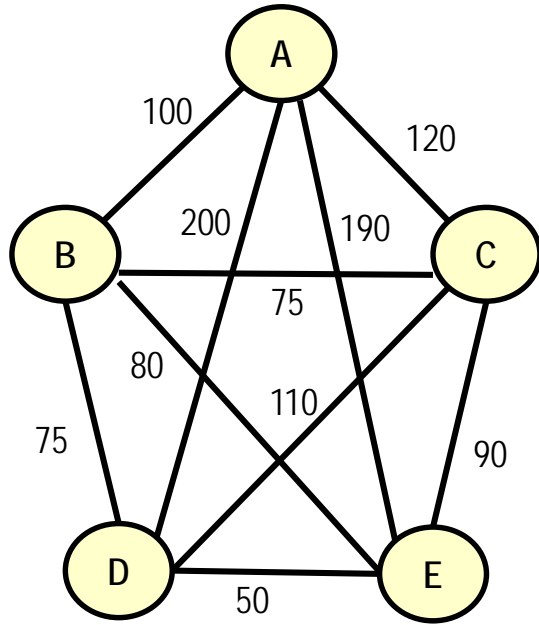
Arijit Mondal & Partha P Chakrabarti

Indian Institute of Technology Kharagpur

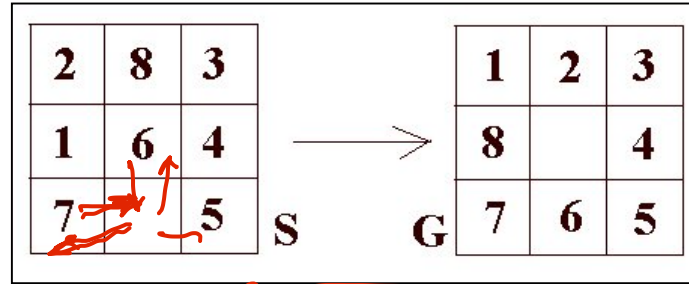
BASICS OF STATE SPACE MODELLING

- **STATE or CONFIGURATION:**
 - A set of variables which define a state or configuration
 - Domains for every variable and constraints among variables to define a valid configuration
- **STATE TRANSFORMATION RULES or MOVES:**
 - A set of RULES which define which are the valid set of NEXT STATE of a given State
 - It also indicates who can make these Moves (OR Nodes, AND nodes, etc)
- **STATE SPACE or IMPLICIT GRAPH**
 - The Complete Graph produced out of the State Transformation Rules.
 - Typically too large to store. Could be Infinite.
- **INITIAL or START STATE(s), GOAL STATE(s)**
- **SOLUTION(s), COSTS**
 - Depending on the problem formulation, it can be a PATH from Start to Goal or a Sub-graph of And-ed Nodes
- **SEARCH ALGORITHMS**
 - Intelligently explore the Implicit Graph or State Space by examining only a small sub-set to find the solution
 - To use Domain Knowledge or HEURISTICS to try and reach Goals faster

EXAMPLE PROBLEMS



Travelling Salesperson



Sliding Puzzle



Knapsack Packing

shortest path

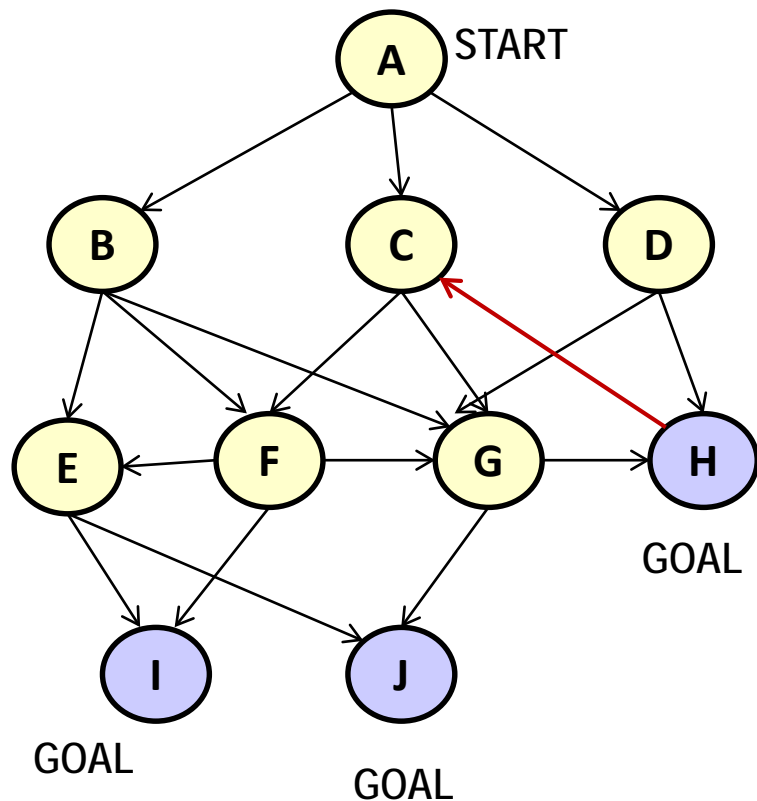
which subset
should I
take so that
we get maximum
value within
the weight limits
MAXIMIZATION

SEARCHING IMPLICIT GRAPHS

The various Search Algorithms include


- BASIC Algorithms: Depth-First (DFS), Breadth-first (BFS), Iterative Deepening (IDS)
- COST-based Algorithms: Depth-First Branch-and-Bound, Best First Search, Best-First Iterative Deepening
- Widely Used Algorithms: A* and IDA* (Or Graphs), AO* (And/Or Graphs), Alpha-beta Pruning (Game-Trees)

EXAMPLE: SEARCHING A STATE SPACE GRAPH



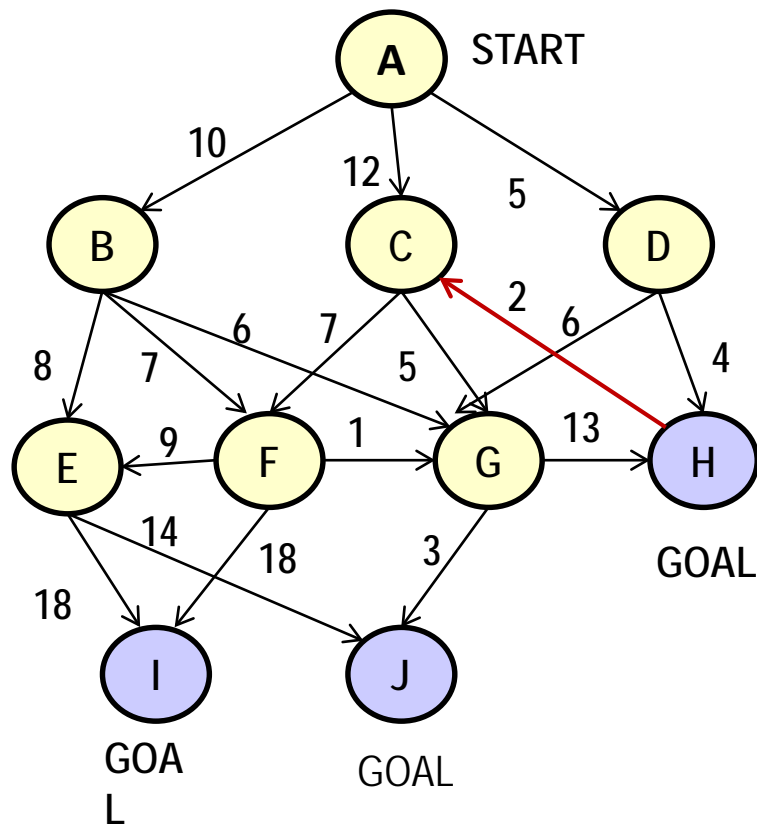
- DEPTH-FIRST SEARCH (DFS)
- BREADTH-FIRST SEARCH (BFS)
- ITERATIVE DEEPENDING SEARCH (IDS)
- PROPERTIES
 - SOLUTION GUARANTEES
 - MEMORY REQUIREMENTS

BASIC ALGORITHMS: DFS, IDS, BFS

1. [Initialize] Initially the OPEN List contains the Start Node s . CLOSED List is Empty.
2. [Select] Select the first Node n on the OPEN List. If OPEN is empty, Terminate
3. [Goal Test] If n is Goal, then decide on Termination or Continuation / Cost Updation
4. [Expand] 
 - a) Generate the successors n_1, n_2, \dots, n_k , of node n , based on the State Transformation Rules
 - b) Put n in LIST CLOSED ✓
 - c) For each n_i , not already in OPEN or CLOSED List, put n_i in the stack FRONT (for DFS) / END (for BFS) of OPEN List Queue
 - d) For each n_i already in OPEN or CLOSED decide based on cost of the paths
5. [Continue] Go to Step 2

Algorithm IDS Performs DFS Level by Level Iteratively (DFS (1), DFS (2), and so on)

SEARCHING STATE SPACE GRAPHS WITH EDGE COSTS



- COST ORDERED SEARCH:

- DFBB

- Best First Search, *→ ordered by $g(n)$ values*

- Best First IDS

- Use of HEURISTIC Estimates:
Algorithm A* (Or Graphs), AO*
(And/Or Graphs)

- PROPERTIES

- SOLUTION GUARANTEES

- MEMORY REQUIREMENTS

$g(n)$ = cost/current cost of the node from start

HEURISTIC SEARCH

❑ STATE or CONFIGURATION: ✓

- A set of variables which define a state or configuration
- Domains for every variable and constraints among variables to define a valid configuration

❑ STATE TRANSFORMATION RULES or MOVES: ✓

- A set of RULES which define which are the valid set of NEXT STATE of a given State
- It also indicates who can make these Moves (OR Nodes, AND nodes, etc)

❑ STATE SPACE or IMPLICIT GRAPH ✓

- The Complete Graph produced out of the State Transformation Rules.
- Typically too large to store. Could be Infinite.

❑ INITIAL or START STATE(s), GOAL STATE(s)

❑ SOLUTION(s), COSTS ✓

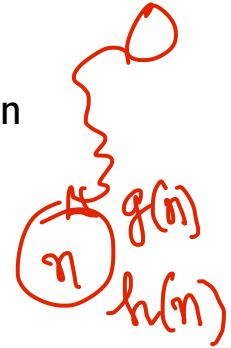
- Depending on the problem formulation, it can be a PATH from Start to Goal or a Sub-graph of And-ed Nodes

❑ HEURISTICS

- Estimates of cost from a given state to goal. This, along with the current cost of the path from start till now is used to guide the search

❑ HEURISTIC SEARCH ALGORITHMS

- Algorithm A*, Depth-First Branch & Bound, IDA*, AO*, Alpha-Beta, etc
- Knowledge vs Search ←



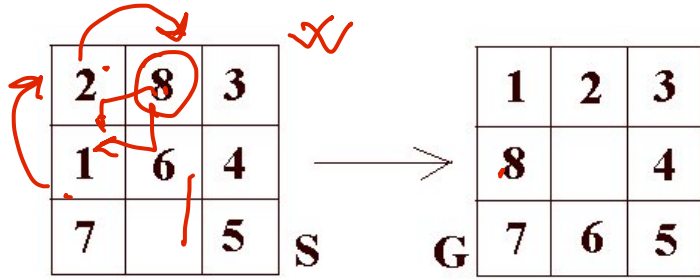
$h(n)$ = estimated cost of the best path from n to goal

heuristic estimate

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

↑

EXAMPLE OF HEURISTICS: 8 PUZZLE PROBLEM



HEURISTIC 1: NUMBER OF MISPLACED TILES

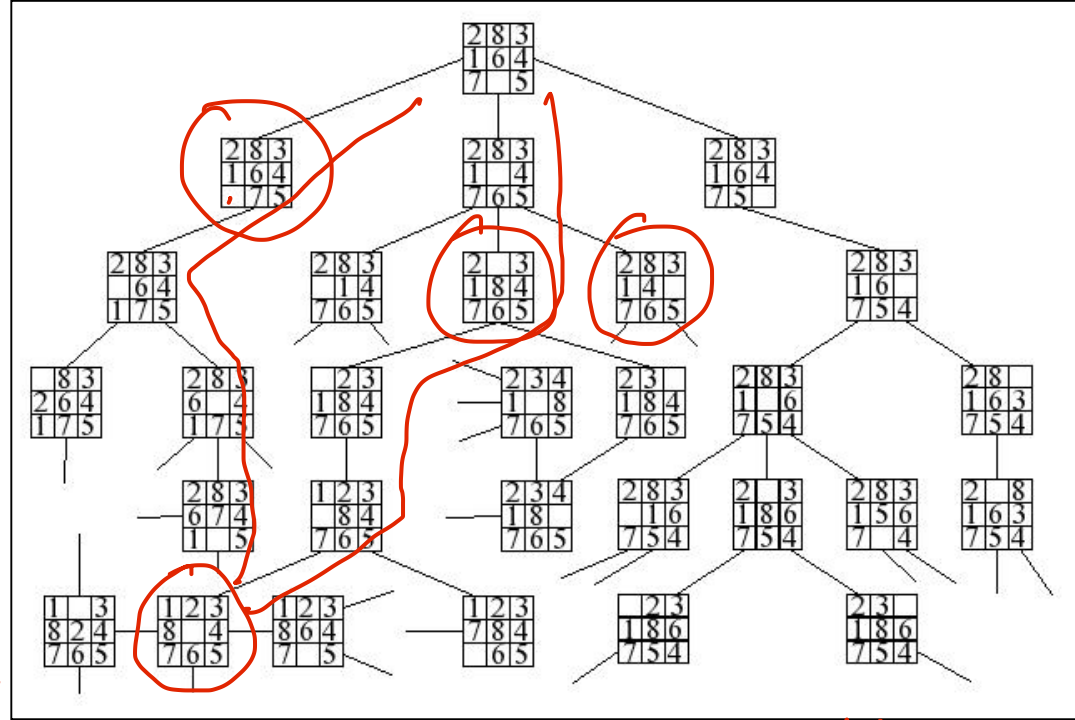
HEURISTIC 2: SUM OF TILE DISTANCES FROM GOAL

$h(n)$ = estimated cost of the best path from n to goal.

4 misplaced tiles. lower bound

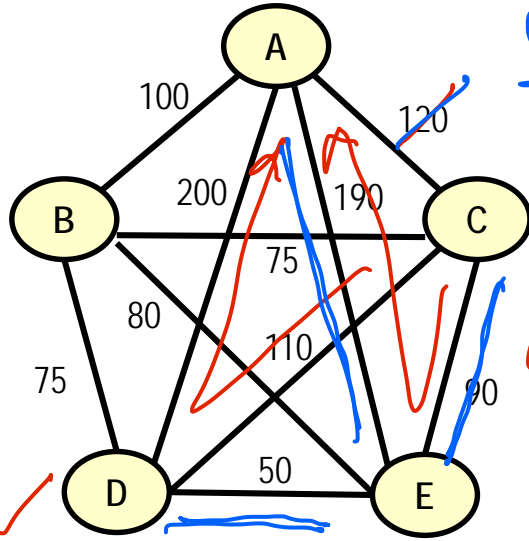
$$1 + 1 + 0 + 0 + 0 + 1 + 0 + 2 = 5$$

Both are underestimates
 h_2 is more accurate than h_1



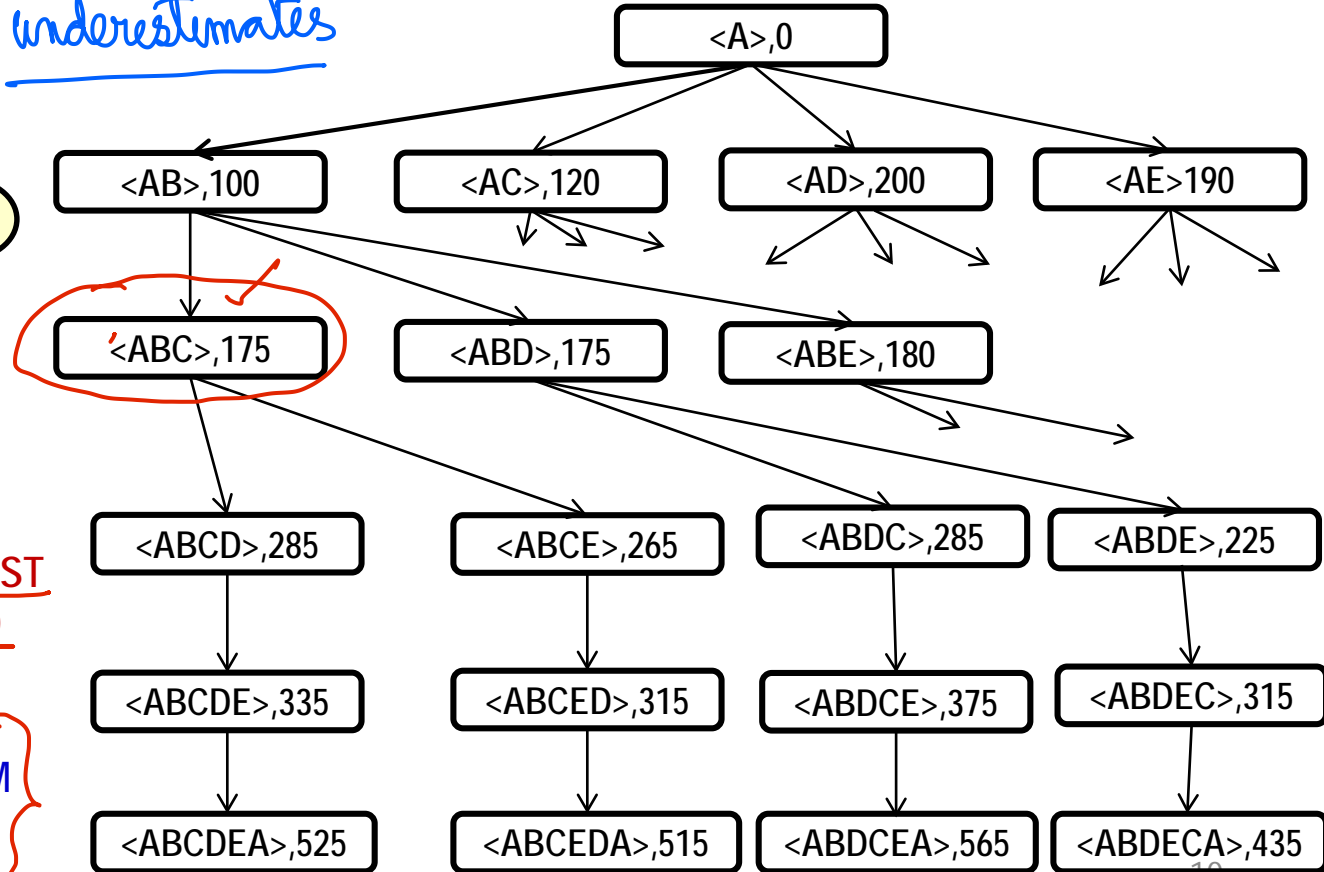
TRAVELLING SALESPERSON PROBLEM

underestimates

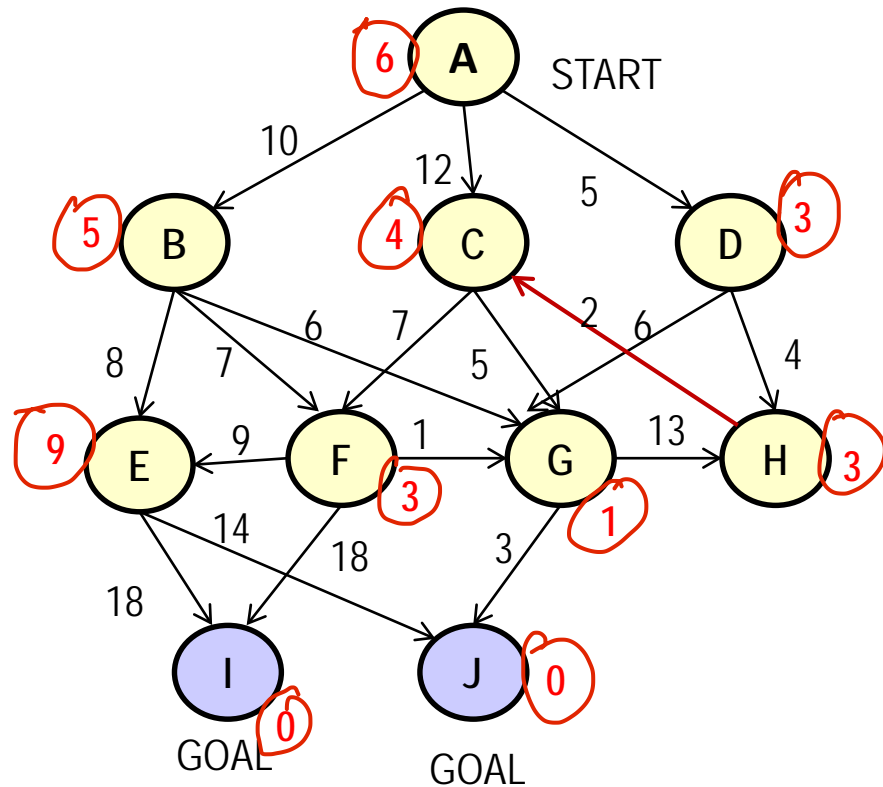


HEURISTIC 1: COST OF SHORTEST PATH FROM CURRENT NODE TO START THROUGH REMAINING NODES ONLY

HEURISTIC 2: COST OF MINIMUM COST SPANNING TREE OF REMAINING NODES



SEARCHING STATE SPACES WITH EDGE COSTS, HEURISTIC ESTIMATES



- HEURISTIC SEARCH ALGORITHMS:
 - DFBB
 - A*: Best First Search,
 - IDA*: Iterative Deepening A*
 - Every edge (n, m) in the graph has a cost $c(n, m) > 0$.
 - **HEURISTIC Estimates:** $h(n) \geq 0$ at every node is the estimated cost of the minimum cost path from node n to goal
- **PROPERTIES**
 - SOLUTION GUARANTEES
 - MEMORY REQUIREMENTS

ALGORITHM A* (BEST FIRST SEARCH IN OR GRAPHS)

Each Node n in the algorithm has a cost $g(n)$ and a heuristic estimate $h(n)$ $f(n) = g(n) + h(n)$.

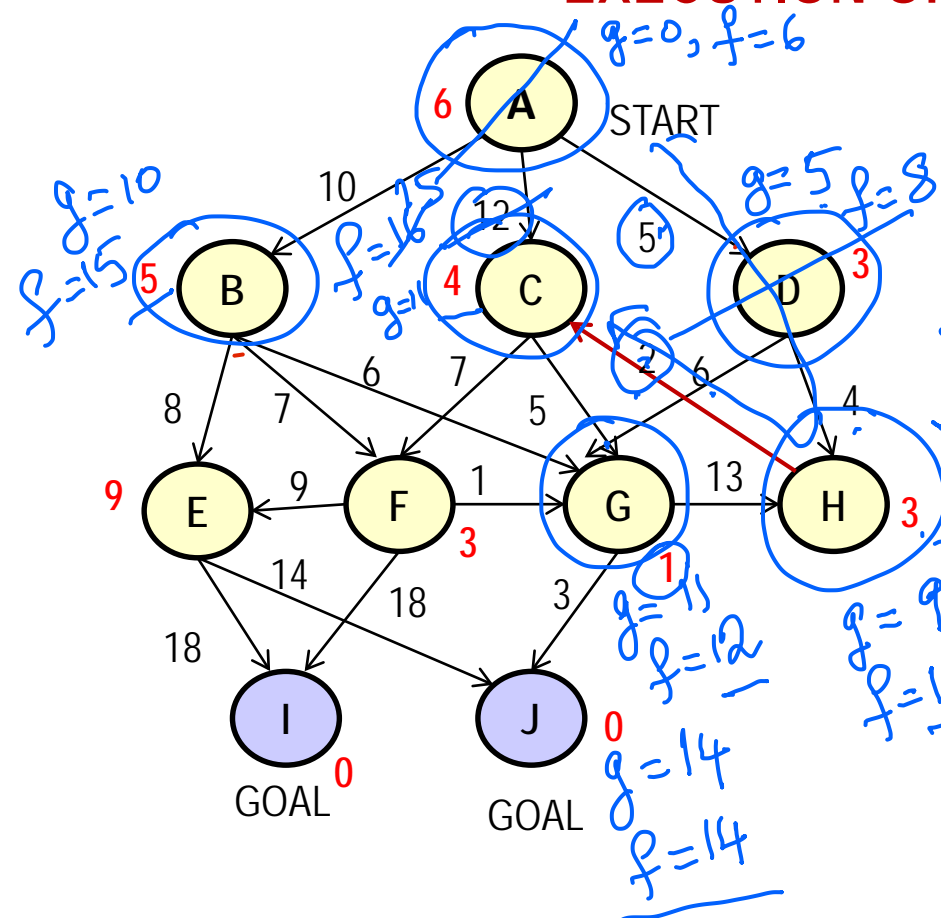
Assume all $c(n,m) > 0$

1. [Initialize] Initially the OPEN List contains the Start Node s . $g(s) = 0$, $f(s) = h(s)$.
CLOSED List is Empty.
2. [Select] Select the Node n on the OPEN List with minimum $f(n)$. If OPEN is empty, Terminate with Failure
3. [Goal Test, Terminate] If n is Goal, then Terminate with Success and path from s to n .
4. [Expand]
 - a) Generate the successors n_1, n_2, \dots, n_k , of node n , based on the State Transformation Rules
 - b) Put n in LIST CLOSED
 - c) For each n_i , not already in OPEN or CLOSED List, compute
 - a) $g(n_i) = g(n) + c(n, n_i)$, $f(n_i) = g(n_i) + h(n_i)$, Put n_i in the OPEN List
 - b) For each n_i already in OPEN, if $g(n_i) > g(n) + c(n, n_i)$, then revise costs as:
 - a) $g(n_i) = g(n) + c(n, n_i)$, $f(n_i) = g(n_i) + h(n_i)$
5. [Continue] Go to Step 2

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

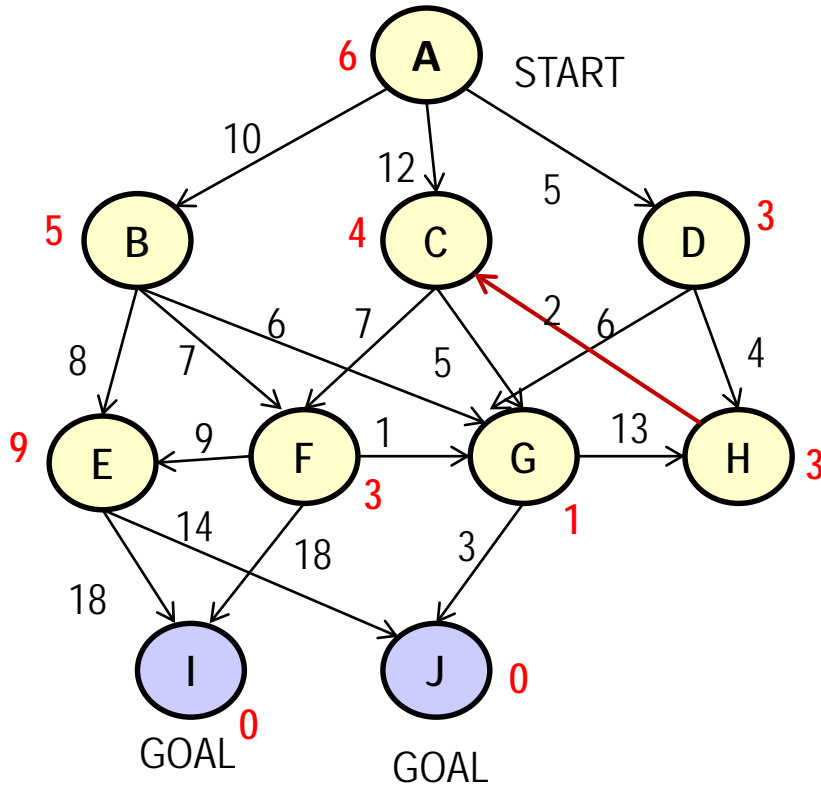
Diagram illustrating the cost calculation for a node n . It shows a node n with arrows pointing to its $g(n)$ and $h(n)$ values. A formula $g' = g(n) + c(n, n_i)$ is shown, indicating the update of the g cost for a successor node n_i .

EXECUTION OF ALGORITHM A*



	OPEN	CLOSED
1	A	
2	B, C, D	A
3	B, C, G, H	D
4	B, C, G	H
5	B, C, J	G
		J

EXECUTION OF ALGORITHM A*

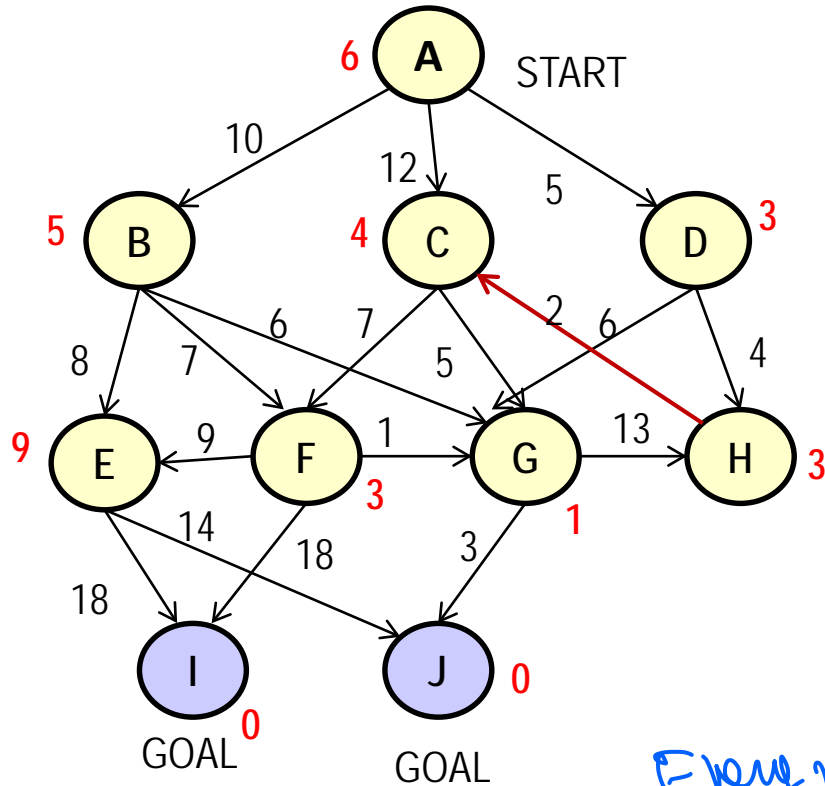


1. OPEN = {A[0,6,6]}, CLOSED = {} ✓
2. OPEN = {B[10,5,15,A], C[12,4,16,A], D[5,3,8,A]},
CLOSED = {A} ✓
3. OPEN = {B[10,5,15,A], C[12,4,16,A], G[11,1,12,D],
H[9,3,12,D]}, CLOSED = {A,D} ✓
4. OPEN = {B[10,5,15,A], C[11,4,15,H], G[11,1,12,D]},
CLOSED = {A,D,H} ✓
5. OPEN = {B[10,5,15,A], C[11,4,15,H], J[14,0,14,G]},
CLOSED = {A[,D[A],H[D],G[D]}
6. Goal J found. Terminate with cost 14 and path
A,D,G,J.

Best-First Algorithm

First solution is the one
we terminate with

PROPERTIES OF ALGORITHM A*



IF HEURISTIC ESTIMATES ARE NON-NEGATIVE
LOWER BOUNDS AND EDGE COSTS ARE POSITIVE:

- FIRST SOLUTION IS OPTIMAL ✓
- NO NODE IN CLOSED IS EVER REOPENED ✓
- WHENEVER A NODE IS REMOVED FROM OPEN ITS MINIMUM COST FROM START IS FOUND
- EVERY NODE n WITH $f(n)$ LESS THAN OPTIMAL COST IS EXPANDED
- IF HEURISTICS ARE MORE ACCURATE THEN SEARCH IS LESS

never come to a node in CLOSED
with lower cost

Every node $f(n) < \text{optimal cost}$ is expanded

$h_1 \leq h_2 \rightarrow$ more accurate

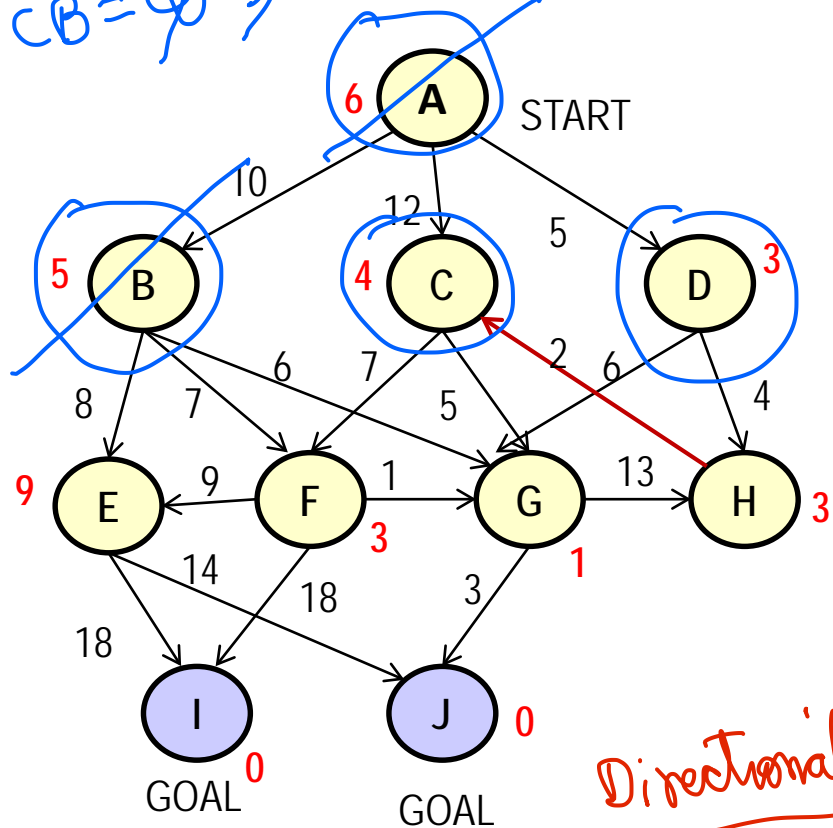
ALGORITHM DFBB

DEPTH FIRST BRANCH AND BOUND (DFBB)

1. Initialize Best-Cost to INFINITY PRUNING
2. Perform DFS with costs and Backtrack from any node n whose $f(n) \geq \text{Best-Cost}$ ✓
3. On reaching a Goal Node, update Best-Cost to the current best
4. Continue till OPEN becomes empty

CB = ~~9~~ 36 32 20

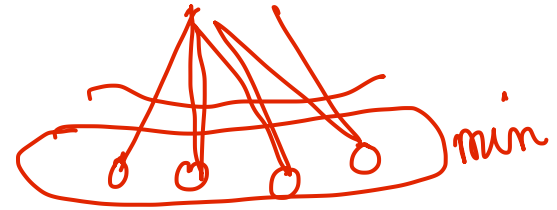
EXECUTION OF DFBB



Directional

	OPEN					CLOSED
1	A D CB					
2	B C D					A
3	E F G C D					B
4	I J F G C D					E
5	J F G C D					I
6	G C D					J
7	J H C D					J

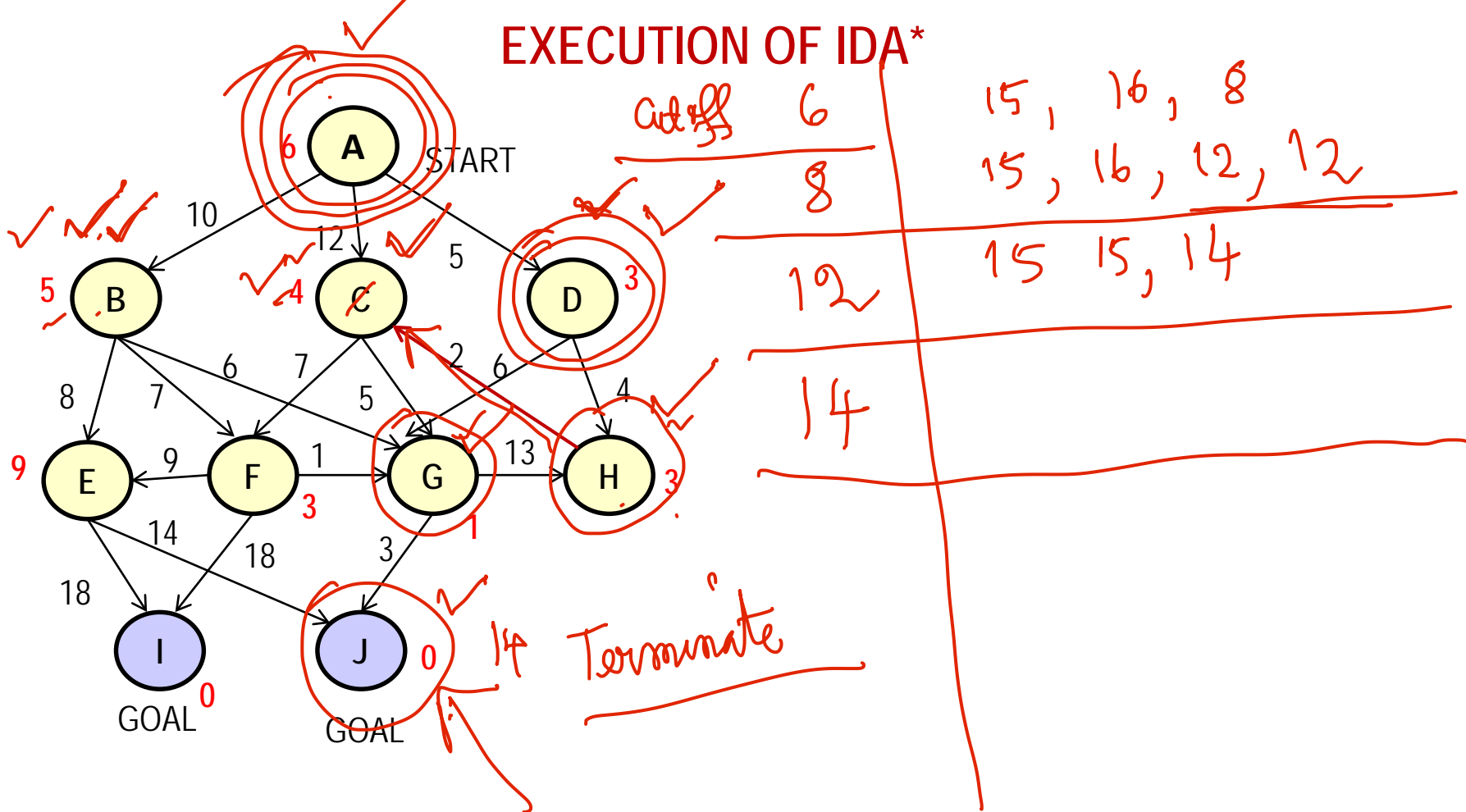
ALGORITHM IDA*



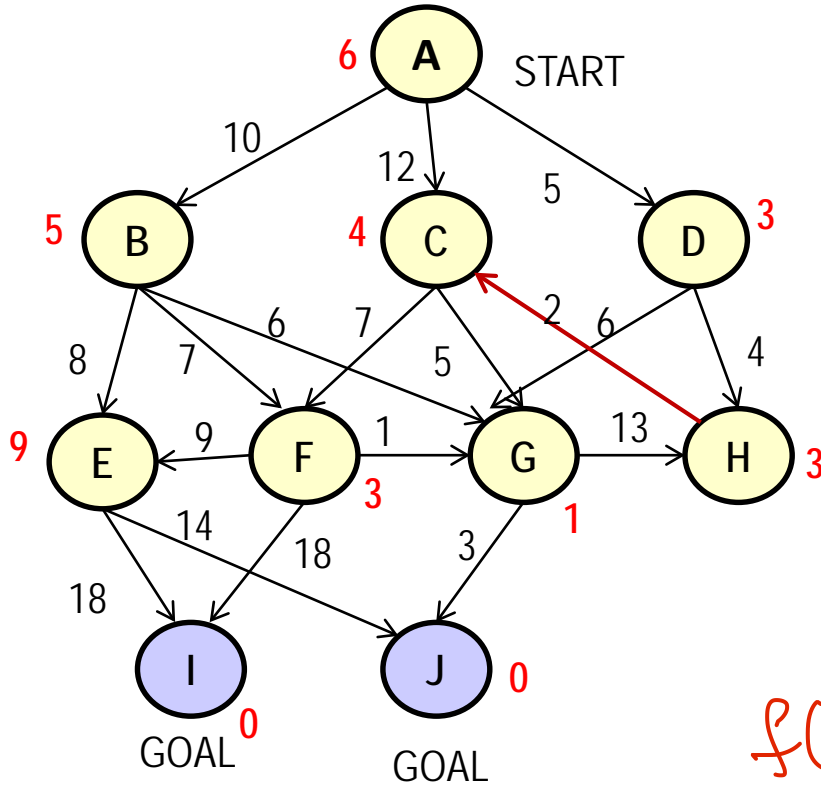
ITERATIVE DEEPENING A* (IDA*)

1. Set Cut-off Bound to $f(s)$
2. Perform DFBB with Cut-off Bound. Backtrack from any node whose $f(n) > \text{Cut-off Bound}$.
3. If Solution is Found, at the end of one Iteration, Terminate with Solution
4. If Solution is not found in any iteration, then update Cut-off Bound to the lowest $f(n)$ among all nodes from which the algorithm Backtracked.
5. Go to Step 2
6. PROPERTIES OF DFBB AND IDA*: Solution Cost, Memory, Node expansions, Heuristic Accuracy, Performance on Trees / Graphs

EXECUTION OF IDA*



COMPARING A*, DFBB & IDA*



1. TERMINATION ISSUES ✓
2. MEMORY AND NODE EXPANSIONS ✓
3. HEURISTIC ACCURACY EFFECT ✓
4. MAXIMIZATION PROBLEMS

Maximization?

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

$h(n) \Rightarrow$ overestimate

2: Remove the node in OPEN with LARGEST $f(n)$

KNAPSACK PROBLEM: MAXIMIZATION



State configuration

→ items currently chosen
total weight, total value

$\langle \{ \}, 0, 0 \rangle$

taken

not taken

$\langle \{ 1 \}, (10, 1000) \rangle$

$\langle \{ \}, 0, 0 \rangle$

$h(n)$

State transformation rules

item 2

Trace of A^* , DFBB, IDA*

Thank you