STATE SPACE SEARCH
&
CASE STUDY ON IUI

Dr Pradipta Biswas, PhD (Cantab)
Assistant Professor
Indian Institute of Science
http://cpdm.iisc.ernet.in/PBiswas.htm

STATE-SPACE SEARCH

- MANY PROBLEMS IN AI TAKE THE FORM OF ***STATE-SPACE SEARCH***.
- THE ***STATES*** MIGHT BE LEGAL BOARD CONFIGURATIONS IN A GAME, TOWNS AND CITIES IN SOME SORT OF ROUTE MAP, COLLECTIONS OF MATHEMATICAL PROPOSITIONS, ETC.
- THE ***STATE-SPACE*** IS THE CONFIGURATION OF THE POSSIBLE STATES AND HOW THEY CONNECT TO EACH OTHER E.G. THE LEGAL MOVES BETWEEN STATES.
- WHEN WE DO NOT HAVE AN ***ALGORITHM*** WHICH TELLS US DEFINITIVELY HOW TO NEGOTIATE THE STATE-SPACE WE NEED TO SEARCH THE STATE-SPACE TO FIND AN OPTIMAL PATH FROM A START STATE TO A GOAL STATE.
- WE CAN ONLY DECIDE WHAT TO DO (OR WHERE TO GO), BY CONSIDERING THE POSSIBLE MOVES FROM THE CURRENT STATE, AND TRYING TO LOOK AHEAD AS FAR AS POSSIBLE. CHESS, FOR EXAMPLE, IS A VERY DIFFICULT STATE-SPACE SEARCH PROBLEM.

STATE-SPACE MODEL

- INITIAL STATE
- OPERATORS: MAPS A STATE INTO A NEXT STATE
    - ALTERNATIVE: SUCCESSORS OF STATE
- GOAL PREDICATE: TEST TO SEE IF GOAL ACHIEVED
- OPTIONAL:
    - COST OF OPERATORS
    - COST OF SOLUTION

WHAT YOU SHOULD KNOW

- CREATE A **STATE-SPACE MODEL**
- ESTIMATE NUMBER OF STATES
- IDENTIFY GOAL OR OBJECTIVE FUNCTION
- IDENTIFY OPERATORS

UNINFORMED SEARCH STRATEGIES

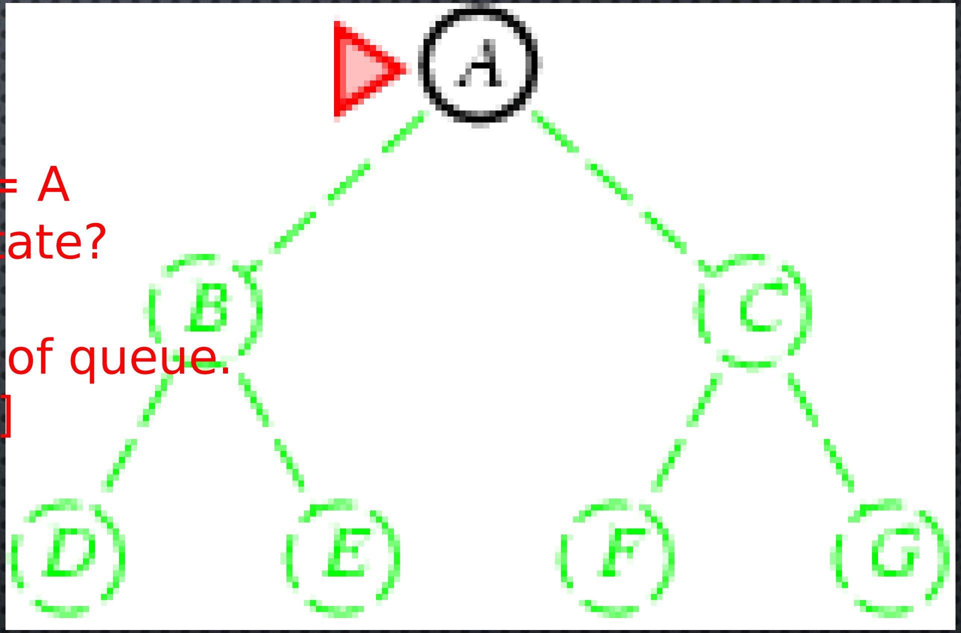
- UNINFORMED (BLIND):
  - YOU HAVE NO CLUE WHETHER ONE NON-GOAL STATE IS BETTER THAN ANY OTHER. YOUR SEARCH IS BLIND. YOU DON’T KNOW IF YOUR CURRENT EXPLORATION IS LIKELY TO BE FRUITFUL.
- VARIOUS BLIND STRATEGIES:
  - BREADTH-FIRST SEARCH
  - UNIFORM-COST SEARCH
  - DEPTH-FIRST SEARCH
  - ITERATIVE DEEPENING SEARCH (GENERALLY PREFERRED)
  - BIDIRECTIONAL SEARCH (PREFERRED IF APPLICABLE)

BREADTH-FIRST SEARCH

- EXPAND SHALLOWEST UNEXPANDED NODE
- *FRONTIER* (OR FRINGE): NODES IN QUEUE TO BE EXPLORED
- *FRONTIER* IS A FIRST-IN-FIRST-OUT (FIFO) QUEUE, I.E., NEW SUCCESSORS GO AT END OF THE QUEUE.
- *GOAL-TEST* WHEN INSERTED.

Initial state = A
Is A a goal state?

Put A at end of queue.
frontier = [A]



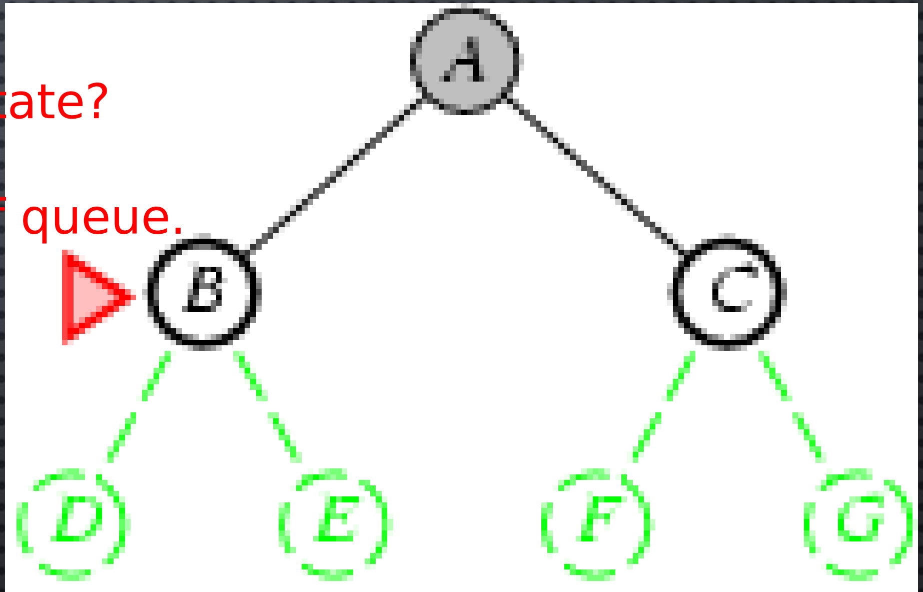
Future= green dotted circles
Frontier=white nodes
Expanded/active=gray nodes
Forgotten/reclaimed= black nodes

BREADTH-FIRST SEARCH

- EXPAND SHALLOWEST UNEXPANDED NODE
- *FRONTIER* IS A FIFO QUEUE, I.E., NEW SUCCESSORS GO AT END

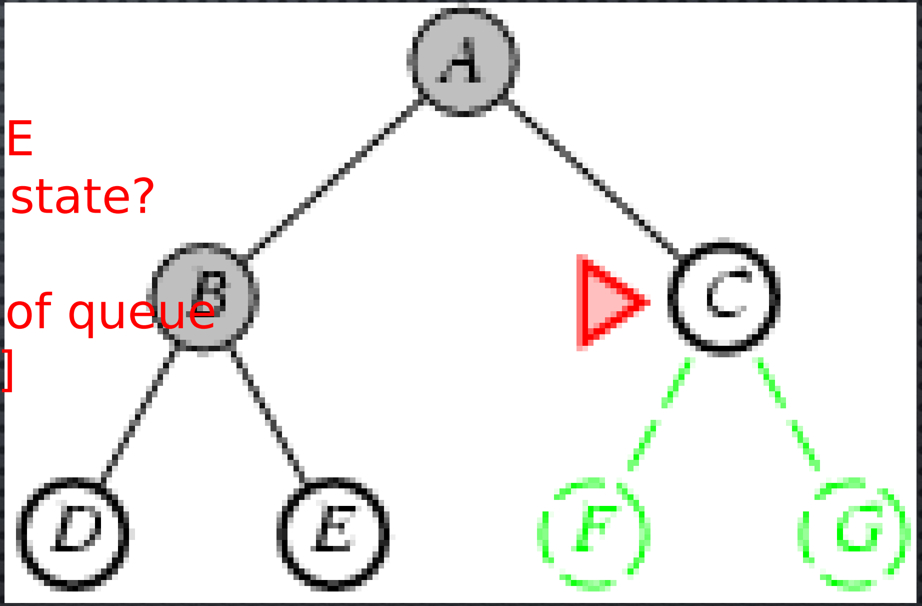
Expand A to B, C.
Is B or C a goal state?

Put B, C at end of queue.
frontier = [B,C]



BREADTH-FIRST SEARCH

- EXPAND SHALLOWEST UNEXPANDED NODE
- *FRONTIER* IS A FIFO QUEUE, I.E., NEW SUCCESSORS GO AT END

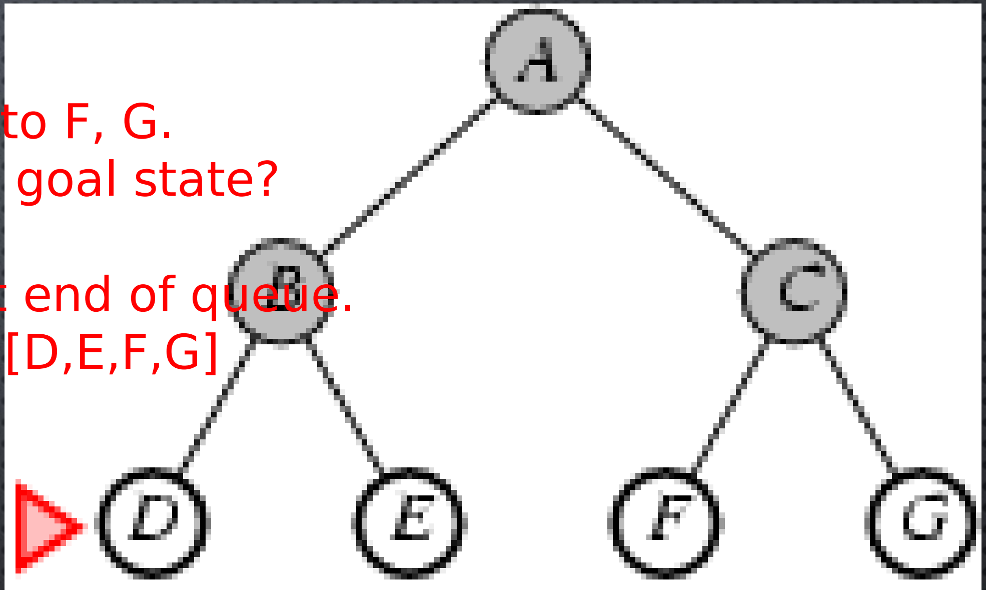


Expand B to D, E
Is D or E a goal state?

Put D, E at end of queue
frontier=[C,D,E]

BREADTH-FIRST SEARCH

- EXPAND SHALLOWEST UNEXPANDED NODE
- *FRONTIER* IS A FIFO QUEUE, I.E., NEW SUCCESSORS GO AT END



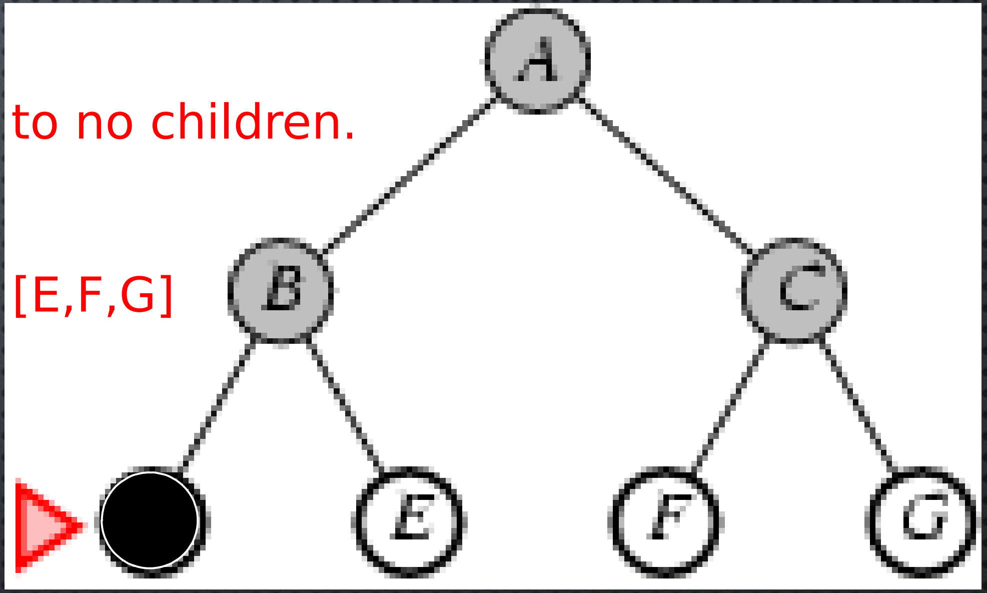
Expand C to F, G.
Is F or G a goal state?

Put F, G at end of queue.
frontier = [D,E,F,G]

BREADTH-FIRST SEARCH

Expand D to no children.
Forget D.

frontier = [E,F,G]



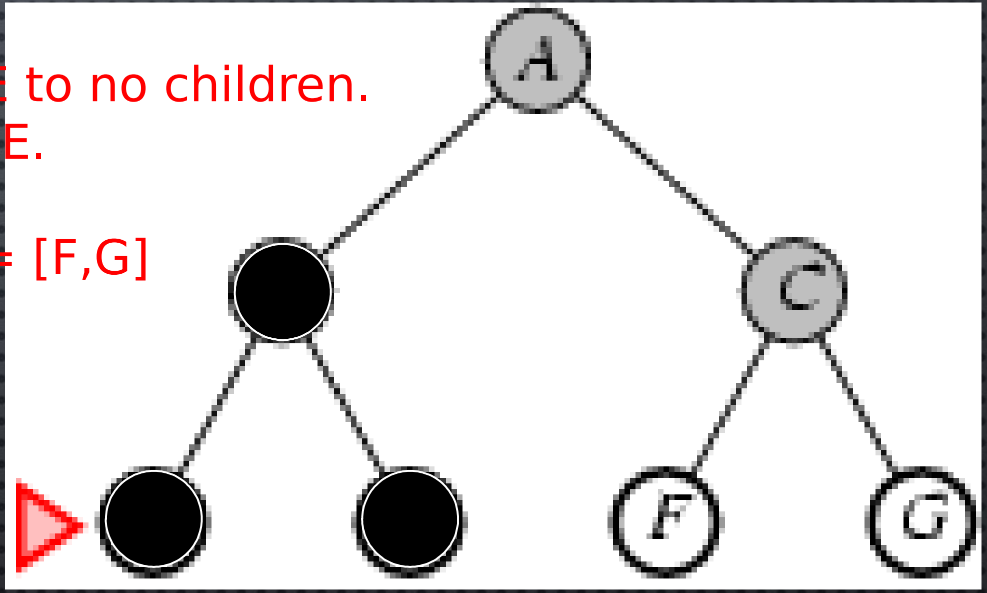
- EXPAND SHALLOWEST UNEXPANDED NODE

- *FRONTIER* IS A FIFO QUEUE, I.E., NEW SUCCESSORS GO AT END

BREADTH-FIRST SEARCH

Expand E to no children.
Forget B,E.

frontier = [F,G]



- EXPAND SHALLOWEST UNEXPANDED NODE

- *FRONTIER* IS A FIFO QUEUE, I.E., NEW SUCCESSORS GO AT END

PROPERTIES OF BREADTH-FIRST SEARCH

- COMPLETE? YES, IT ALWAYS REACHES A GOAL (IF *B* IS FINITE)
- TIME? $1+B+B^2+B^3+\ldots + B^D$ = O(B^D)

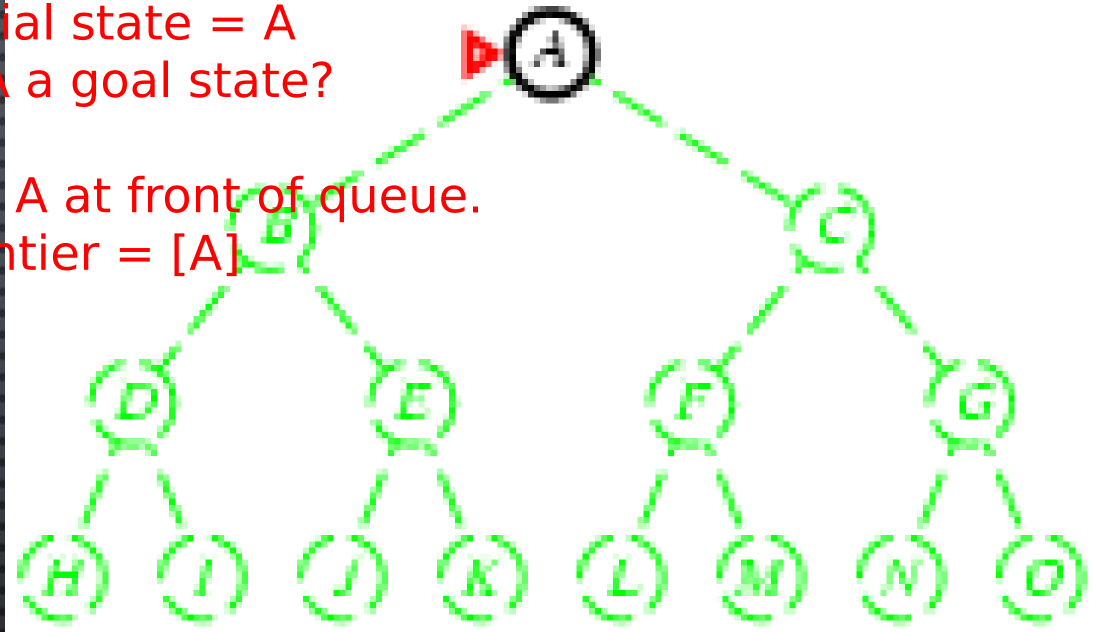
  (THIS IS THE NUMBER OF NODES WE GENERATE)
- SPACE? $O(B^D)$ (KEEPS EVERY NODE IN MEMORY,

  EITHER IN FRINGE OR ON A PATH TO FRINGE).
- OPTIMAL? NO, FOR GENERAL COST FUNCTIONS.

  YES, IF COST IS A NON-DECREASING FUNCTION ONLY OF DEPTH.
  - WITH F(D) ≥ F(D-1), E.G., STEP-COST = CONSTANT:
    - ALL OPTIMAL GOAL NODES OCCUR ON THE SAME LEVEL
    - OPTIMAL GOAL NODES ARE ALWAYS SHALLOWER THAN NON-OPTIMAL GOALS
    - AN OPTIMAL GOAL WILL BE FOUND BEFORE ANY NON-OPTIMAL GOAL
- SPACE IS THE BIGGER PROBLEM (MORE THAN TIME)

DEPTH-FIRST SEARCH

- EXPAND *DEEPEST* UNEXPANDED NODE
- *FRONTIER* = LAST IN FIRST OUT (LIFO) QUEUE, I.E., NEW SUCCESSORS GO AT THE FRONT OF THE QUEUE.
- *GOAL-TEST* WHEN INSERTED.



Initial state = A
Is A a goal state?

Put A at front of queue.
frontier = [A]

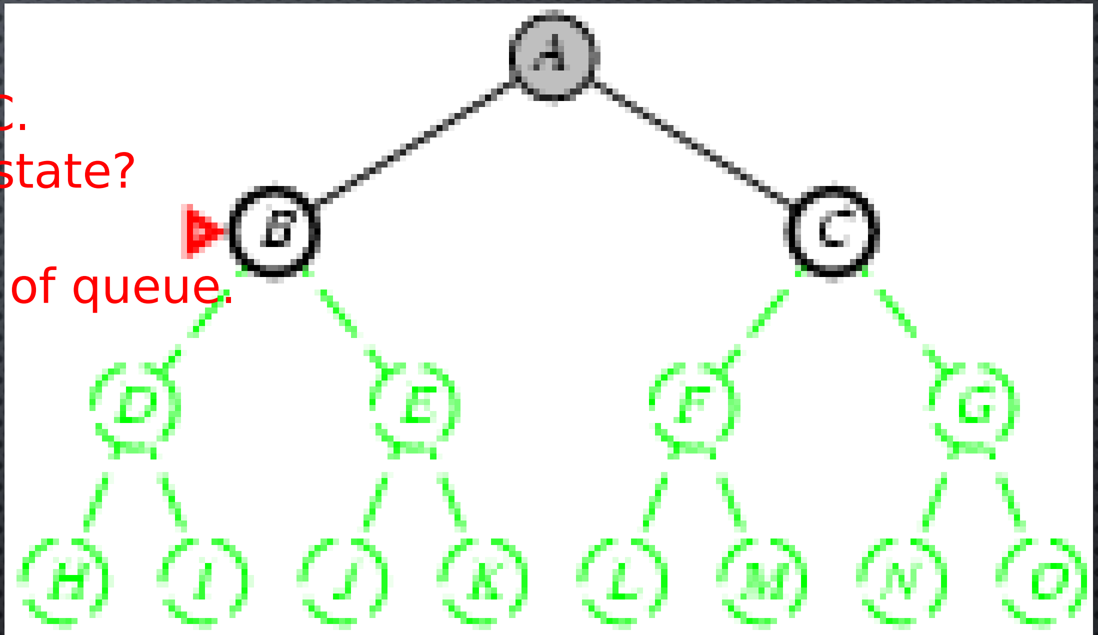
Future= green dotted circles
Frontier=white nodes
Expanded/active=gray nodes
Forgotten/reclaimed= black nodes

DEPTH-FIRST SEARCH

Expand A to B, C.
Is B or C a goal state?

Put B, C at front of queue.
frontier = [B,C]

- EXPAND DEEPEST UNEXPANDED NODE
  - *FRONTIER* = LIFO QUEUE, I.E., PUT SUCCESSORS AT FRONT



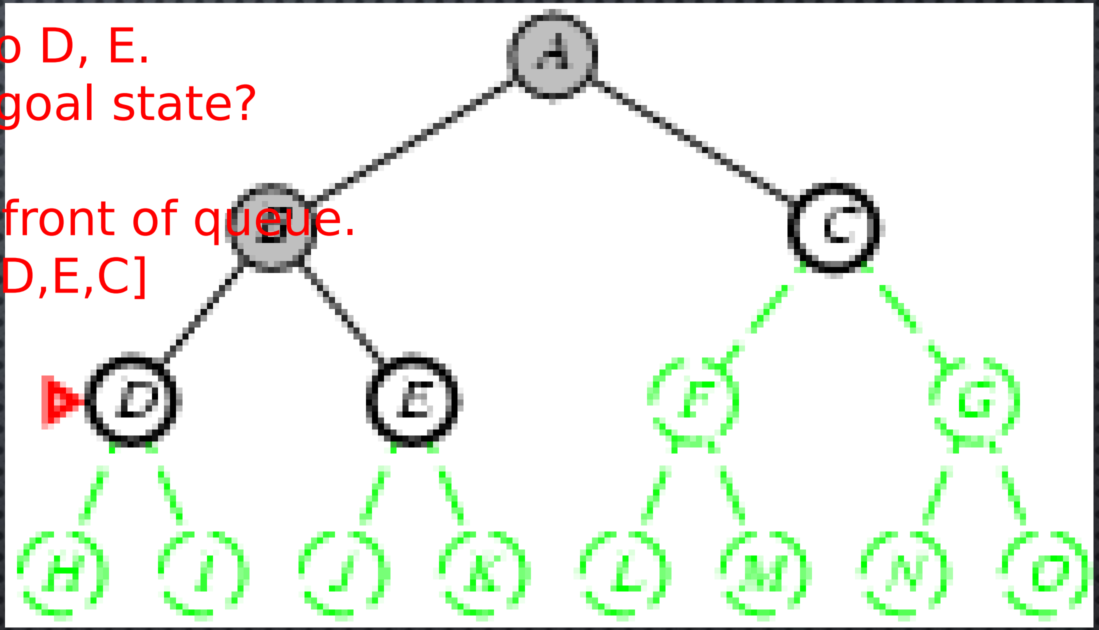
Future= green dotted circles
Frontier=white nodes
Expanded/active=gray nodes
Forgotten/reclaimed= black nodes

DEPTH-FIRST SEARCH

- EXPAND DEEPEST UNEXPANDED NODE
  - *FRONTIER* = LIFO QUEUE, I.E., PUT SUCCESSORS AT FRONT

Expand B to D, E.
Is D or E a goal state?

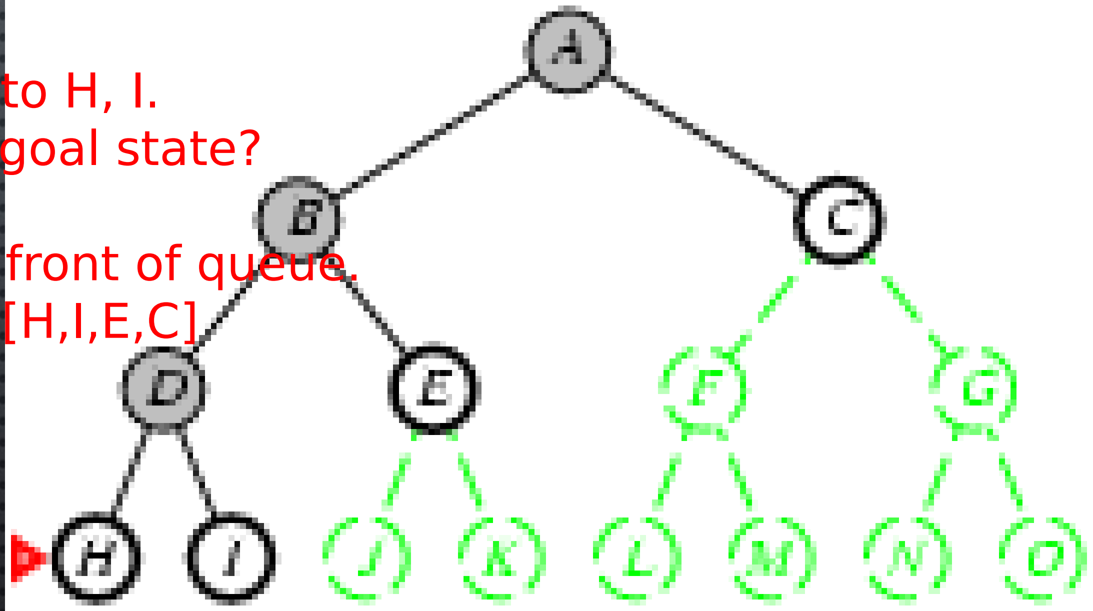
Put D, E at front of queue.
frontier = [D,E,C]



Future= green dotted circles
Frontier=white nodes
Expanded/active=gray nodes
Forgotten/reclaimed= black nodes

DEPTH-FIRST SEARCH

- EXPAND DEEPEST UNEXPANDED NODE
  - *FRONTIER* = LIFO QUEUE, I.E., PUT SUCCESSORS AT FRONT



Expand D to H, I.
Is H or I a goal state?

Put H, I at front of queue.
frontier = [H,I,E,C]

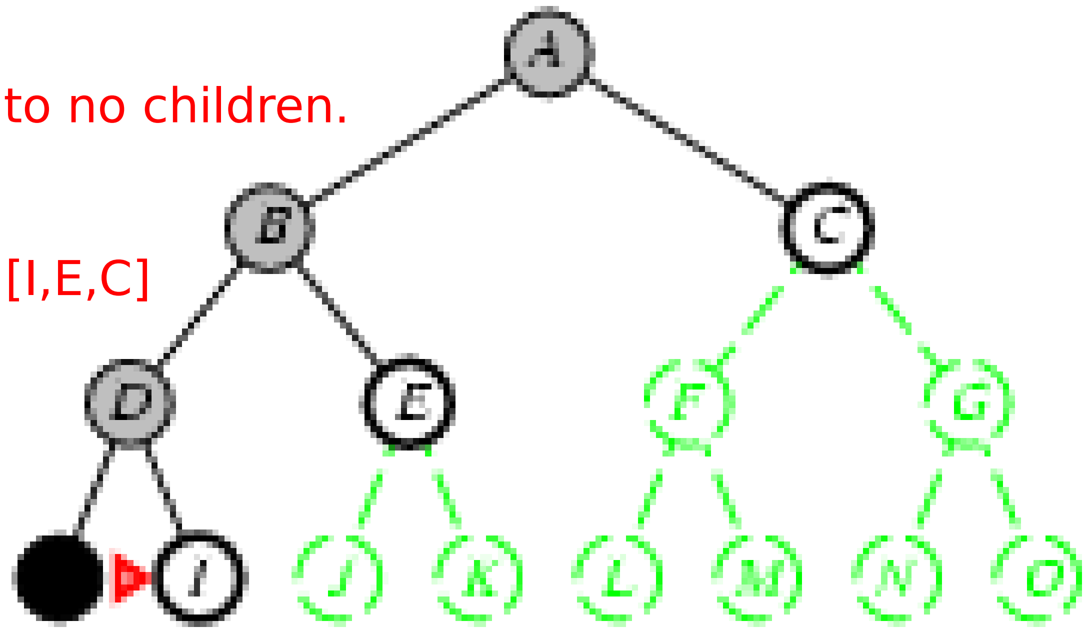
Future= green dotted circles
Frontier=white nodes
Expanded/active=gray nodes
Forgotten/reclaimed= black nodes

DEPTH-FIRST SEARCH

- EXPAND DEEPEST UNEXPANDED NODE
  - *FRONTIER* = LIFO QUEUE, I.E., PUT SUCCESSORS AT FRONT

Expand H to no children.
Forget H.

frontier = [I,E,C]



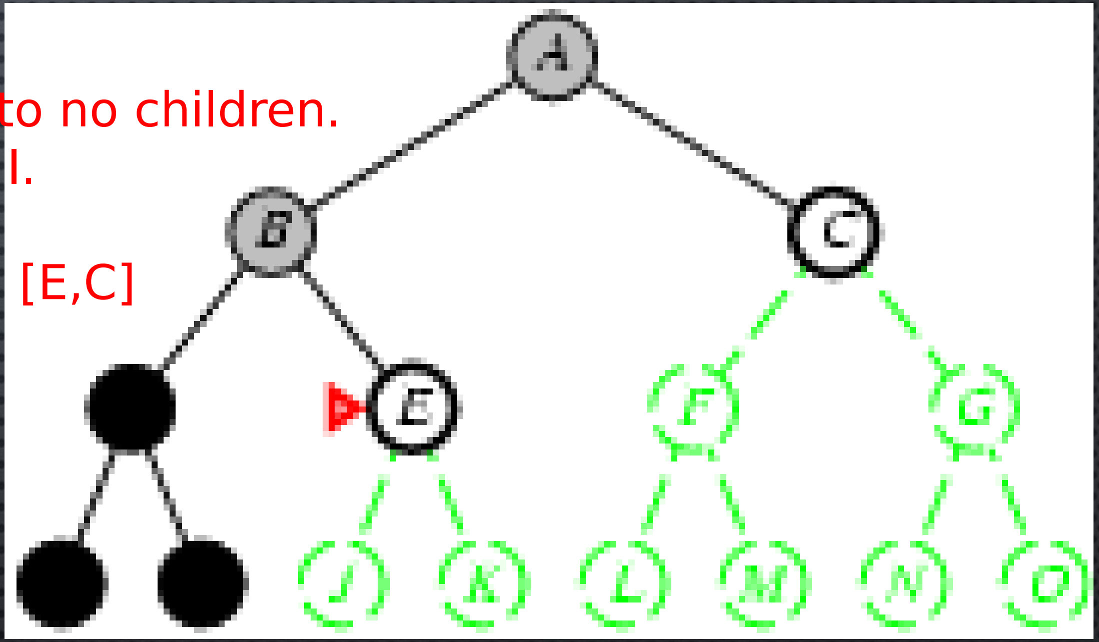
Future= green dotted circles
Frontier=white nodes
Expanded/active=gray nodes
Forgotten/reclaimed= black nodes

DEPTH-FIRST SEARCH

- EXPAND DEEPEST UNEXPANDED NODE
  - *FRONTIER* = LIFO QUEUE, I.E., PUT SUCCESSORS AT FRONT

Expand I to no children.
Forget D, I.

frontier = [E,C]



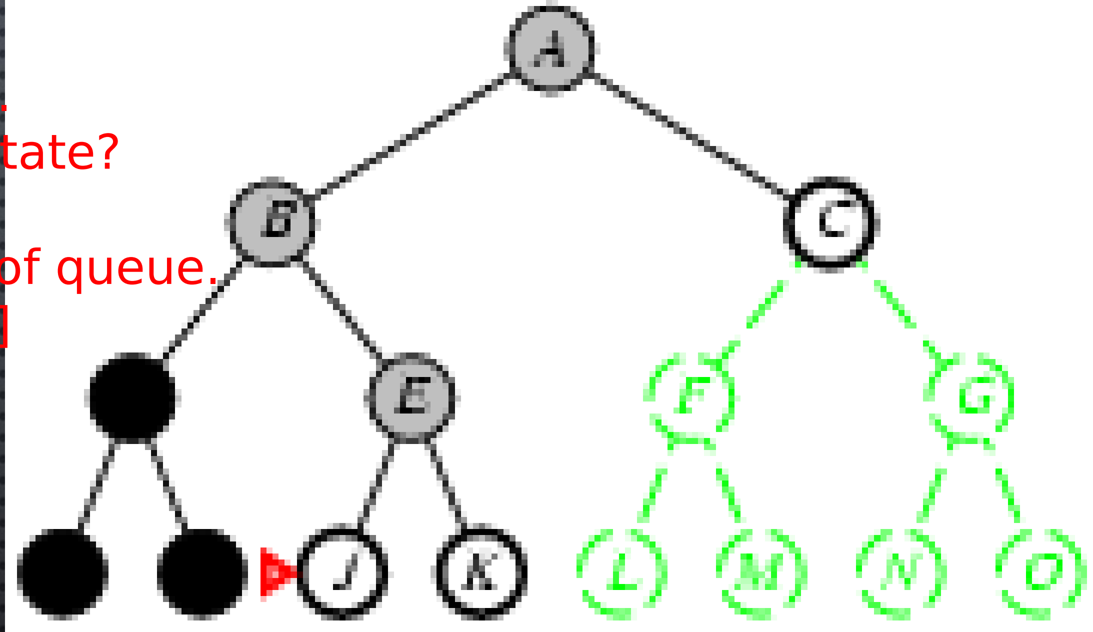
Future= green dotted circles
Frontier=white nodes
Expanded/active=gray nodes
Forgotten/reclaimed= black nodes

DEPTH-FIRST SEARCH

Expand E to J, K.
Is J or K a goal state?

Put J, K at front of queue.
frontier = [J,K,C]

- EXPAND DEEPEST UNEXPANDED NODE
  - *FRONTIER* = LIFO QUEUE, I.E., PUT SUCCESSORS AT FRONT



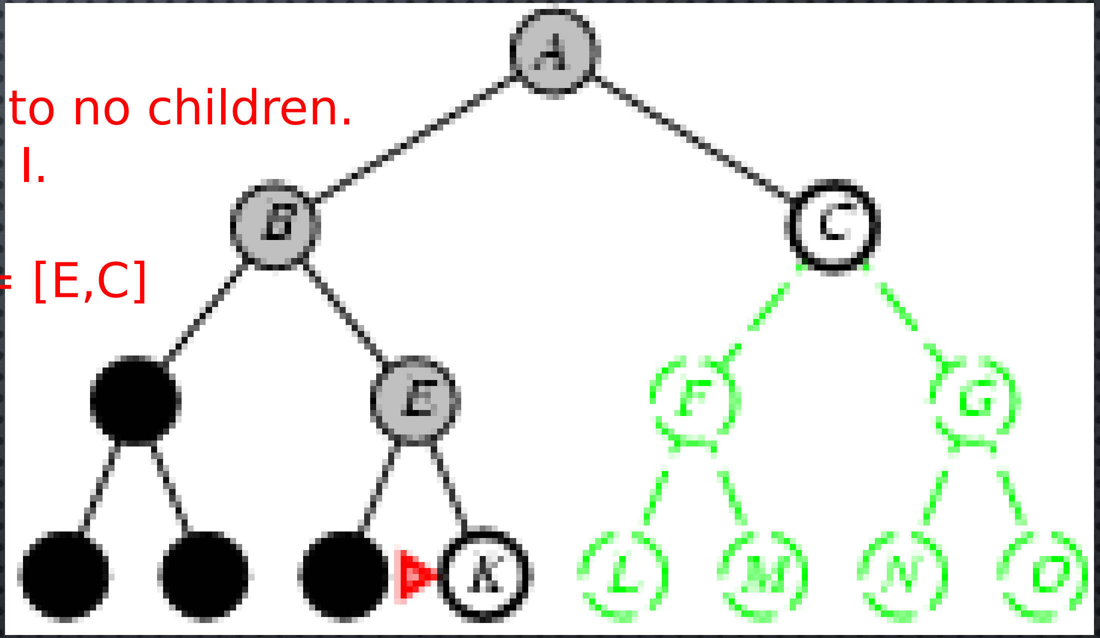
Future= green dotted circles
Frontier=white nodes
Expanded/active=gray nodes
Forgotten/reclaimed= black nodes

DEPTH-FIRST SEARCH

- EXPAND DEEPEST UNEXPANDED NODE
  - *FRONTIER* = LIFO QUEUE, I.E., PUT SUCCESSORS AT FRONT

Expand I to no children.
Forget D, I.

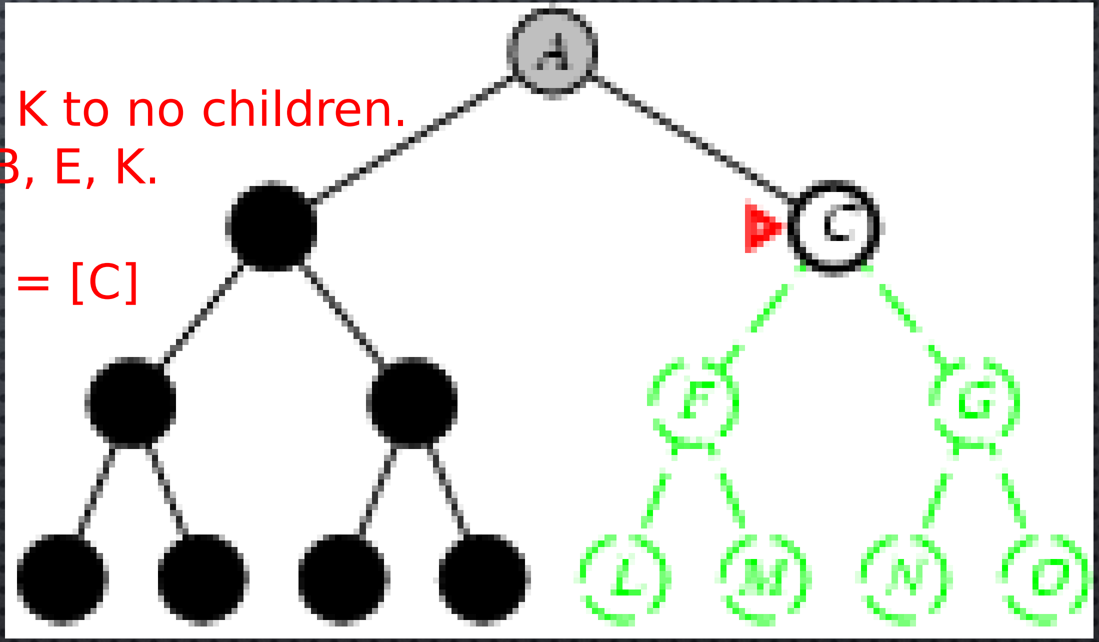
frontier = [E,C]



Future= green dotted circles
Frontier=white nodes
Expanded/active=gray nodes
Forgotten/reclaimed= black nodes

DEPTH-FIRST SEARCH

- EXPAND DEEPEST UNEXPANDED NODE
  - *FRONTIER* = LIFO QUEUE, I.E., PUT SUCCESSORS AT FRONT



Expand K to no children.
Forget B, E, K.

frontier = [C]

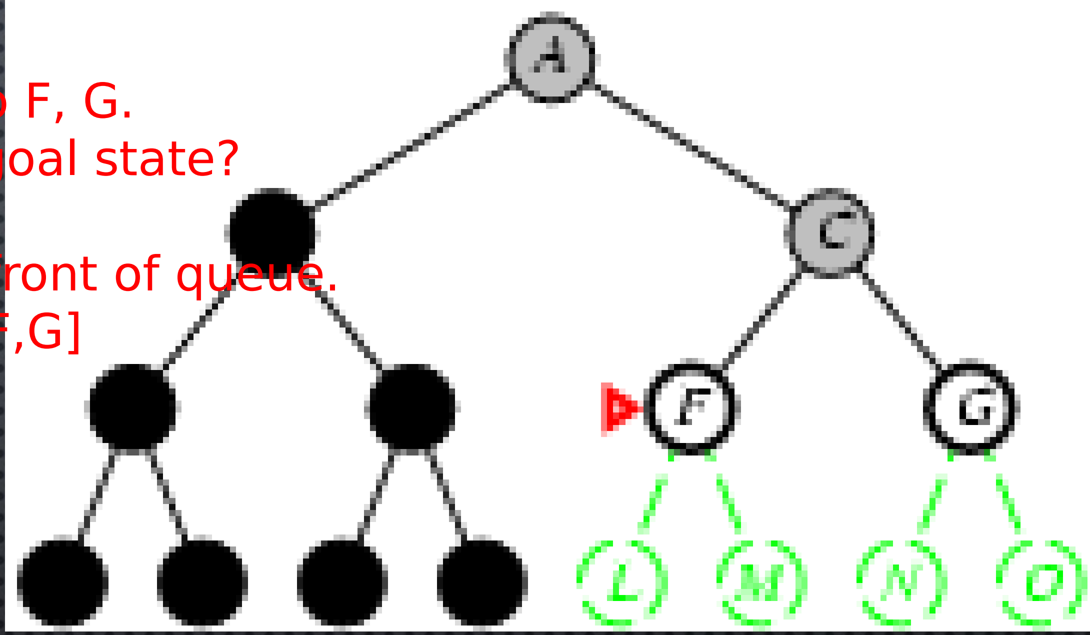
Future= green dotted circles
Frontier=white nodes
Expanded/active=gray nodes
Forgotten/reclaimed= black nodes

DEPTH-FIRST SEARCH

- EXPAND DEEPEST UNEXPANDED NODE
  - *FRONTIER* = LIFO QUEUE, I.E., PUT SUCCESSORS AT FRONT

Expand C to F, G.
Is F or G a goal state?

Put F, G at front of queue.
frontier = [F,G]



Future= green dotted circles
Frontier=white nodes
Expanded/active=gray nodes
Forgotten/reclaimed= black nodes

PROPERTIES OF DEPTH-FIRST SEARCH

- COMPLETE? NO: FAILS IN LOOPS/INFINITE-DEPTH SPACES
    - CAN MODIFY TO AVOID LOOPS/REPEATED STATES ALONG PATH
        - CHECK IF CURRENT NODES OCCURRED BEFORE ON PATH TO ROOT
    - CAN USE GRAPH SEARCH (REMEMBER ALL NODES EVER SEEN)
        - PROBLEM WITH GRAPH SEARCH: SPACE IS EXPONENTIAL, NOT LINEAR
    - STILL FAILS IN INFINITE-DEPTH SPACES (MAY MISS GOAL ENTIRELY)
- TIME? $O(B^M)$ WITH M =MAXIMUM DEPTH OF SPACE
    - TERRIBLE IF M IS MUCH LARGER THAN D
    - IF SOLUTIONS ARE DENSE, MAY BE MUCH FASTER THAN BFS
- SPACE? $O(BM)$, I.E., LINEAR SPACE!
    - REMEMBER A SINGLE PATH + EXPANDED UNEXPLORED NODES
- OPTIMAL? NO: IT MAY FIND A NON-OPTIMAL GOAL FIRST

DEPTH-FIRST VS. BREADTH-FIRST

ADVANTAGES OF DEPTH-FIRST:

- NEEDS RELATIVELY SMALL MEMORY FOR STORING THE STATE-SPACE.

DISADVANTAGES OF DEPTH-FIRST:

- CAN SOMETIMES FAIL TO FIND A SOLUTION;
- NOT GUARANTEED TO FIND AN *OPTIMAL* SOLUTION;
- CAN TAKE A LOT LONGER TO FIND A SOLUTION.

ADVANTAGES OF BREADTH-FIRST:

- GUARANTEED TO FIND A SOLUTION (IF ONE EXISTS);
- DEPENDING ON THE PROBLEM, CAN BE GUARANTEED TO FIND AN *OPTIMAL* SOLUTION.

DISADVANTAGES OF BREADTH-FIRST:

- NEEDS A LOT OF MEMORY FOR STORING THE STATE SPACE IF THE SEARCH SPACE HAS A HIGH BRANCHING FACTOR.

ITERATIVE DEEPENING SEARCH

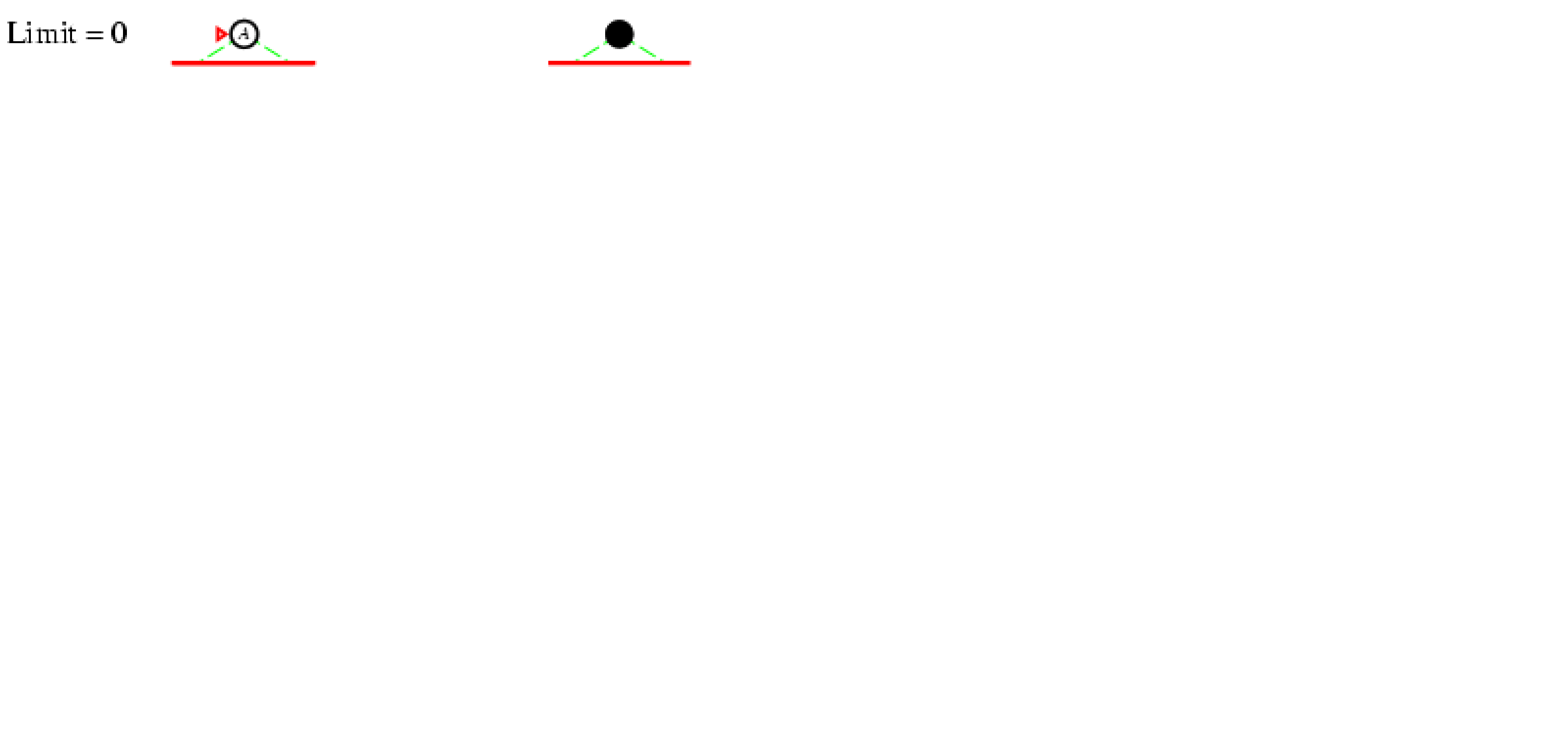
- To avoid the infinite depth problem of DFS,
  only search until depth L,
  i.e., we don’t expand nodes beyond depth L.
   Depth-Limited Search

- What if solution is deeper than L? Increase L iteratively.
   Iterative Deepening Search

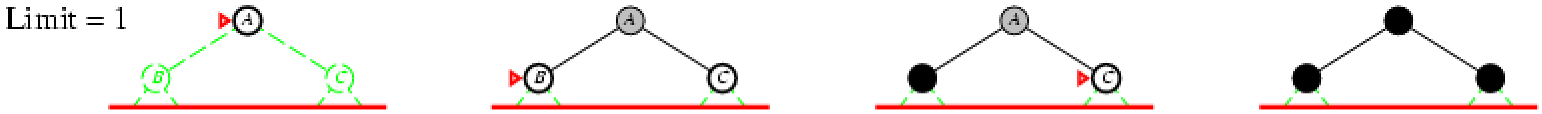
- This inherits the memory advantage of Depth-first search

- Better in terms of space complexity than Breadth-first search.

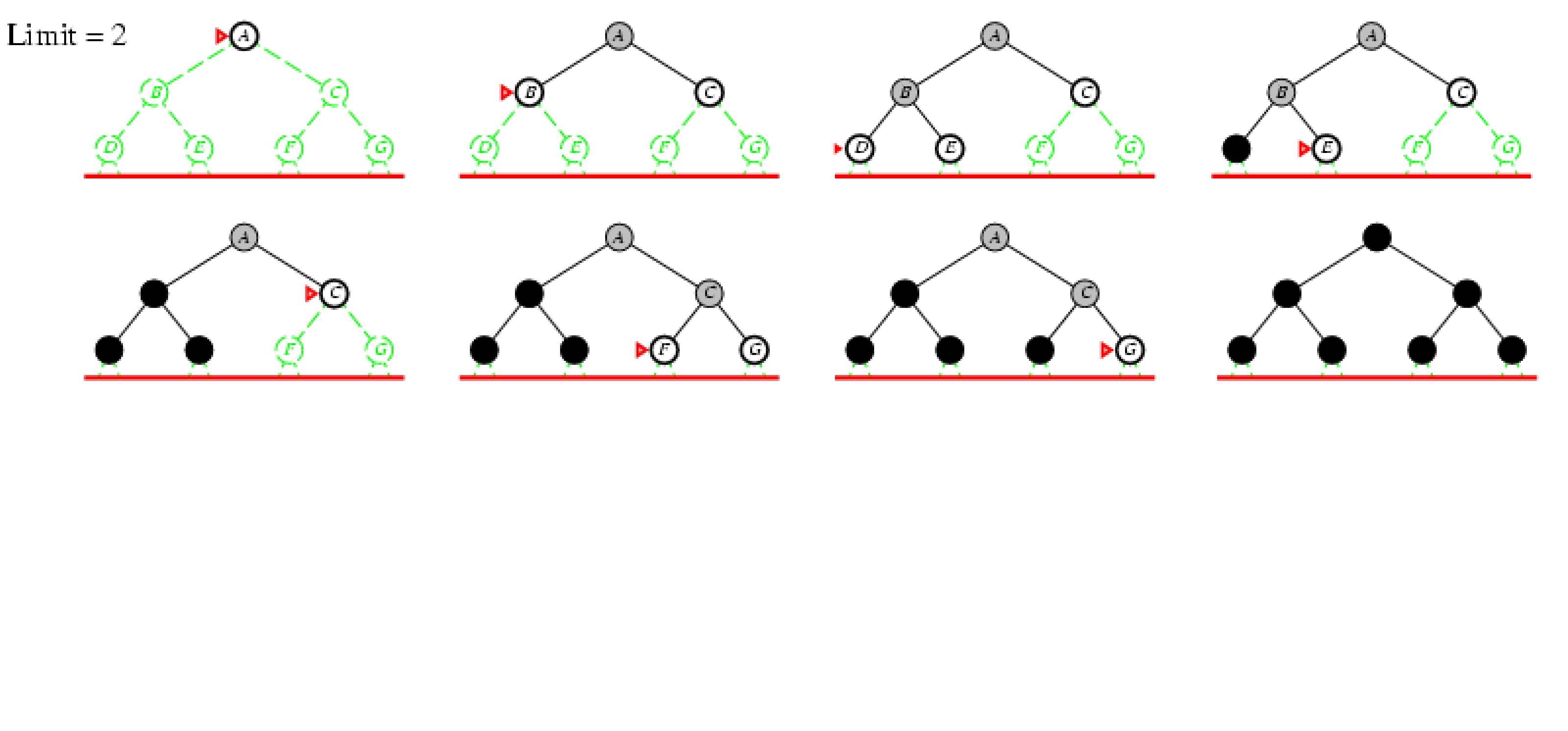
ITERATIVE DEEPENING SEARCH $L=0$



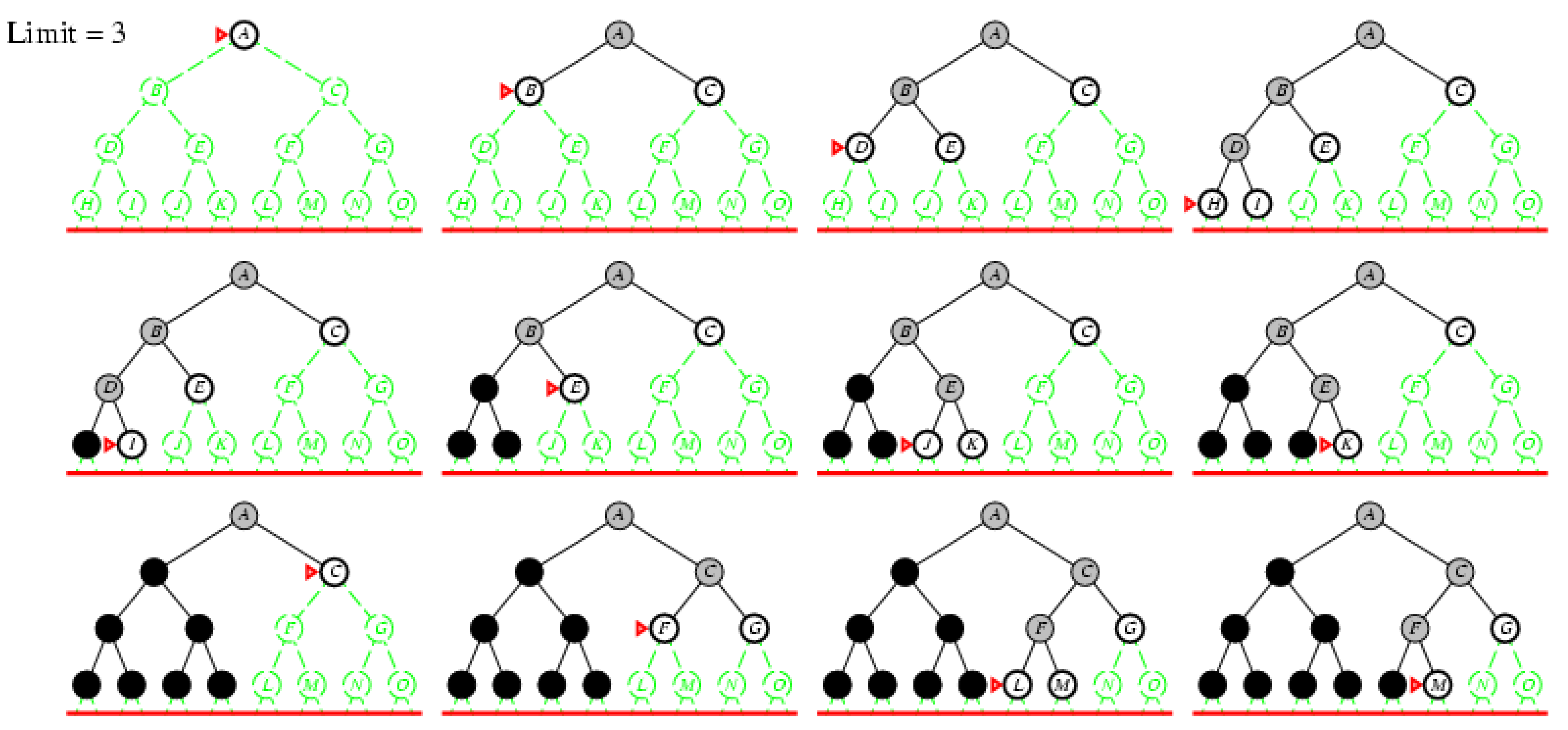
ITERATIVE DEEPENING SEARCH *L*=1



ITERATIVE DEEPENING SEARCH $L=2$



ITERATIVE DEEPENING SEARCH *L*=3



BIDIRECTIONAL SEARCH

- IDEA
  - SIMULTANEOUSLY SEARCH FORWARD FROM S AND BACKWARDS FROM G
  - STOP WHEN BOTH “MEET IN THE MIDDLE”
  - NEED TO KEEP TRACK OF THE INTERSECTION OF 2 OPEN SETS OF NODES
- WHAT DOES SEARCHING BACKWARDS FROM G MEAN
  - NEED A WAY TO SPECIFY THE PREDECESSORS OF G
    - THIS CAN BE DIFFICULT,
    - E.G., PREDECESSORS OF CHECKMATE IN CHESS?
  - WHICH TO TAKE IF THERE ARE MULTIPLE GOAL STATES?
  - WHERE TO START IF THERE IS ONLY A GOAL TEST, NO EXPLICIT LIST?

INFORMED SEARCH STRATEGIES

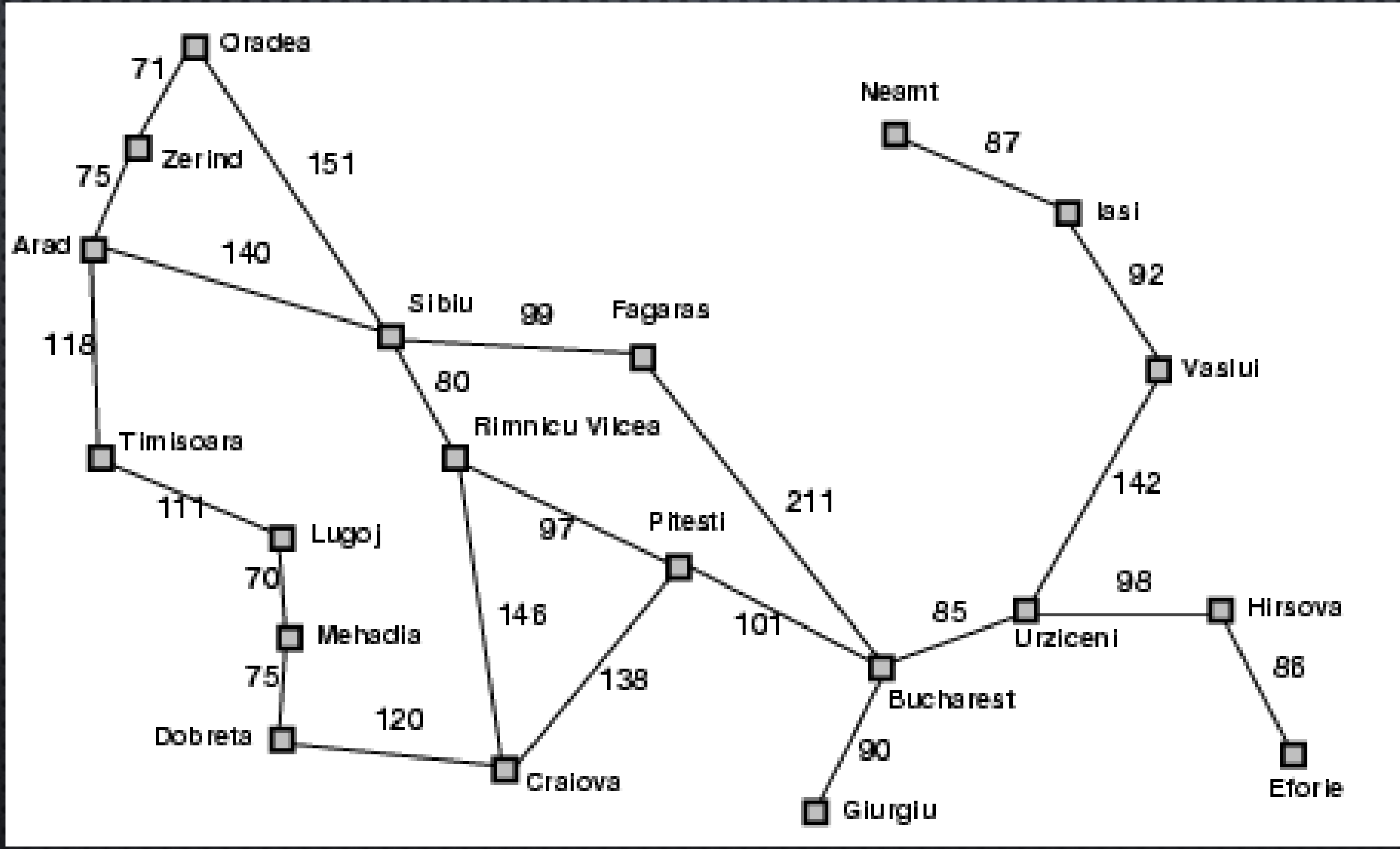
USE HEURISTIC KNOWLEDGE TO INCREAESE EFFICIENCY OF SEARCH:

- **SELECT WHICH NODE TO EXPAND NEXT DURING SEARCH**
- WHILE EXPANDING A NODE DECIDE WHICH SUCCESSORS TO GENERATE AND WHICH TO IGNORE
- REMOVE FROM THE SEARCH SPACE SOME NODES THAT HAVE PREVIOUSLY BEEN GENERATED – PRUNE THE SEARCH SPACE

BEST-FIRST SEARCH

- EVALUATE THE INFORMATION THAT CAN BE OBTAINED BY EXPANDING A NODE AND ITS IMPORTANCE IN GUIDING THE SEARCH
- THE QUALITY OF A NODE IS ESTIMATED BY THE *HEURISTIC SEARCH FUNCTION* **W(N)** FOR NODE N
- GREEDY STRATEGY – GO FOR THE BEST

ROMANIA WITH STRAIGHT-LINE DISTANCE



Straight–line distance to Bucharest

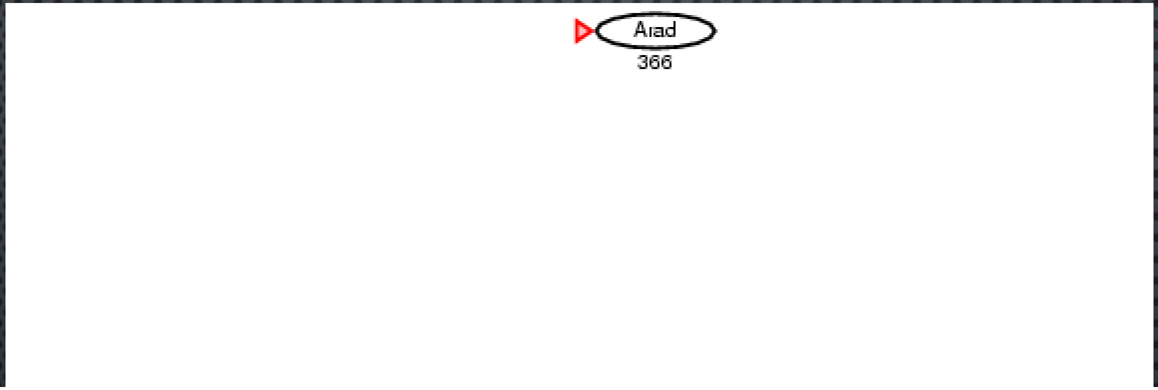
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

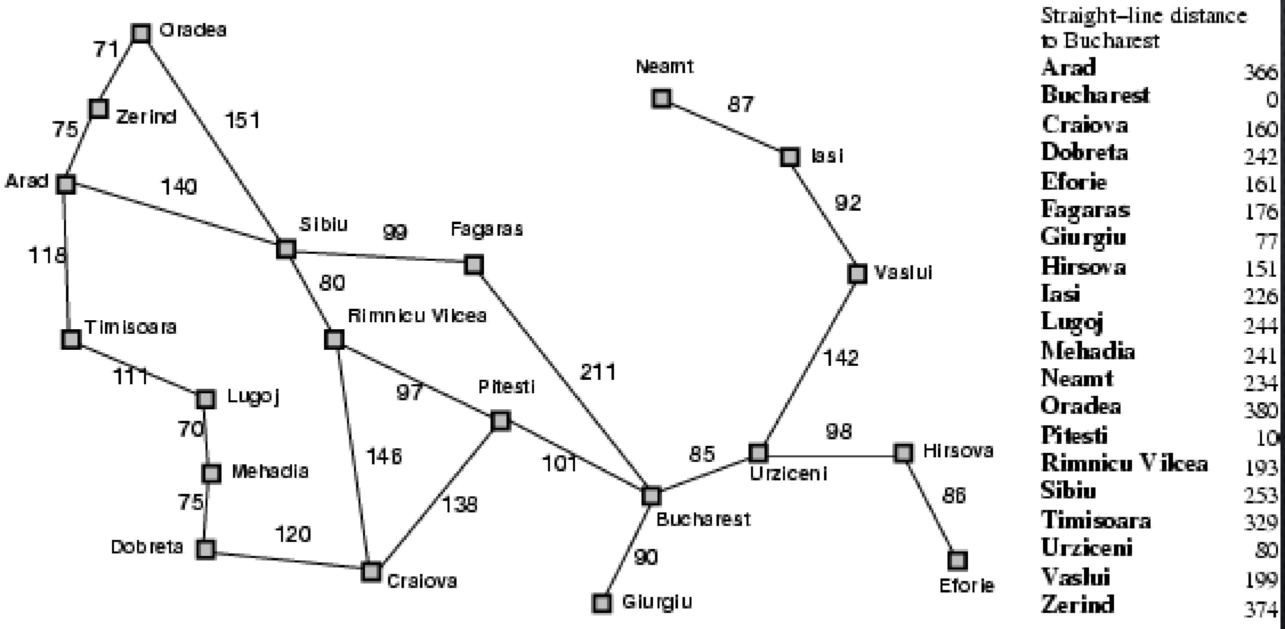
GREEDY BEST-FIRST SEARCH
(OFTEN CALLED JUST “BEST-FIRST”)

- *H(N)* = ESTIMATE OF COST FROM *N* TO *GOAL*
  - E.G., *H(N)* = STRAIGHT-LINE DISTANCE FROM *N* TO BUCHAREST

- GREEDY BEST-FIRST SEARCH EXPANDS THE NODE THAT APPEARS TO BE CLOSEST TO GOAL.
  - *PRIORITY QUEUE SORT FUNCTION = H(N)*

GREEDY BEST-FIRST SEARCH EXAMPLE

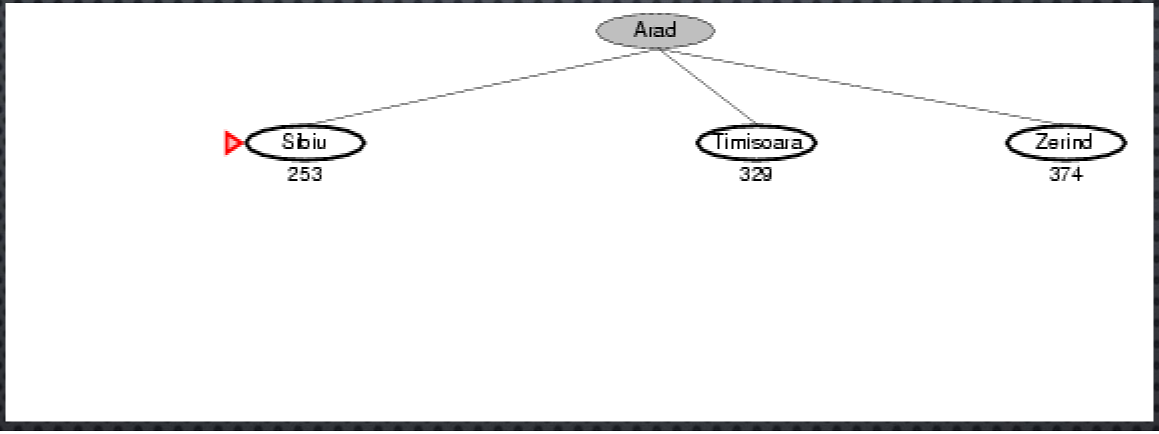


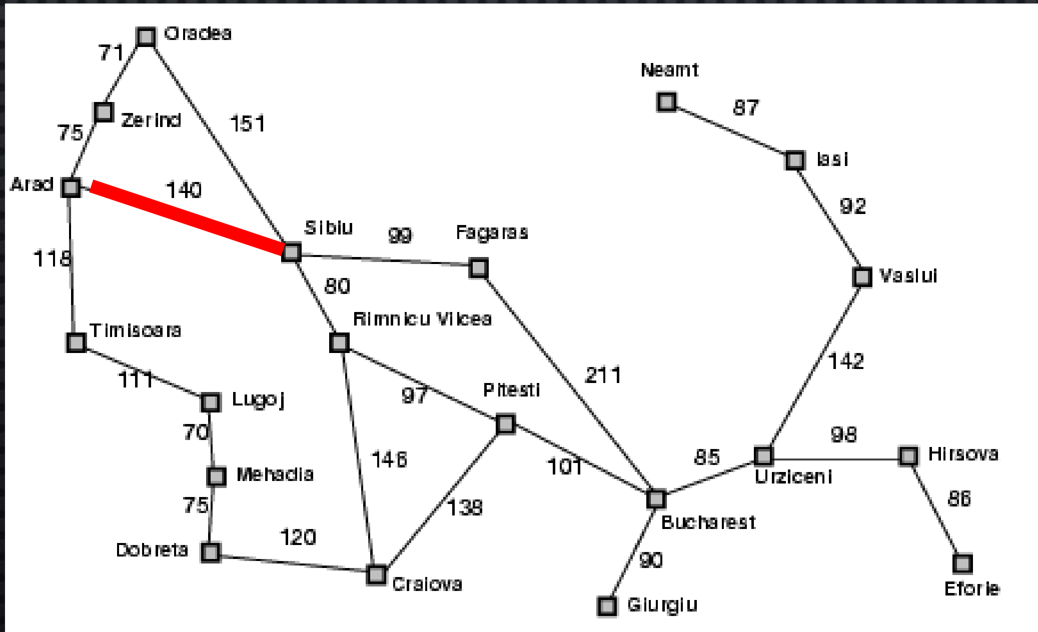


Straight–line distance
to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

GREEDY BEST-FIRST SEARCH EXAMPLE

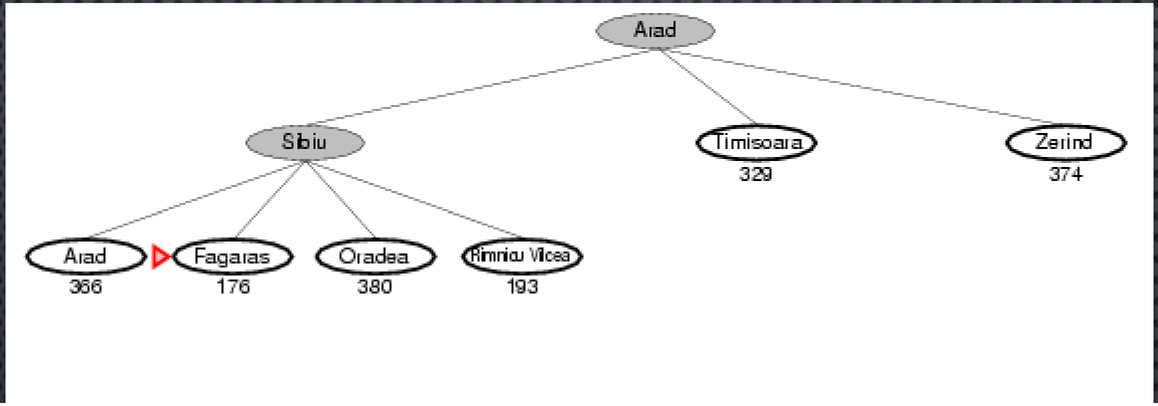


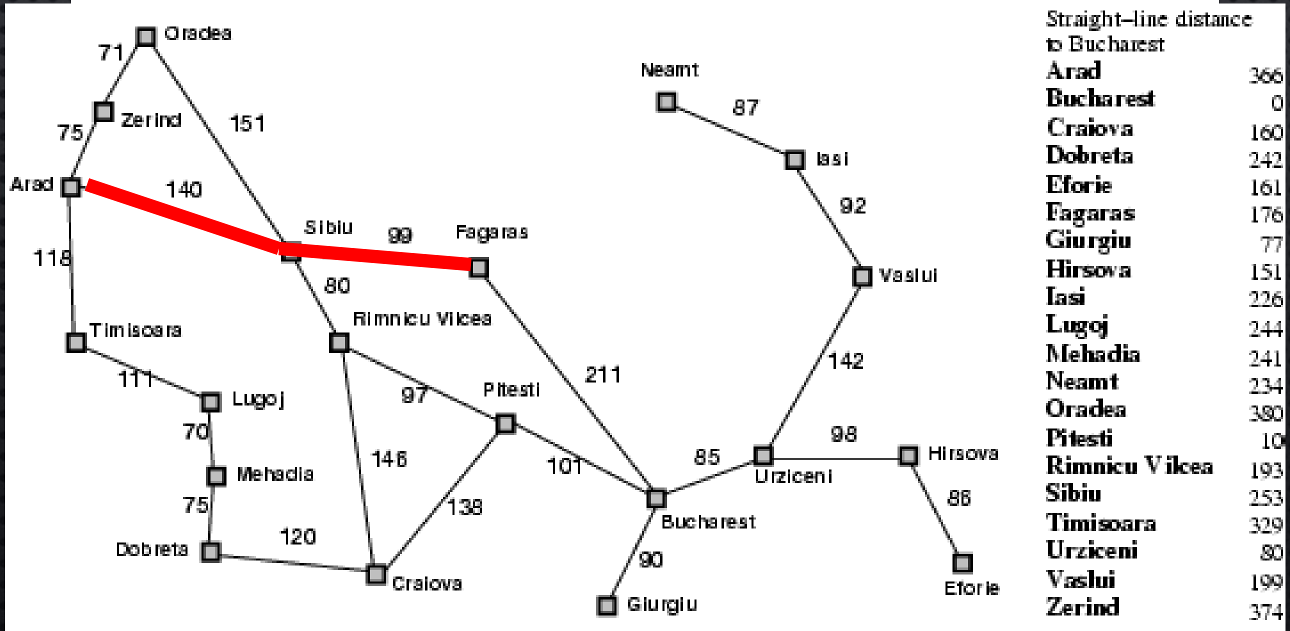


Straight–line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

GREEDY BEST-FIRST SEARCH EXAMPLE

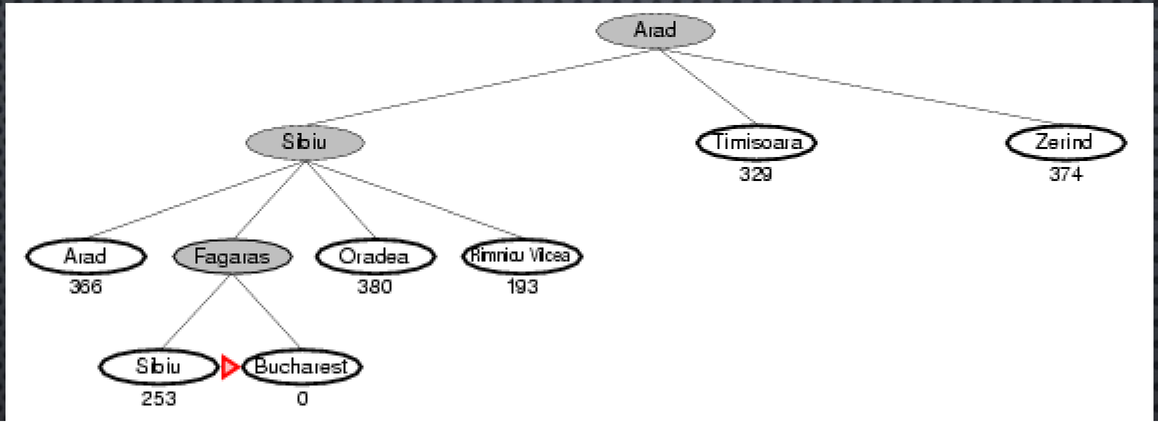


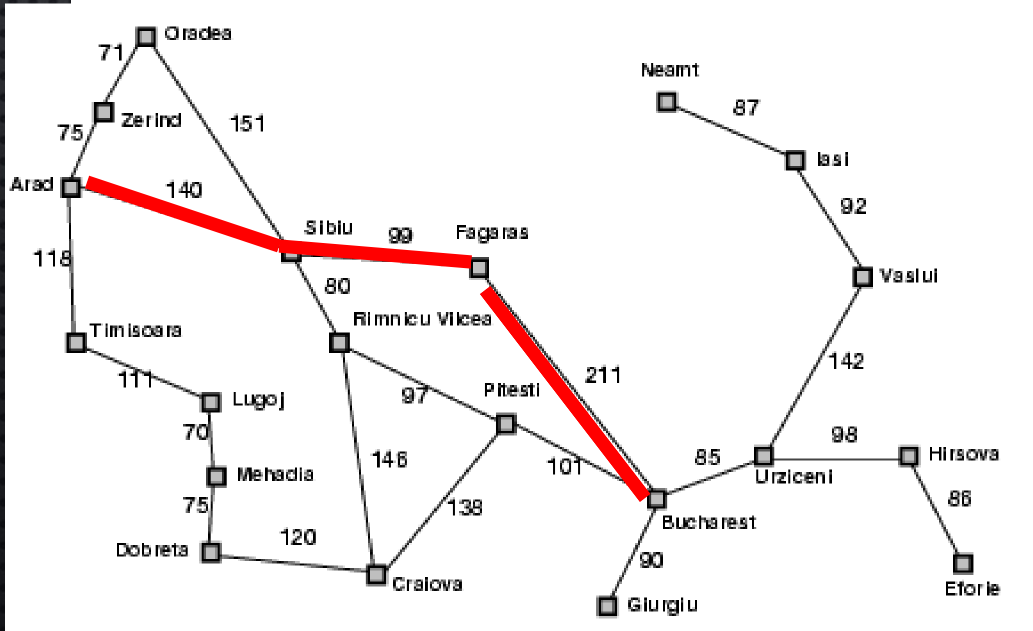


Straight−line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

GREEDY BEST-FIRST SEARCH EXAMPLE

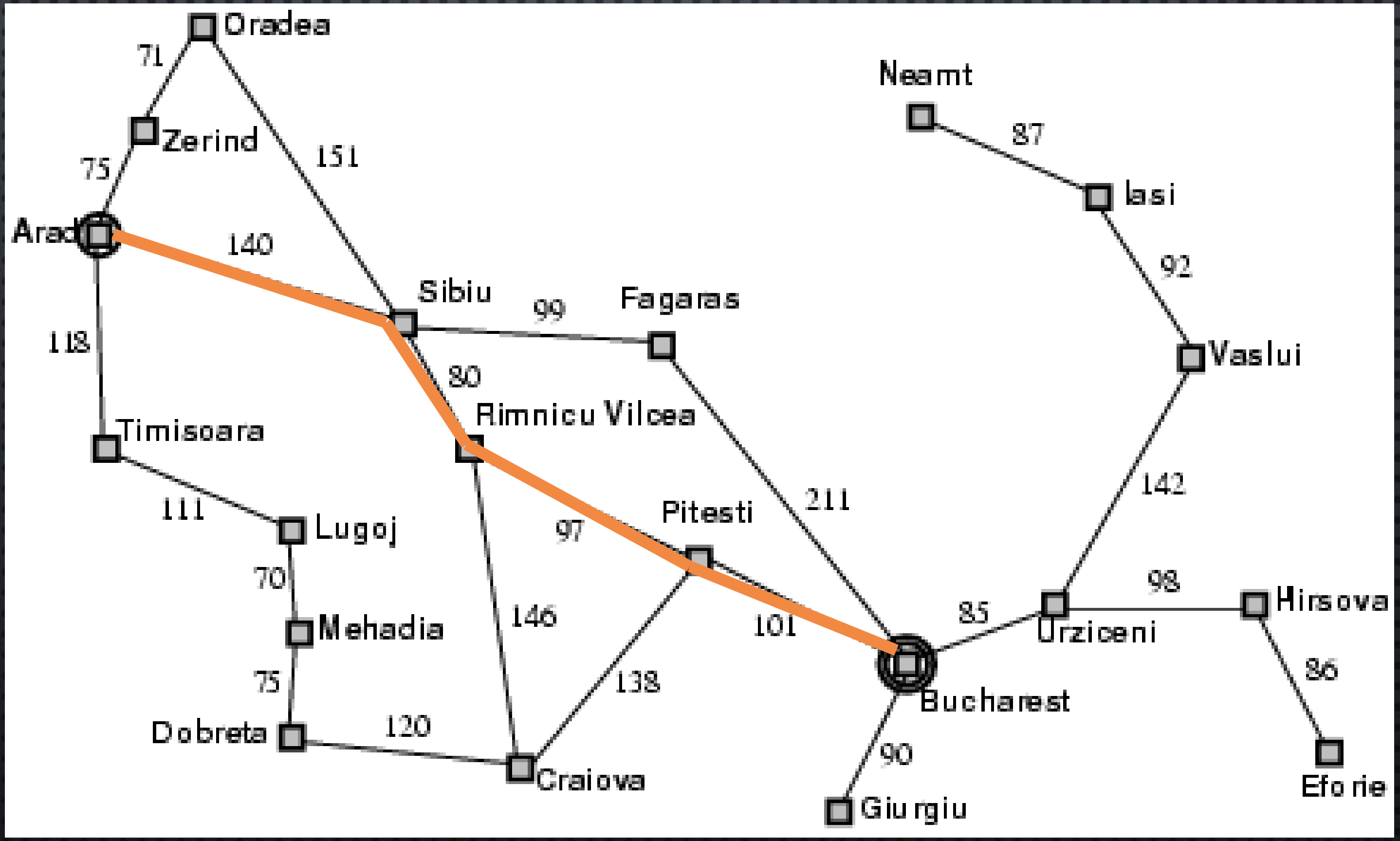




Straight–line distance
to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu V ilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

OPTIMAL PATH



PROPERTIES OF GREEDY BEST-FIRST SEARCH

- COMPLETE?
  - TREE VERSION CAN GET STUCK IN LOOPS.
  - GRAPH VERSION IS COMPLETE IN FINITE SPACES.
- TIME? $O(B^M)$
  - A GOOD HEURISTIC CAN GIVE **DRAMATIC** IMPROVEMENT
- SPACE? $O(B^M)$
  - KEEPS ALL NODES IN MEMORY
- OPTIMAL? NO

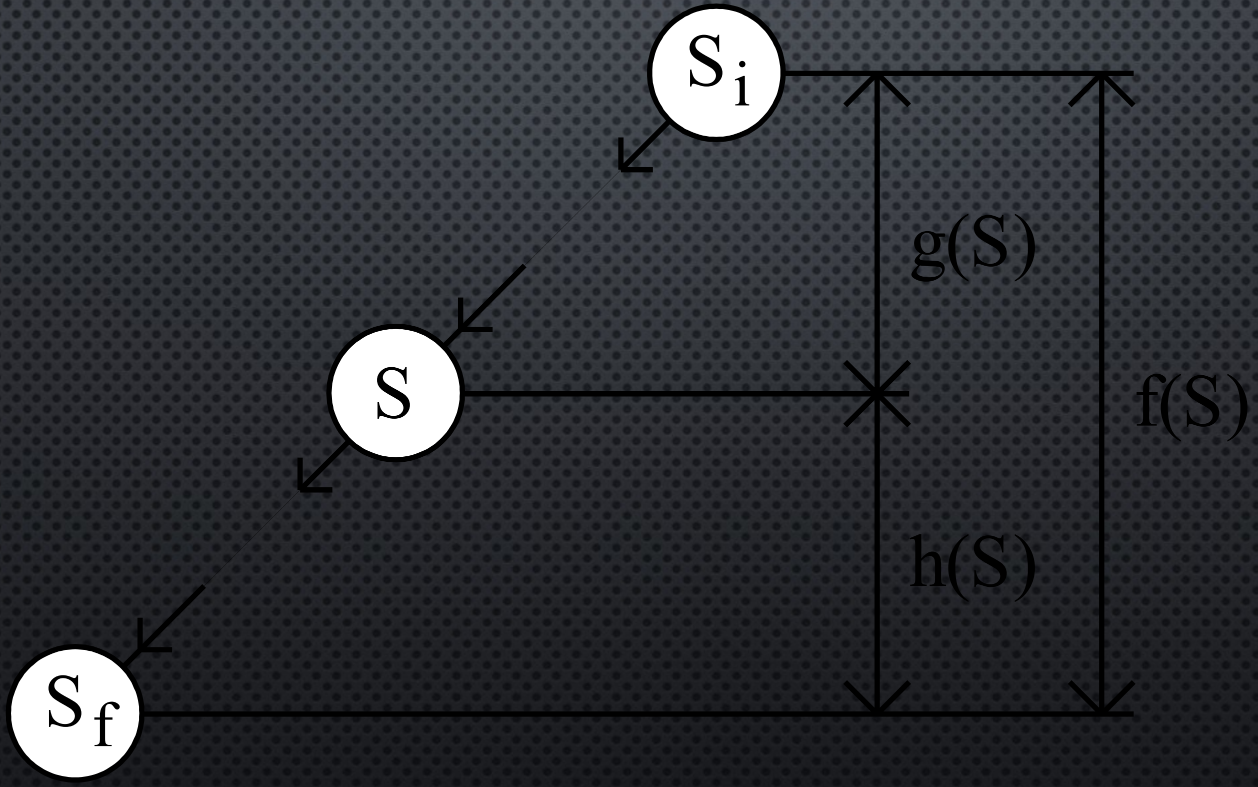
E.G., ARAD → SIBIU → RIMNICU VILCEA → PITESTI → BUCHAREST IS SHORTER!

A* SEARCH

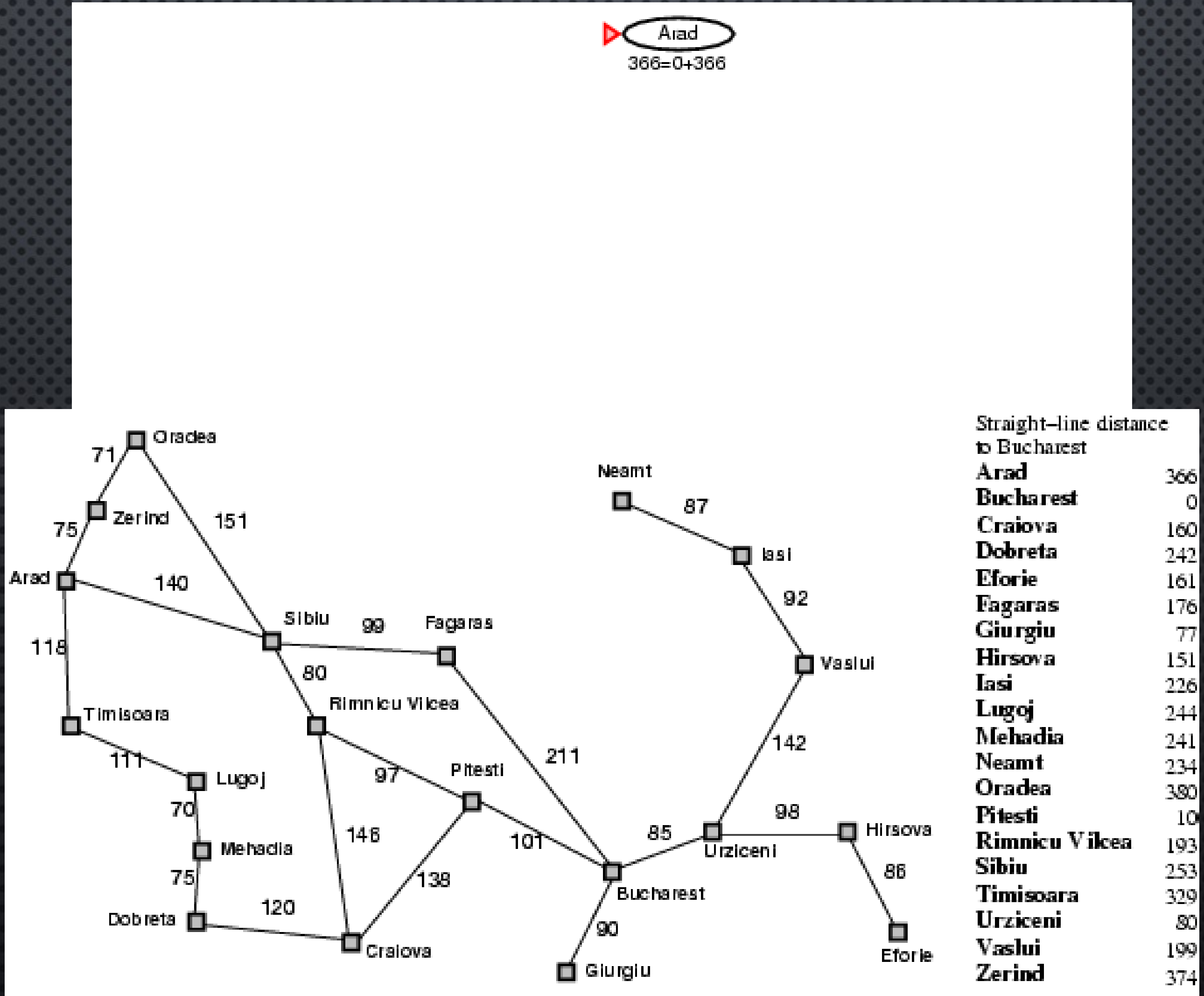
- IDEA: AVOID PATHS THAT ARE ALREADY EXPENSIVE
  - GENERALLY THE PREFERRED SIMPLE HEURISTIC SEARCH
  - OPTIMAL IF HEURISTIC IS: ADMISSIBLE(TREE)/CONSISTENT(GRAPH)
- EVALUATION FUNCTION *F(N) = G(N) + H(N)*
  - G(N) = KNOWN PATH COST SO FAR TO NODE N.
  - H(N) = <u>ESTIMATE</u> OF (OPTIMAL) COST TO GOAL FROM NODE N.
  - F(N) = G(N)+H(N)

    = <u>ESTIMATE</u> OF TOTAL COST TO GOAL THROUGH NODE N.
- *PRIORITY QUEUE SORT FUNCTION = F(N)*

Components of A*



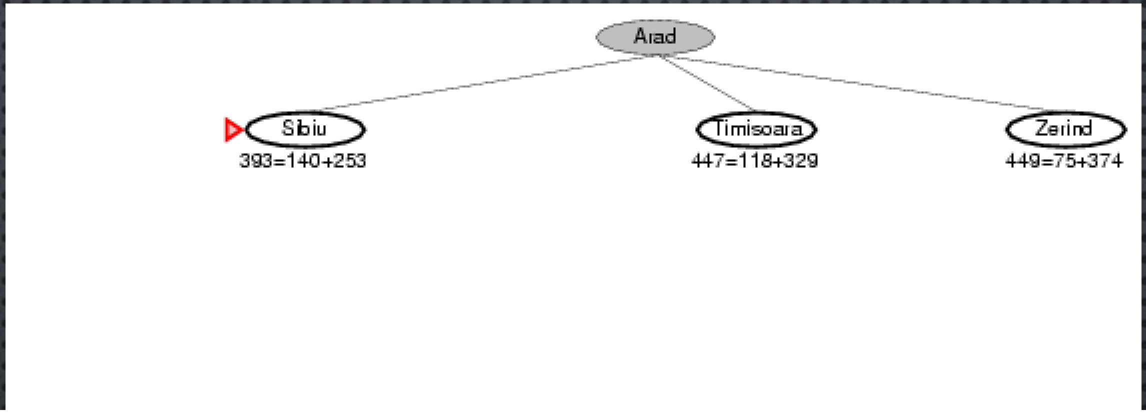
A* SEARCH EXAMPLE

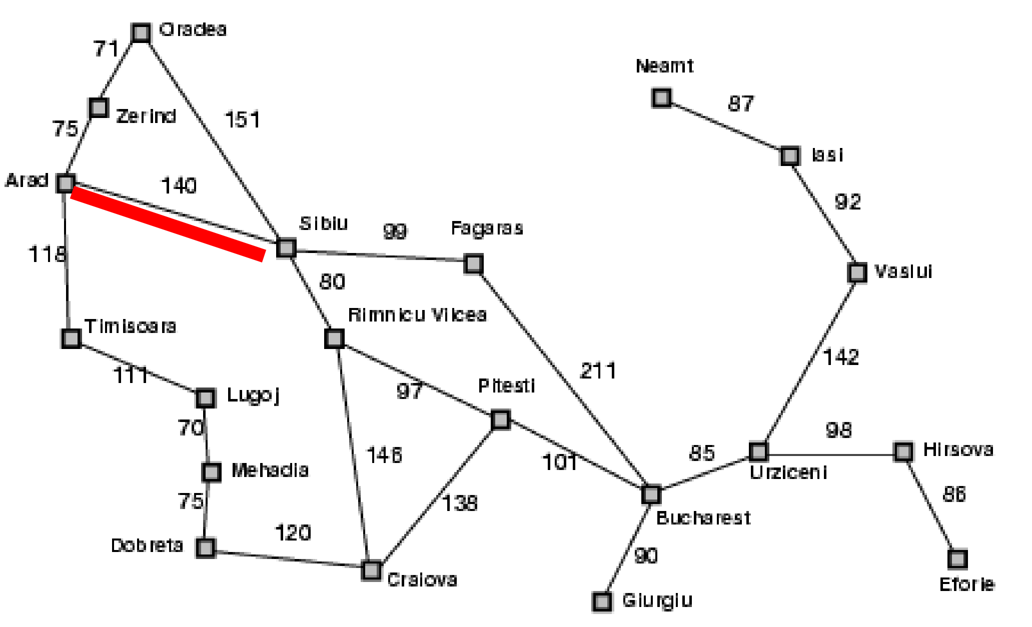


Straight–line distance
to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

A* SEARCH EXAMPLE

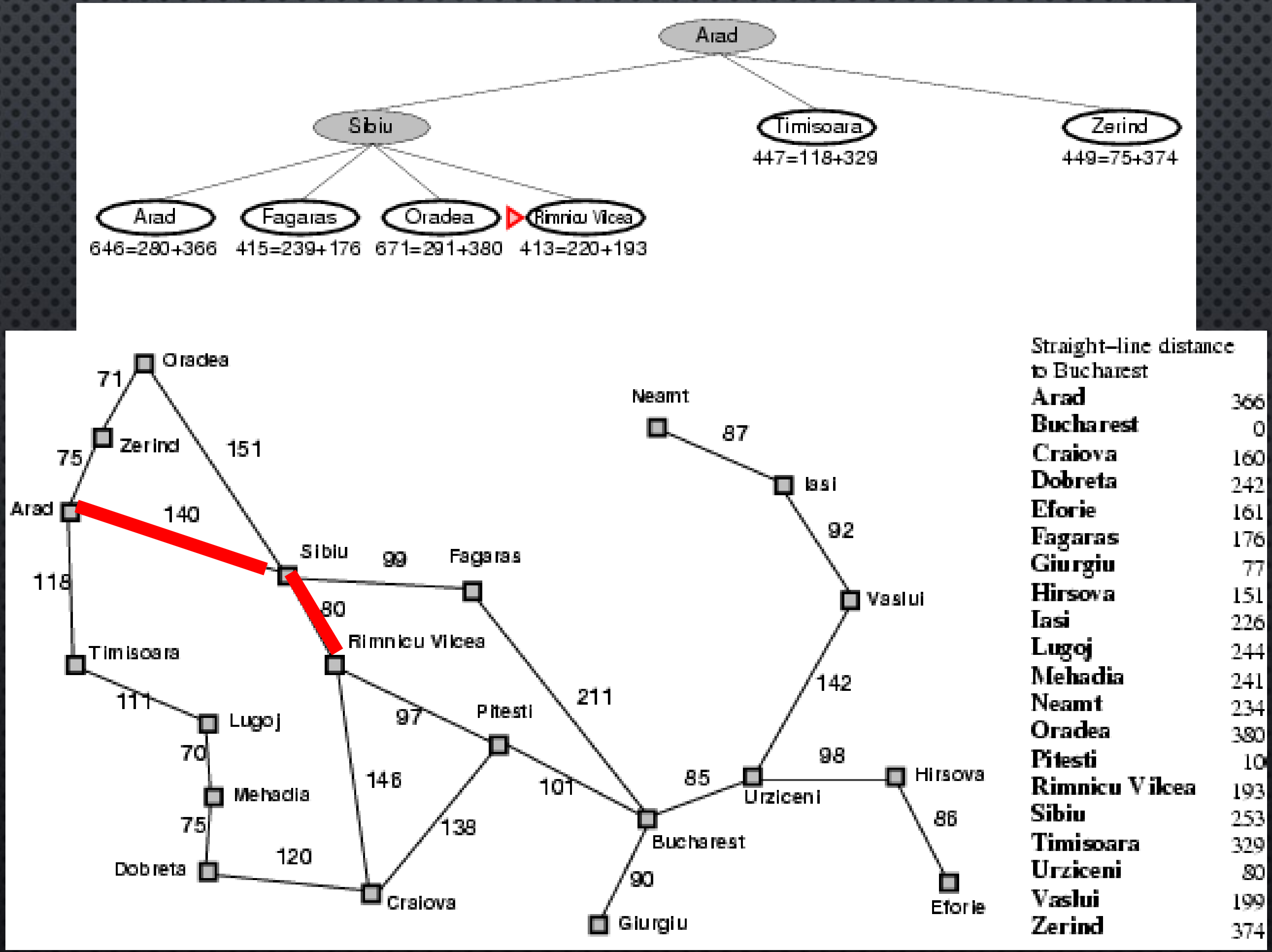




Straight–line distance
to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

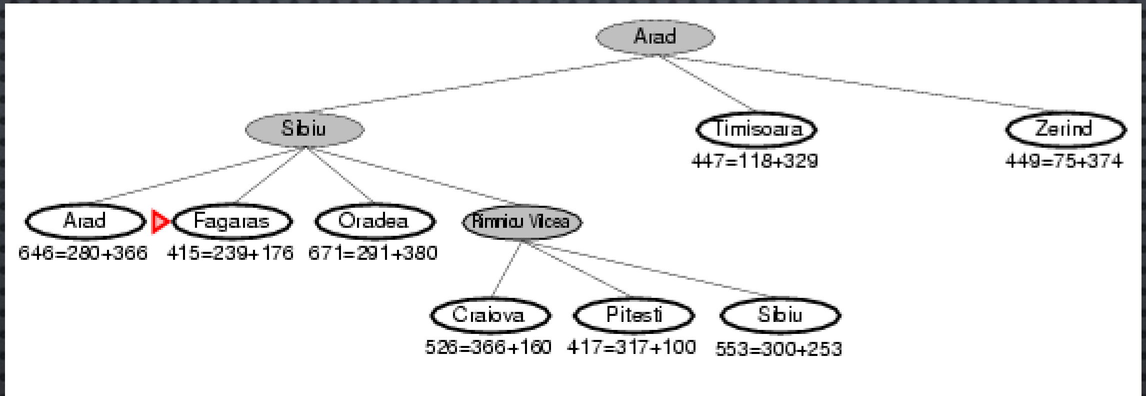
A* SEARCH EXAMPLE

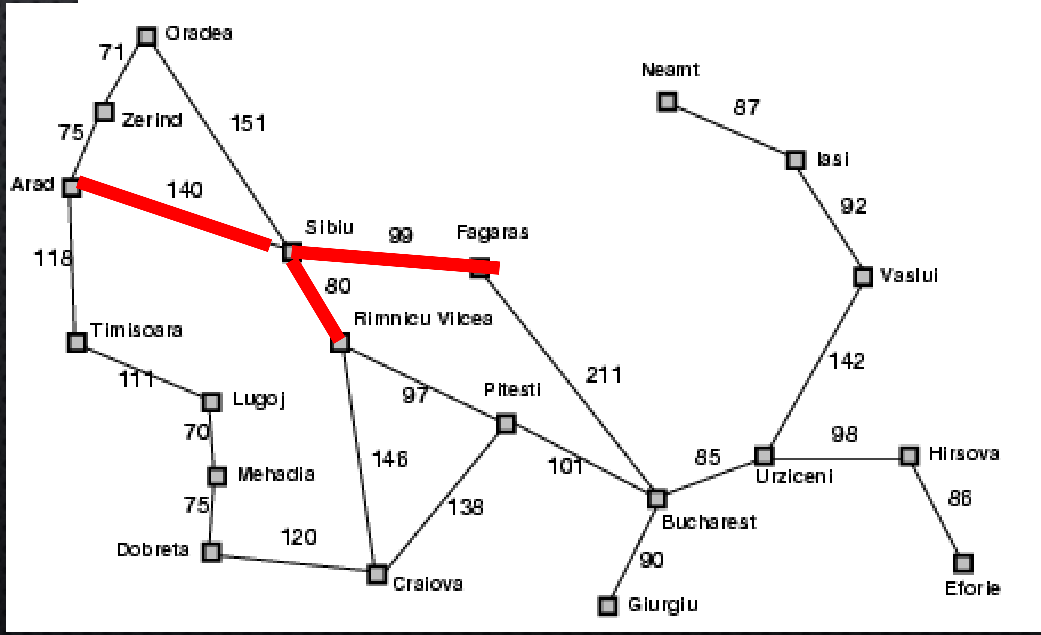


Straight–line distance
to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

A* SEARCH EXAMPLE

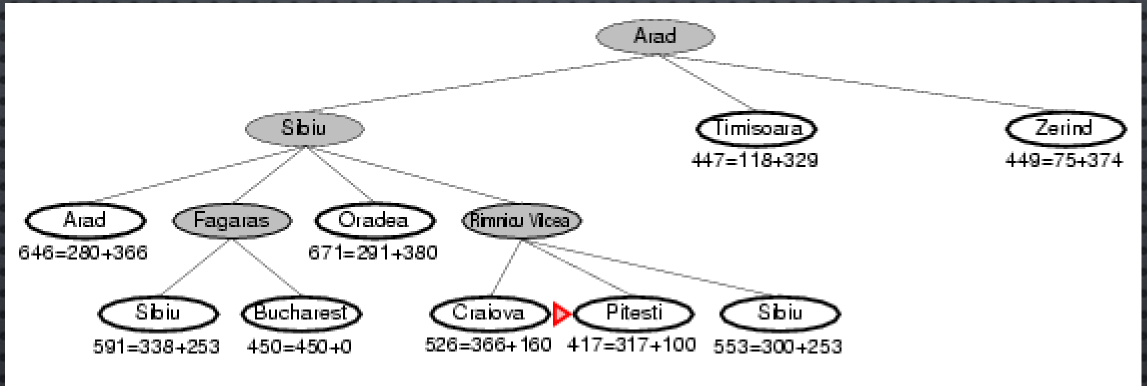


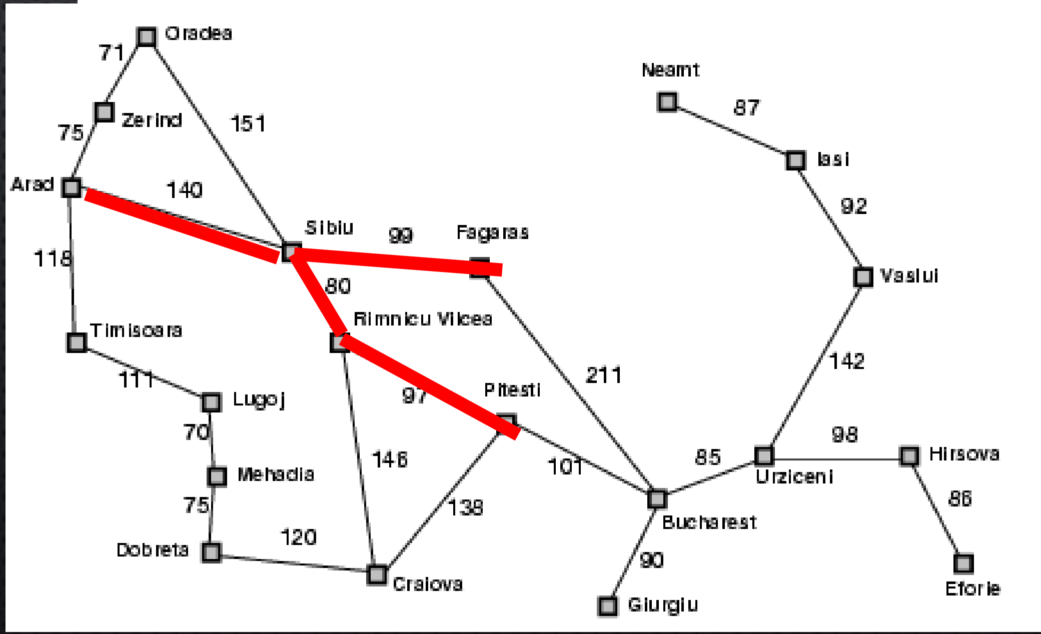


Straight–line distance
to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

A* SEARCH EXAMPLE

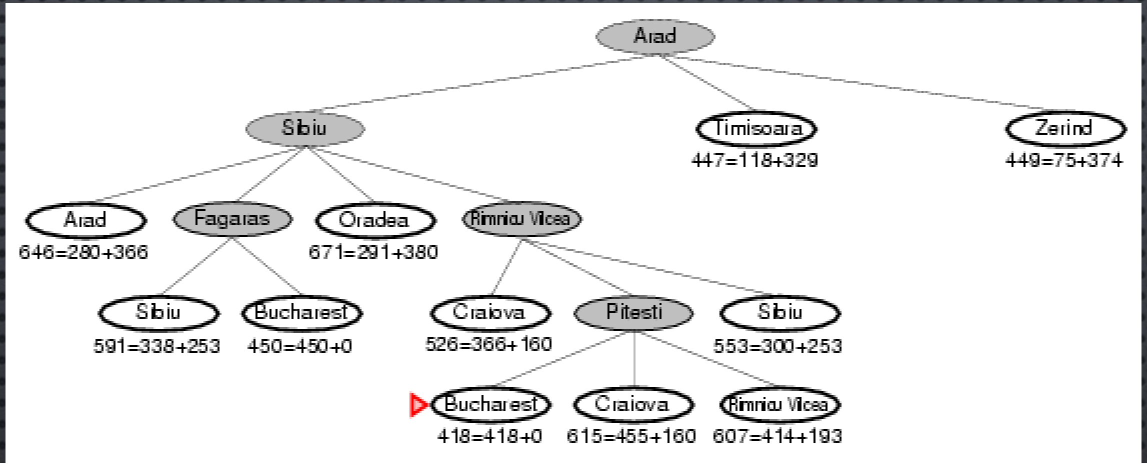


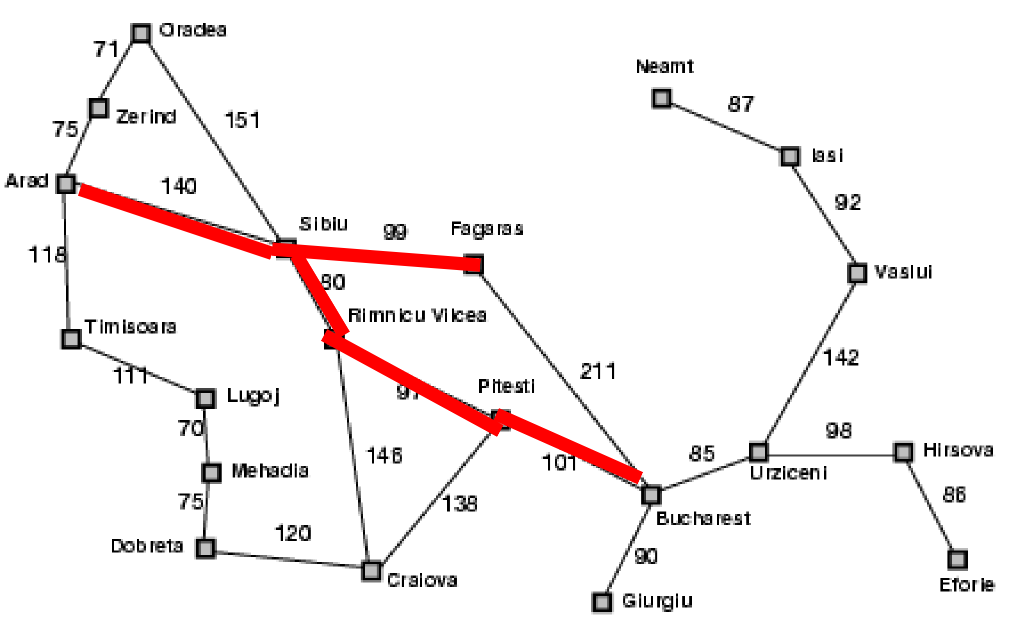


Straight–line distance
to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

A* SEARCH EXAMPLE



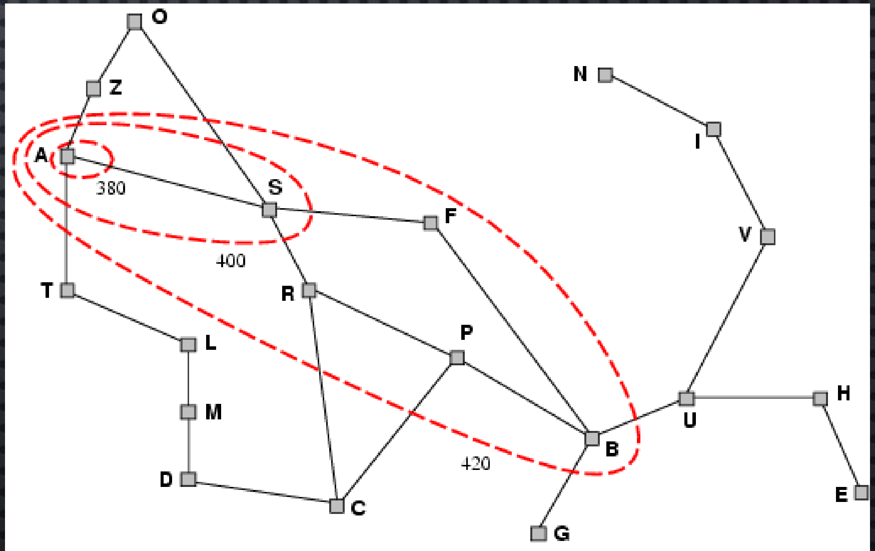


Straight–line distance
to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

CONTOURS OF A* SEARCH

- A* EXPANDS NODES IN ORDER OF INCREASING *F* VALUE
- GRADUALLY ADDS "*F*-CONTOURS" OF NODES
- CONTOUR *I* HAS ALL NODES WITH $F=F_I$, WHERE $F_I < F_{I+1}$



PROPERTIES OF A*

- COMPLETE? YES

  (UNLESS THERE ARE INFINITELY MANY NODES WITH *F ≤ F(G)*; CAN’T HAPPEN IF STEP-COST ≥ E > 0)
- TIME/SPACE? EXPONENTIAL *O(B^{D})*

  EXCEPT IF: $| h(n) - h^{*}(n) | \leq O(\log h^{*}(n))$
- OPTIMAL? YES

  (WITH: TREE-SEARCH, ADMISSIBLE HEURISTIC; GRAPH-SEARCH, CONSISTENT HEURISTIC)
- *OPTIMALLY EFFICIENT?* YES

  (NO OPTIMAL ALGORITHM WITH SAME HEURISTIC IS GUARANTEED TO EXPAND FEWER NODES)

MULTIMODAL HUD FOR AUTO UI

CONTEXT

ISSUES WITH DASHBOARD

- DRIVERS TAKE EYES OFF FROM ROAD WHILE OPERATING DASHBOARD
- DASHBOARD REQUIRES PHYSICAL TOUCH

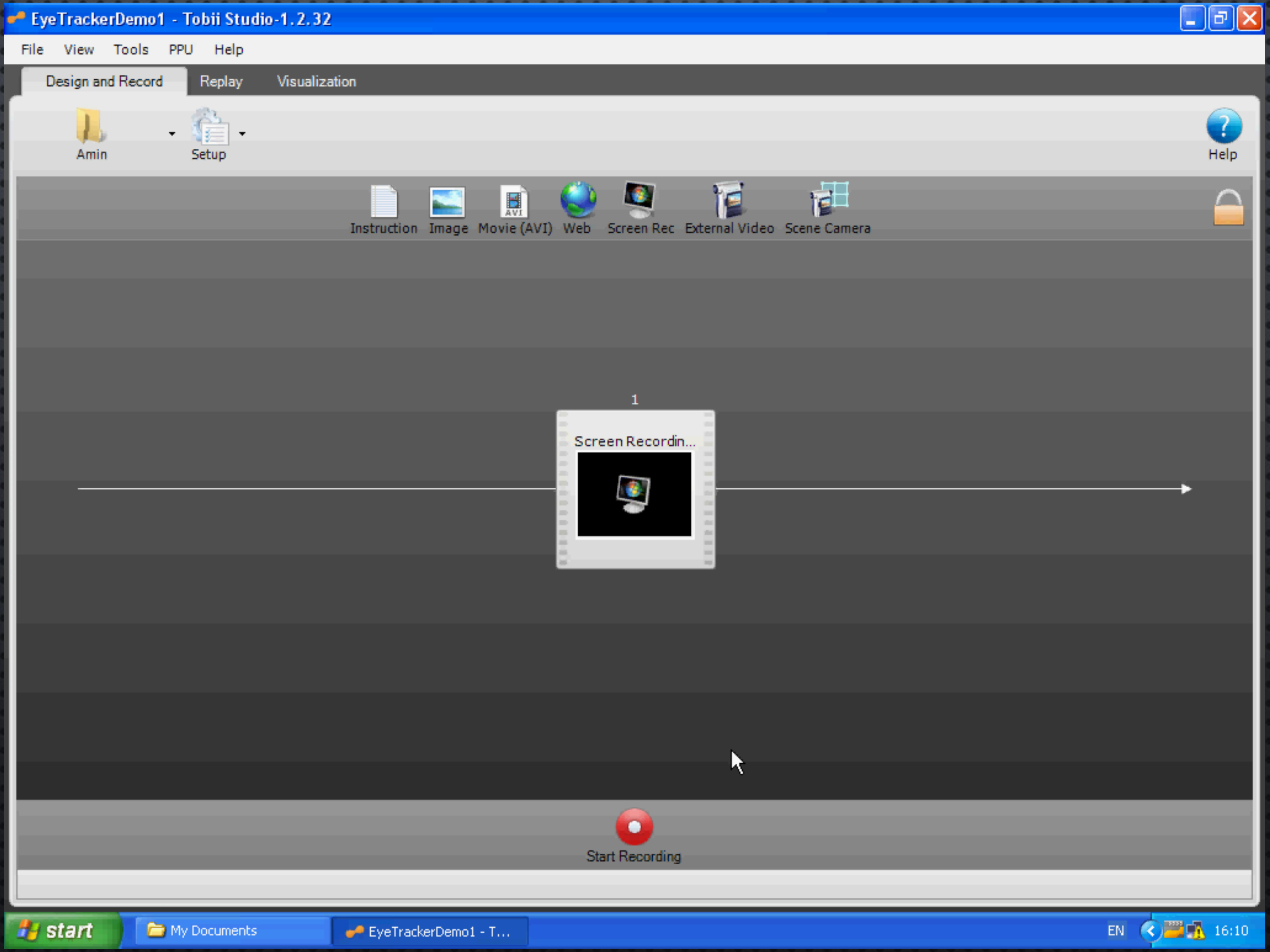
STATE OF THE ART

- **DIRECT VOICE INPUT**
- **GESTURE RECOGNITION SYSTEM**
- **HAND/FINGER MOVEMENT TRACKER**

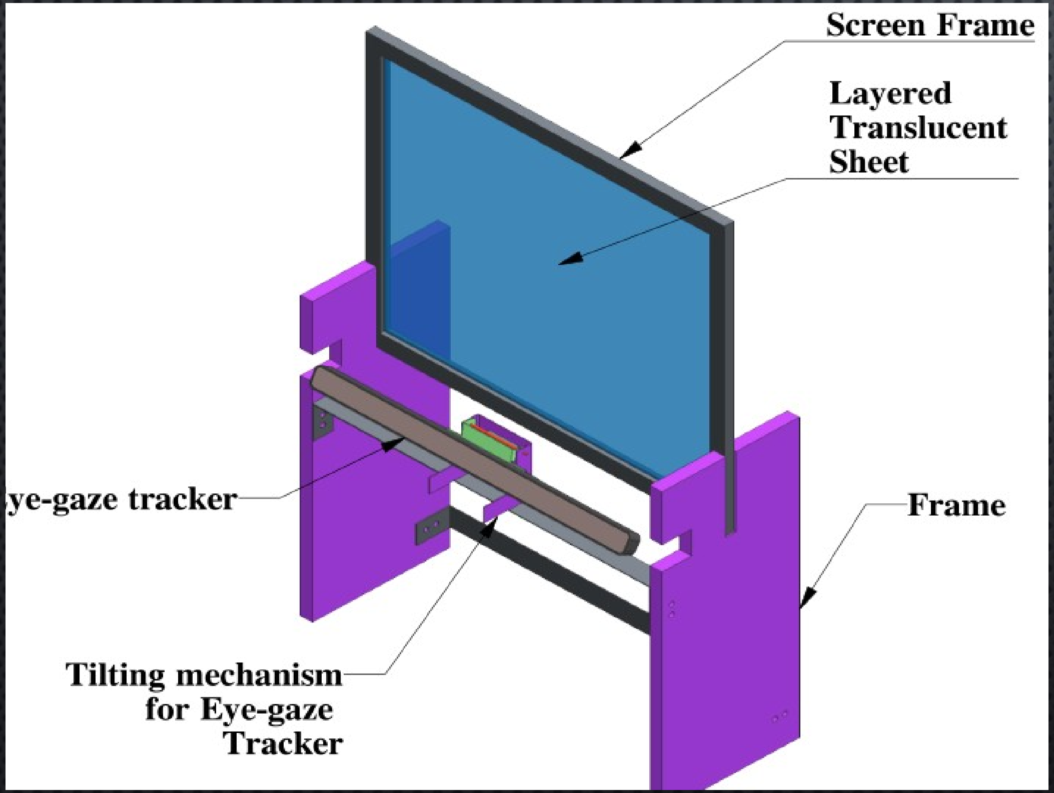
EXISTING PROBLEM

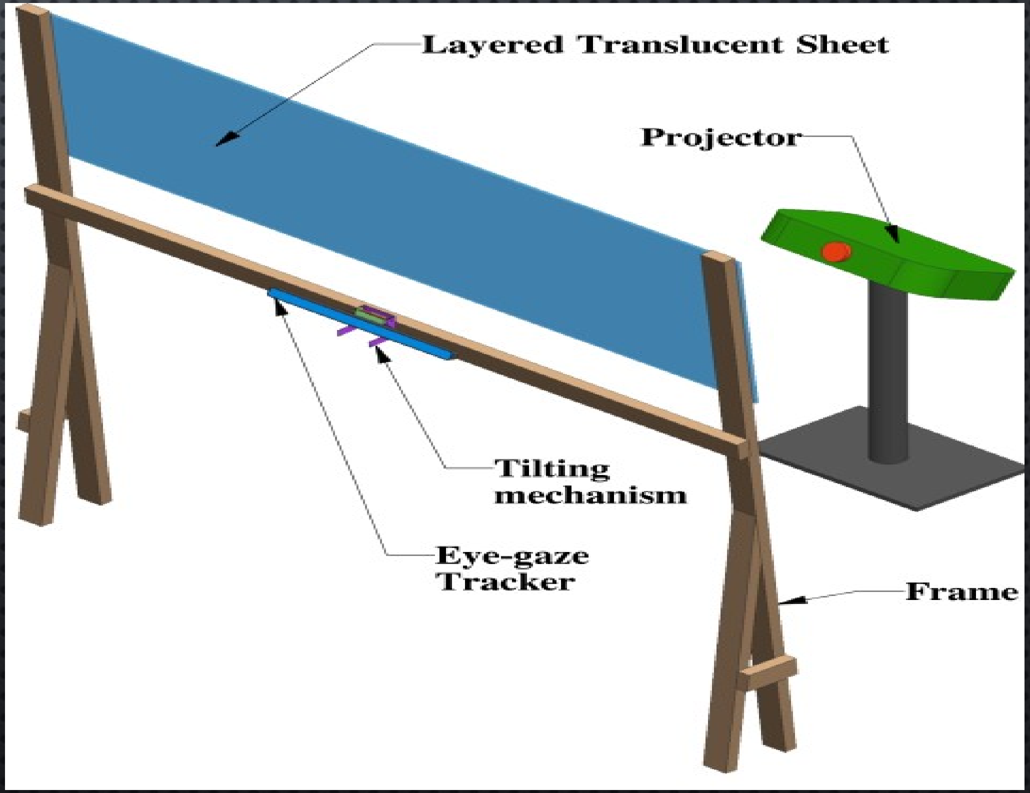
- ACCURACY OF DVI CHANGES FOR DIFFERENT LANGUAGES AND AFFECTIVE STATE
- NEED TO REMEMBER A SET OF GESTURES OR SCREEN SEQUENCE
- INTELLIGENT PREDICTION ALGORITHM CAN NOT IMPROVE LATENCY IN INFRARED SENSOR

EYE MOVEMENT

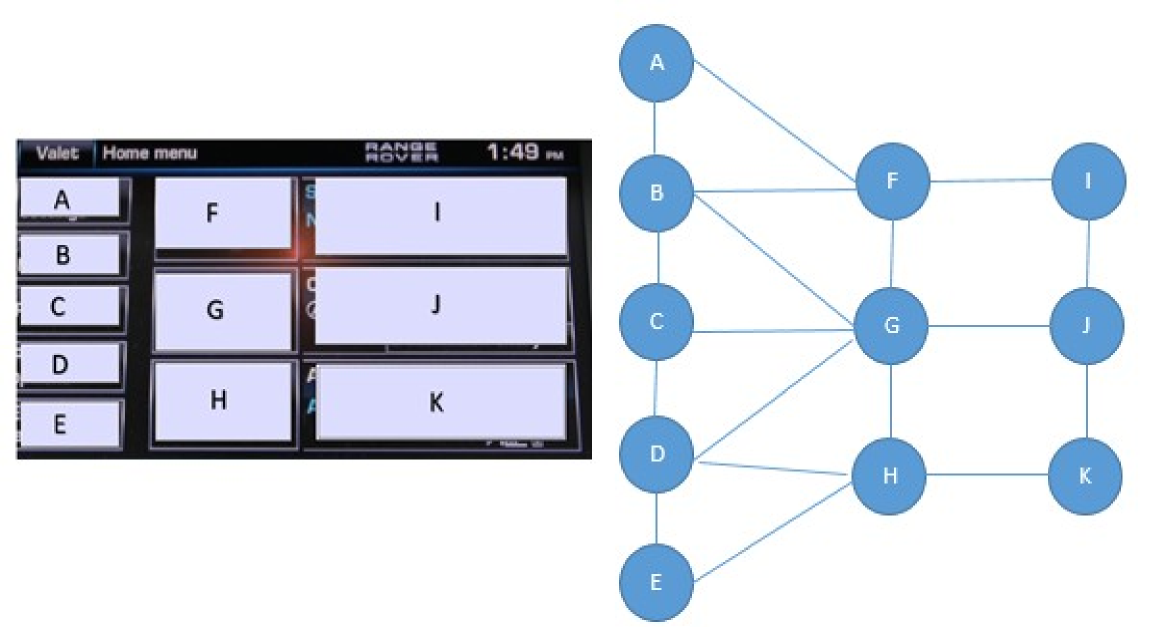


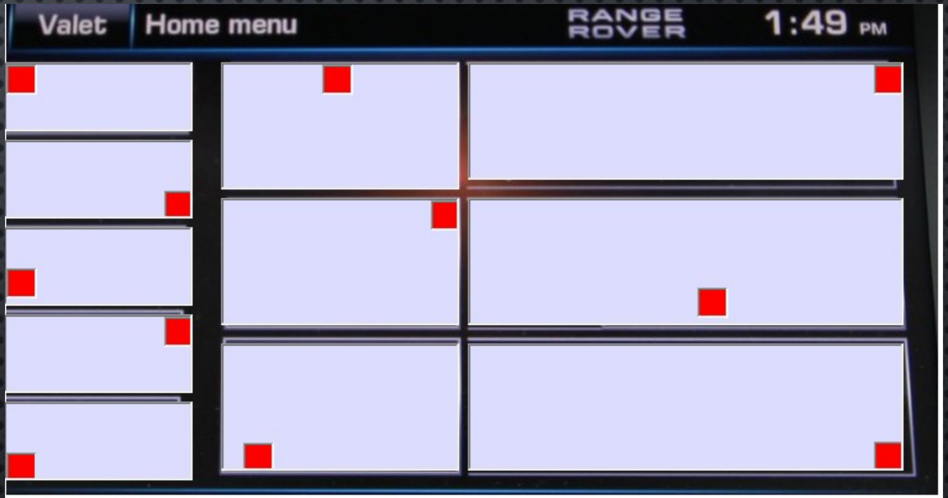
EYE GAZE CONTROLLED PROJECTED DISPLAY


Screen Frame
Layered Translucent Sheet
ye-gaze tracker
Frame
Tilting mechanism for Eye-gaze Tracker



IMPROVING GAZE TRACKING – HOT SPOTS

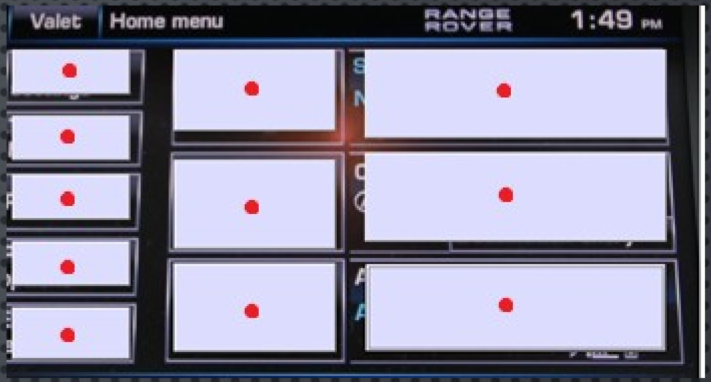




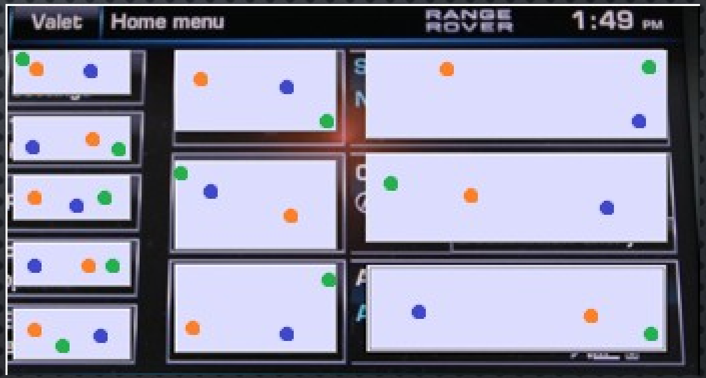
P. Biswas, Interactive Gaze Controlled Projected Display, Patent Application No.: 201641037828

STATE SPACE SEARCH

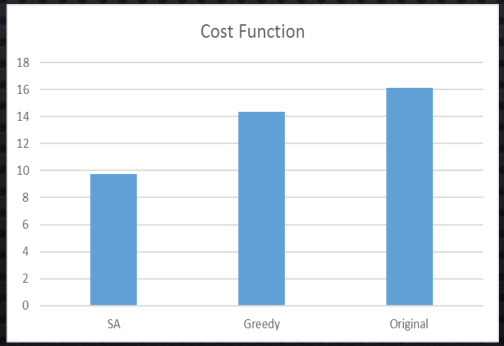
- A STATE SPACE CONSISTS OF
  - A (POSSIBLY INFINITE) SET OF STATES
    - THE START STATE REPRESENTS THE INITIAL PROBLEM
    - EACH STATE REPRESENTS SOME CONFIGURATION REACHABLE FROM THE START STATE
    - SOME STATES MAY BE GOAL STATES (SOLUTIONS)
- A SET OF OPERATORS
    - APPLYING AN OPERATOR TO A STATE TRANSFORMS IT TO ANOTHER STATE IN THE STATE SPACE

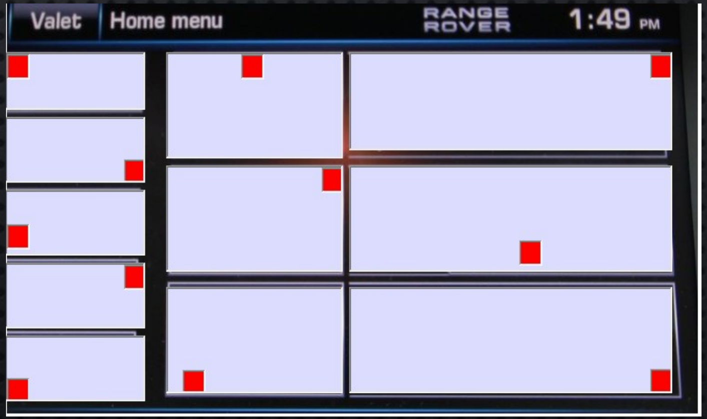


$\sum_{\forall d_{ij}} \frac{1}{d_{ij}^2}$



Each set of colored dots represents a state

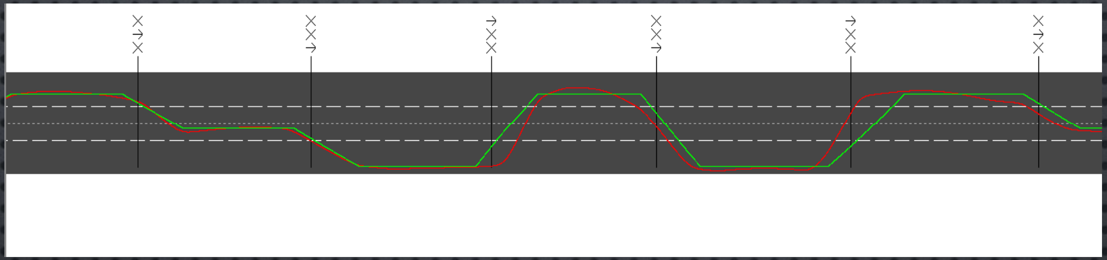




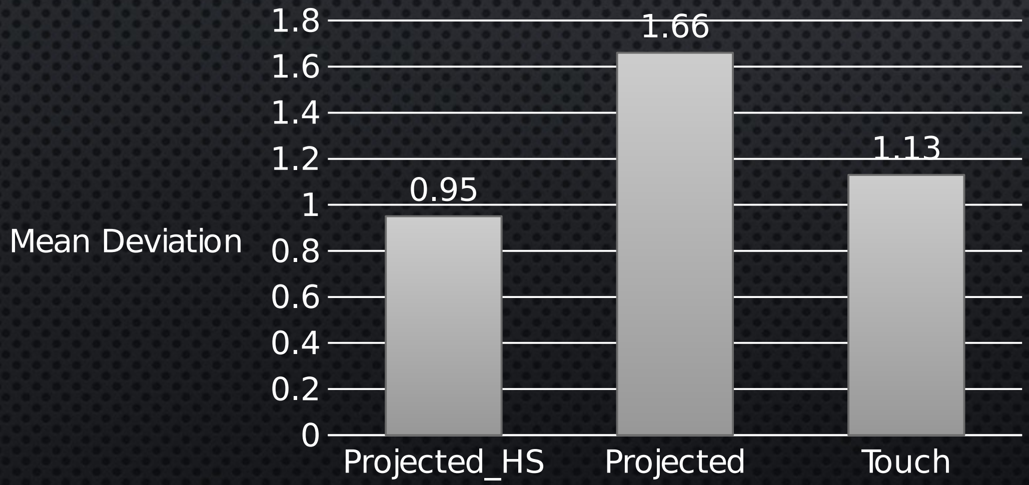
USER STUDY

- COLLECTED DATA FROM 11 PARTICIPANTS, ALL WITH DRIVING EXPERIENCE
- DRIVING TASK INVOLVED ISO 26022 LANE CHANGING TASK
- SECONDARY TASK INVOLVED POINTING AND SELECTION ON A DASHBOARD
  - USING TOUCH SCREEN
  - GAZE TRACKING HUD
  - GAZE TRACKING HUD WITH HOTSPOTS

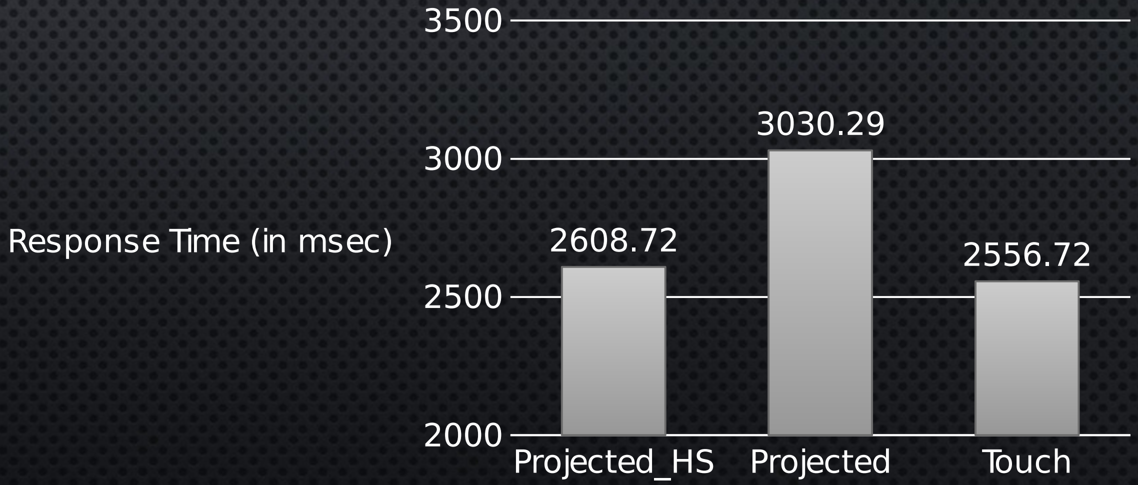
RESULTS FROM DUAL TASK STUDY



Mean Deviation from Lane



Response Time



SUMMARY

- PROPOSED A MULTIMODAL HUD FOR AUTOMOTIVE ENVIRONMENT
- DRIVERS NEED NOT TO TAKE EYES OFF FROM THE ROAD FOR THE PROPOSED SYSTEM
- CAN BE OPERATED AS FAST AS A TOUCHSCREEN WITH IMPROVED DRIVING PERFORMANCE
- PROPOSED A NEW ALGORITHM TO IMPROVE RESPONSE TIMES FOR GAZE CONTROLLED INTERACTIVE SYSTEMS

TAKE AWAY POINTS

- STATE SPACE SEARCH
- UNINFORMED SEARCH
  - BREADTH FIRST
  - DEPTH FIRST
  - IDA
  - BIDIRECTIONAL
- INFORMED SEARCH
  - BEST FIRST
  - A*
- COMPARING SEARCH ALGORITHMS
- CASE STUDY ON IUI
  - FORMULATE STATE SPACE
  - IMPLEMENT
  - EVALUATE