BITS Pilani

Pilani Campus

Artificial and Computational Intelligence

AIMLCZG557

Contributors & Designers of document content : Cluster Course Faculty Team

M2 : Problem Solving Agent using Search

Presented by

Faculty Name

BITS Email ID

Artificial and Computational Intelligence

Disclaimer and Acknowledgement

Disclaimer

- Few content for these slides may have been obtained from prescribed books and various other source on the Internet
- I hereby acknowledge all the contributors for their material and inputs and gratefully acknowledge people others who made their course materials freely available online.
- .I have provided source information wherever necessary
- This is not a full fledged reading materials. Students are requested to refer to the textbook w.r.t detailed content of the presentation deck that is expected to be shared over e-learning portal - taxilla.
- I have added and modified the content to suit the requirements of the class dynamics & live session's lecture delivery flow for presentation

- **Slide Source / Preparation / Review:**
- From BITS Pilani WILP**:** Prof.Raja vadhana, Prof. Indumathi, Prof.Sangeetha
- From BITS Oncampus & External : Mr.Santosh GSK

BITS Pilani, Pilani Campus

Course Plan

M1 Introduction to AI

M2 Problem Solving Agent using Search

M3 Game Playing

M4 Knowledge Representation using Logics

M5 Probabilistic Representation and Reasoning

M6 Reasoning over time

M7 Ethics in AI

Learning Objective

At the end of this class , students Should be able to:

1. Create Search tree for given problem
2. Differentiate between uninformed and informed search requirements
3. Apply GBFS & A* algorithms to the given problem
4. Prove if the given heuristics are admissible and consistent
5. Apply A* variations algorithms to the given problem

Module 2 : Problem Solving Agent using Search

A. Uninformed Search

B. Informed Search

C. Heuristic Functions

D. Local Search Algorithms & Optimization Problems

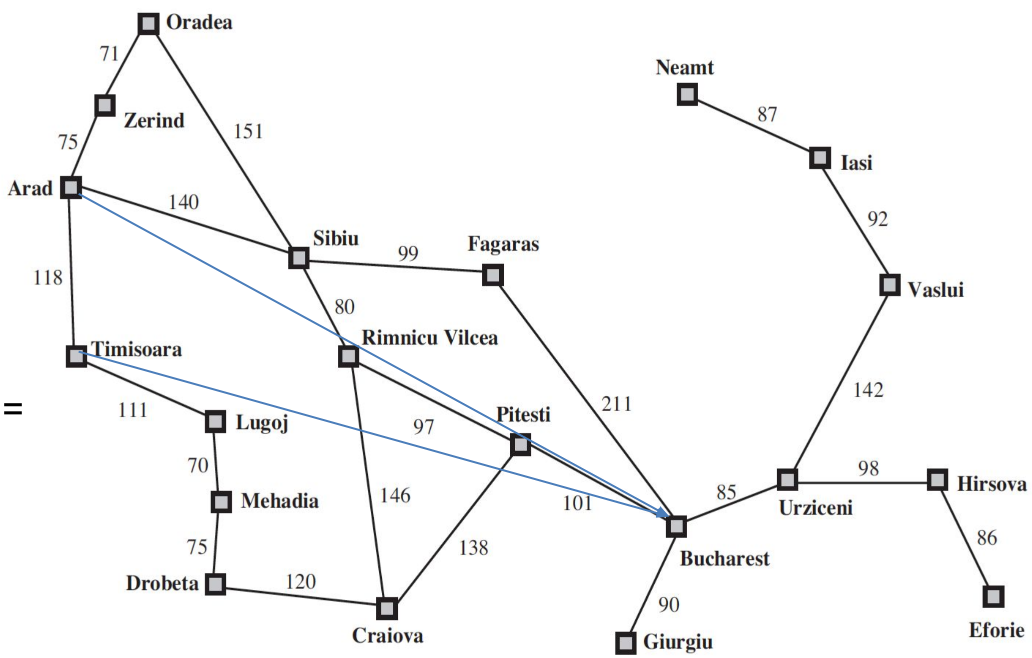
Informed Search
Greedy Best First
A*

Informed /Heuristic Search

Strategies that know if one non-goal state is more promising than another non-goal state

h(Arad) = 366

h(Timisoara) = 329

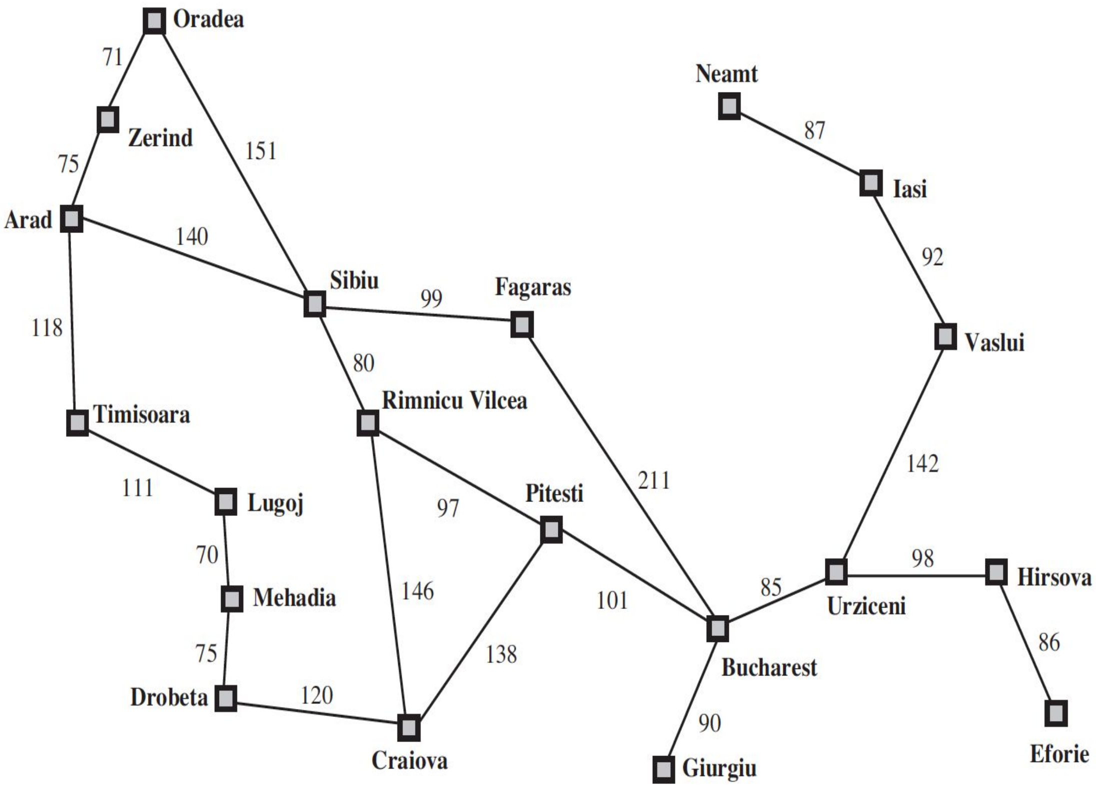


Greedy Best First Search

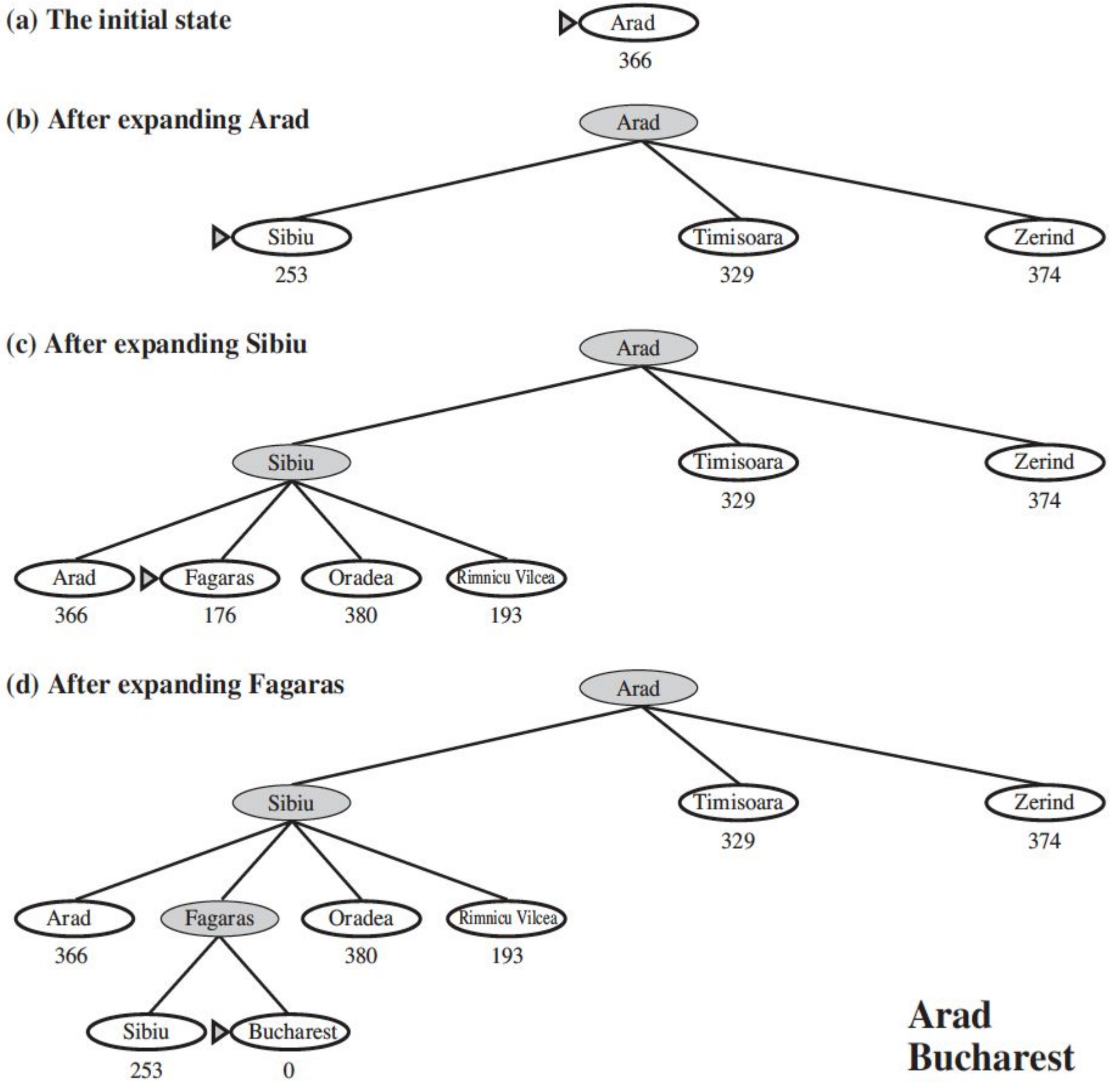
Expands the node that is closest to the goal

Thus, f(n) = h(n)

Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374



Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374



Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Drobeta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374

Greedy Best First Search

Not Optimal

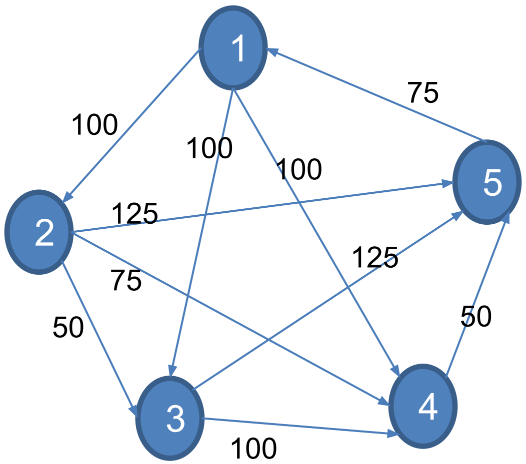
- Because the algorithm is greedy
- It only optimizes for the current action

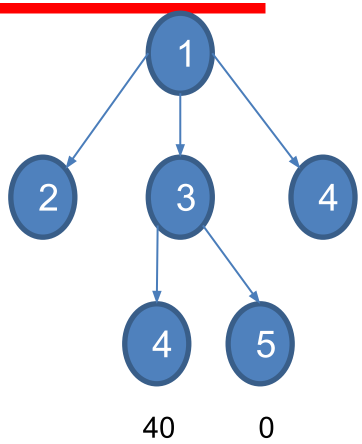
Not Complete

- Often ends up in state with a dead end as the heuristic doesn't guarantee a path but is only an approximation

Time and Space Complexity - $\mathcal{O}(b^m)$ where m – max depth of search tree

Greedy Best First Search



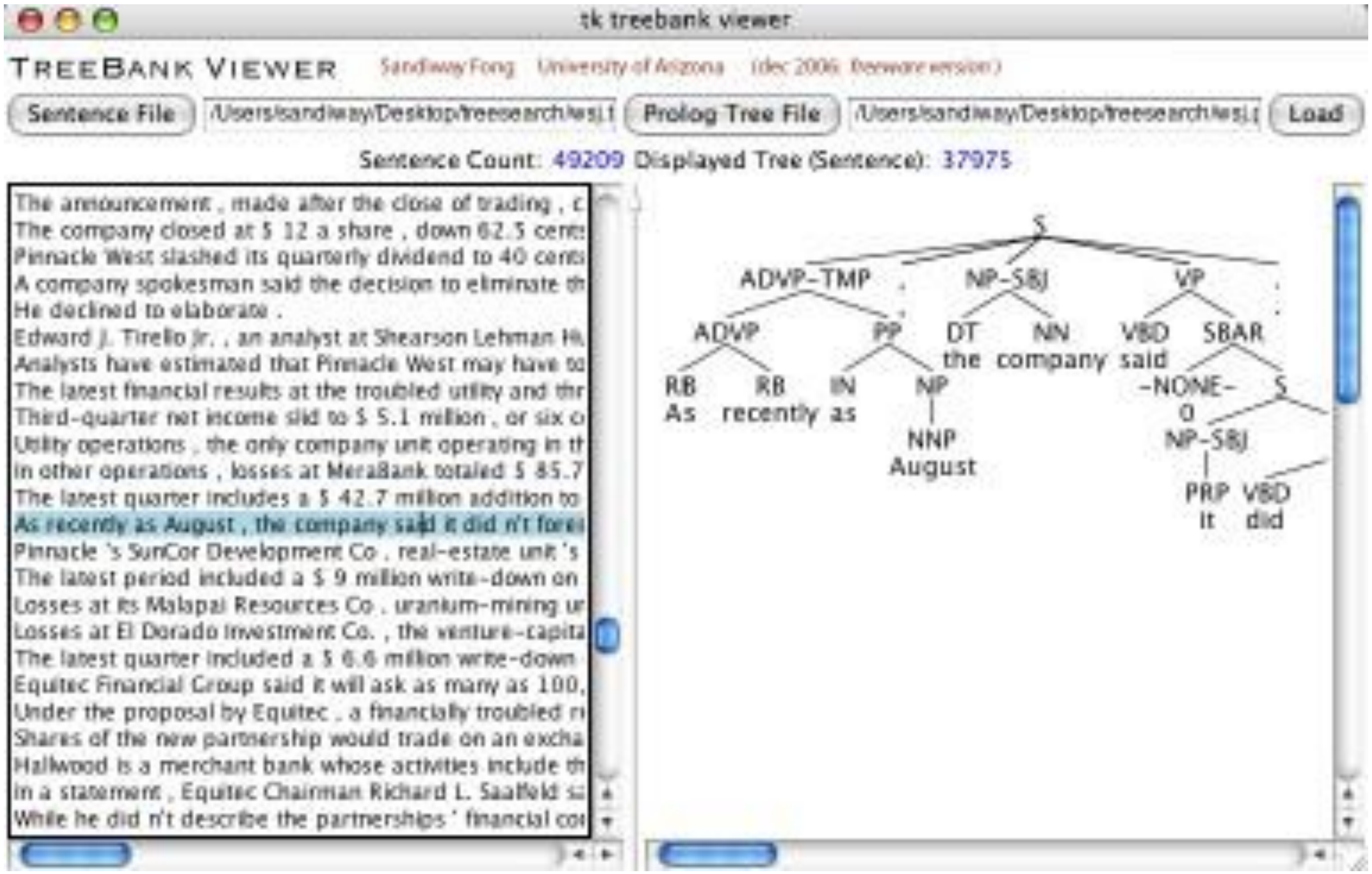


n	h(n)
1	60
2	120
3	30
4	40
5	0

(1)
(1 3) (1 4) (1 2)
(1 3 5) (1 3 4)

C(1-3-5) = 100 + 125 =225
Expanded : 2
Generated : 6
Max Queue Length : 3
Idea: Optimize DFS. Choose next nearest to goal in the same hill.

Case Study – 1 Search in Treebanks



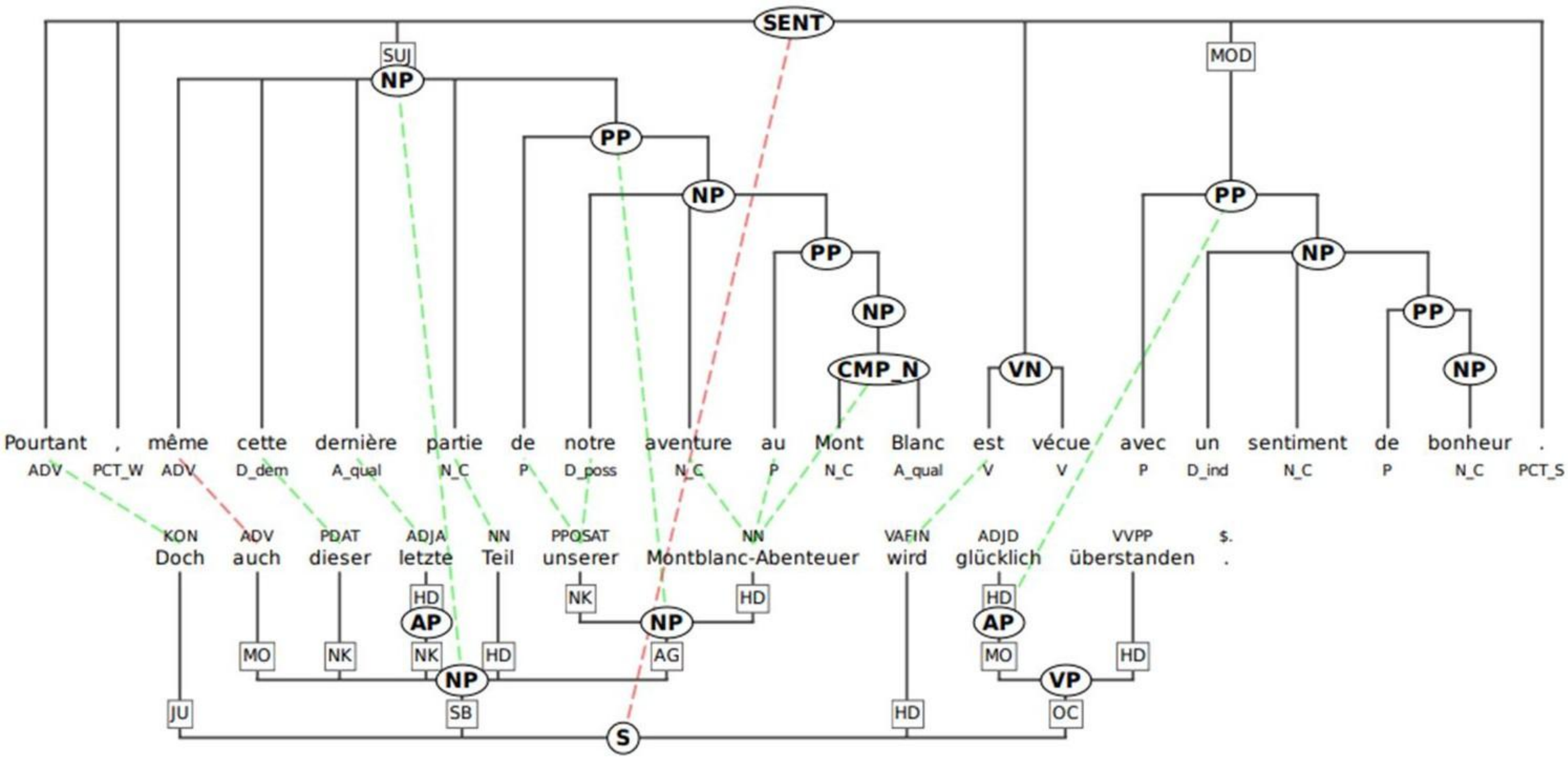
Source Credit :

https://catalog.ldc.upenn.edu/docs/LDC95T7/cl93.html

https://ufal.mff.cuni.cz/pdt3.5

BITS Pilani, Pilani Campus

Case Study – 1 Search in Treebanks



Source Credit :

https://catalog.ldc.upenn.edu/docs/LDC95T7/cl93.html

https://ufal.mff.cuni.cz/pdt3.5

BITS Pilani, Pilani Campus

A* Search

Expands the node which lies in the closest path (estimated cheapest path) to the goal

Evaluation function **f(n) = g(n) + h(n)**

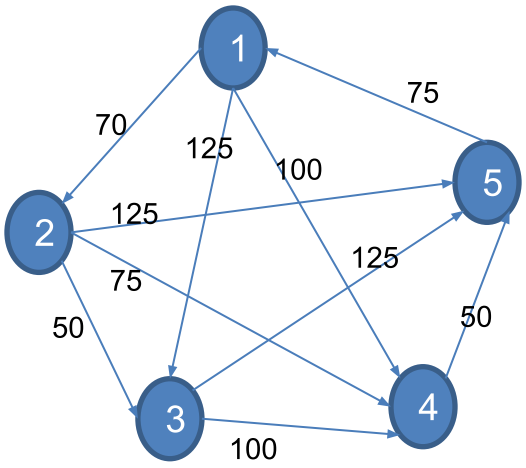
g(n) – the cost to reach the node

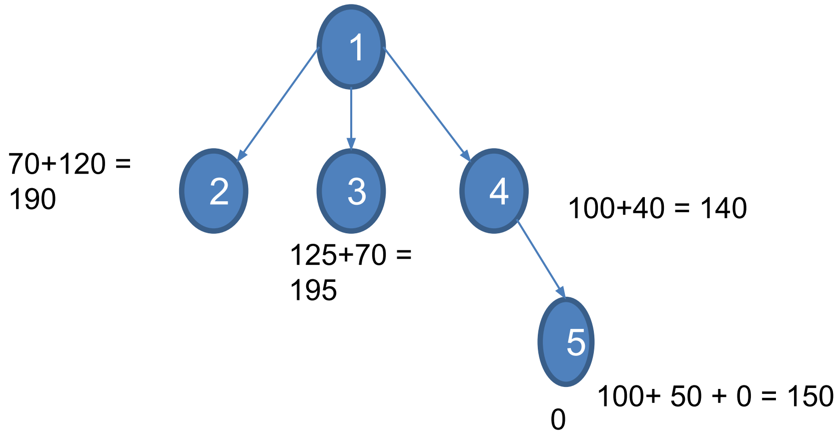
h(n) – the expected cost to go from node to goal

f(n) – estimated cost of cheapest path through node n

Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Drobeta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374

A* Search





n	h(n)
1	60
2	120
3	70
4	40
5	0

(1)
(1 4) (1 2) (1 3)
(1 4 5) (1 2) (1 3)

C(1-4-5) = 100 + 150 =150
Expanded : 2
Generated : 5
Max Queue Length : 3

Optimality of A*

Test for Admissibility

Expands the node which lies in the closest path (estimated cheapest path) to the goal

Evaluation function **f(n) = g(n) + h(n)**

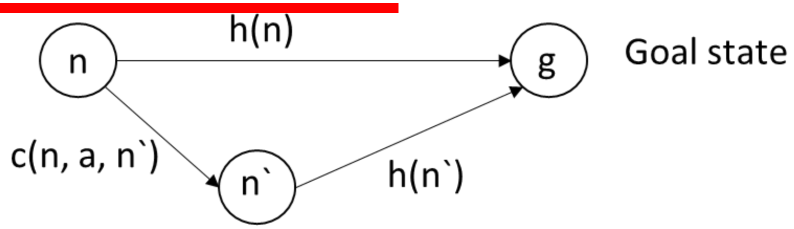
g(n) – the cost to reach the node

h(n) – the expected cost to go from node to goal

f(n) – estimated cost of cheapest path through node n

A heuristic is admissible or optimistic if , $\mathbf{0 \leq h(n) \leq h^*(n)}$, where $h^*(n)$ is the actual cost to reach the goal

A* Search



Optimal on condition

h(n) must satisfies two conditions:

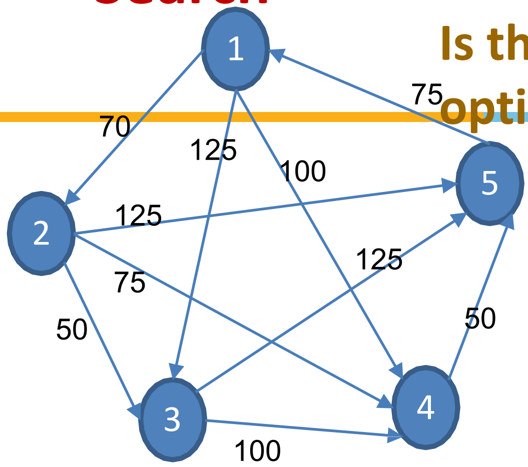
- Admissible Heuristic – one that never overestimates the cost to reach the goal
- Consistency – A heuristic is consistent if for every node n and every successor node n` of n generated by action a,   h(n) <= c(n, a, n`) + h(n`)

Complete

- If the number of nodes with cost <=C* is finite
- If the branching factor is finite
- A* expands no nodes with f(n) > C*, known as pruning

Time Complexity - $\mathcal{O}\left(b^{\Delta}\right)$ where the absolute error $\Delta = h^{*} - h$

A* Search



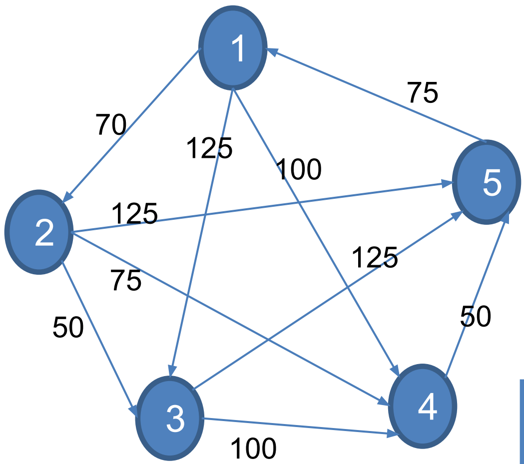
Is theheuristicdesigned leads to optimalsolution?

Assuming **node 3** as goal, taking only sample edges
per node below is checked for consistency

		n	h(n)	Is Admissible? h(n) <= h*(n)	Is Consistent? For every arc (i,j): h(i) <= g(i,j) + h(j)
		1	80		
		2	60		
		3	0		
		4	200		
		5	190		

A* Search

Is the heuristic designed leads to optimal solution?



Assuming node 3 as goal, taking only sample edges per node below is checked for consistency

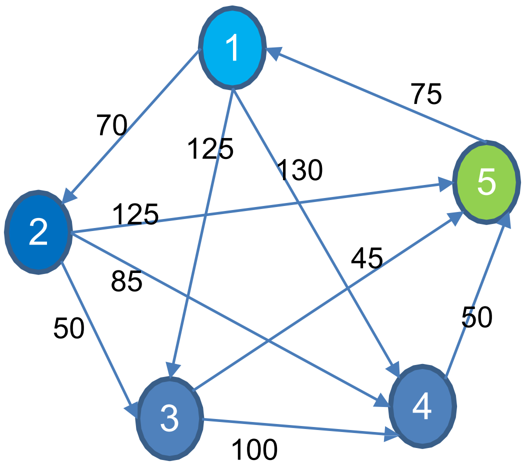
n	h(n)	Is Admissible? h(n) <= h*(n)	Is Consistent? For every arc (i,j): h(i) <= g(i,j) + h(j)
1	80	Y	N (5□1) : 190<=155
2	60	N	Y (1□2) : 80<=130
3	0	Y	
4	200	Y	Y (1□4) : 80<=300 Y (2□4) : 60<=275
5	190	Y	Y (2□5) : 60<=315 Y (4□5) : 200<=240

Variations of A*

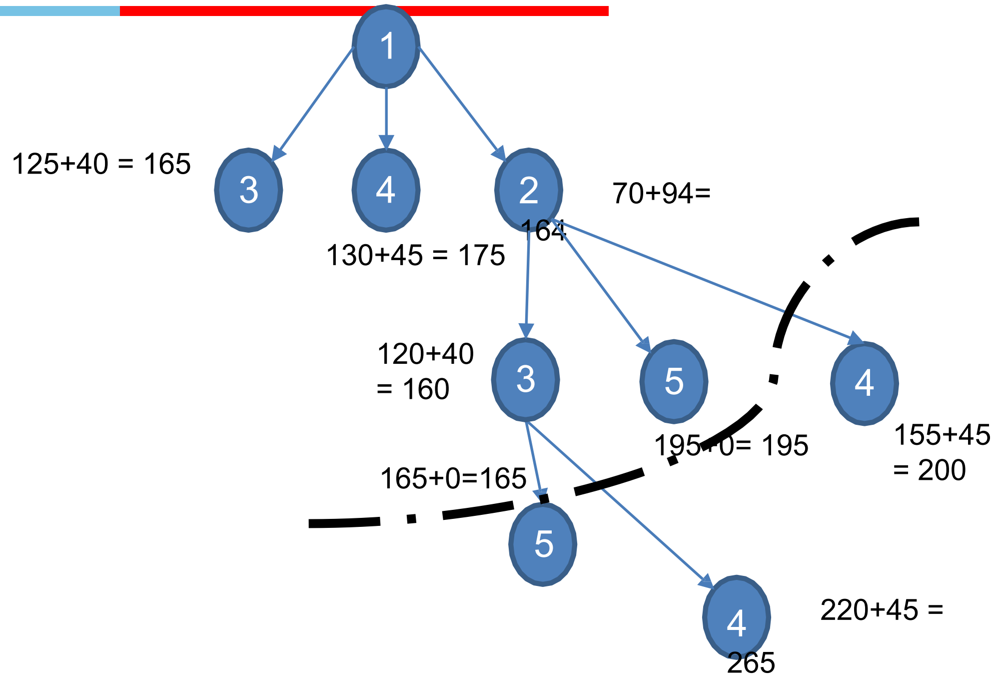
Memory Bounded Heuristics

Iterative Deepening A*

Set limit for f(n)



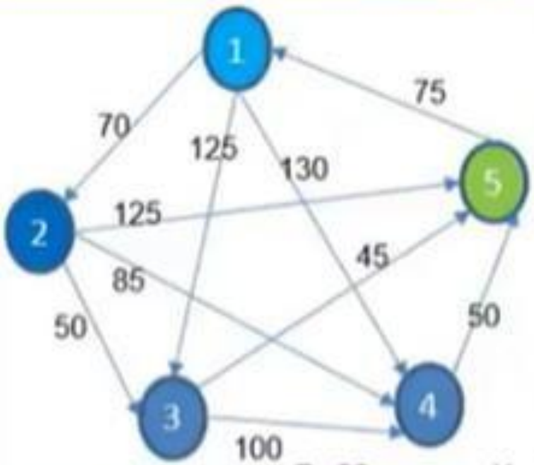
n	h(n)
1	60
2	94
3	40
4	45
5	0

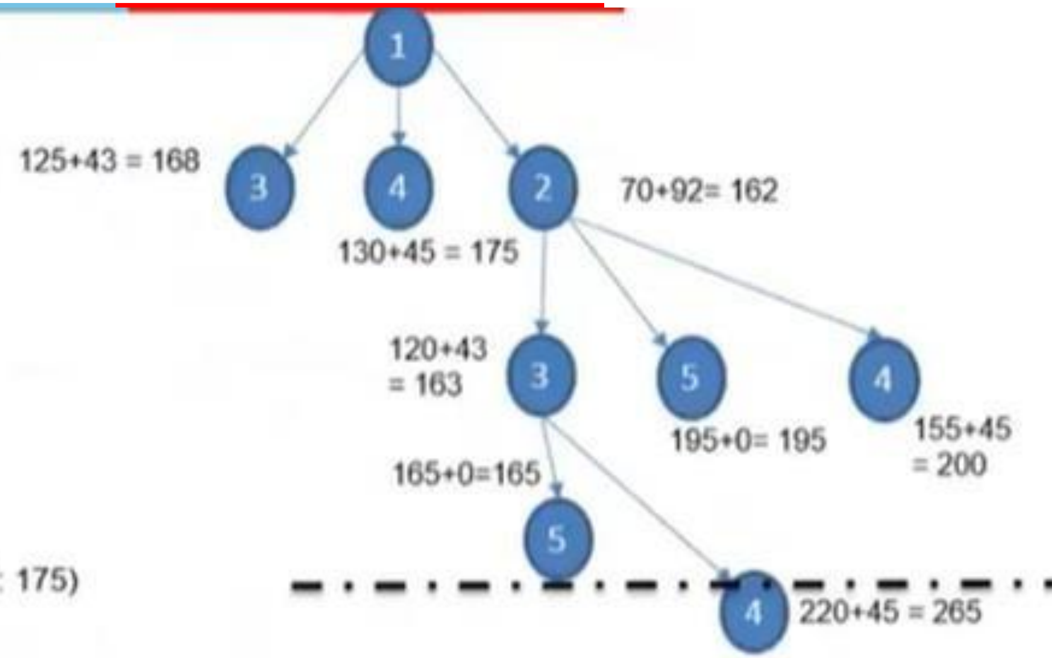


Cut off value is the smallest of f-cost of any node that exceeds the cutoff on previous iterations

Iterative Limit : Eg
f(n) = 180
f(n) = 195
f(n) = 200
.
.
.
.

Iterative Deepening A*





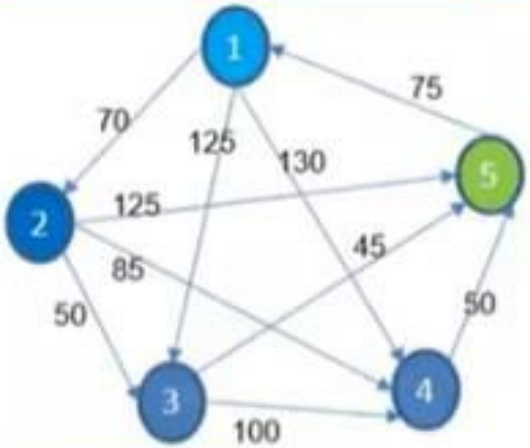
n	h(n)
1	60
2	92
3	43
4	45
5	0

B=60
(1: 60)
TEST-F
(1 2: 162) (1 3: 168) (1 4: 175)

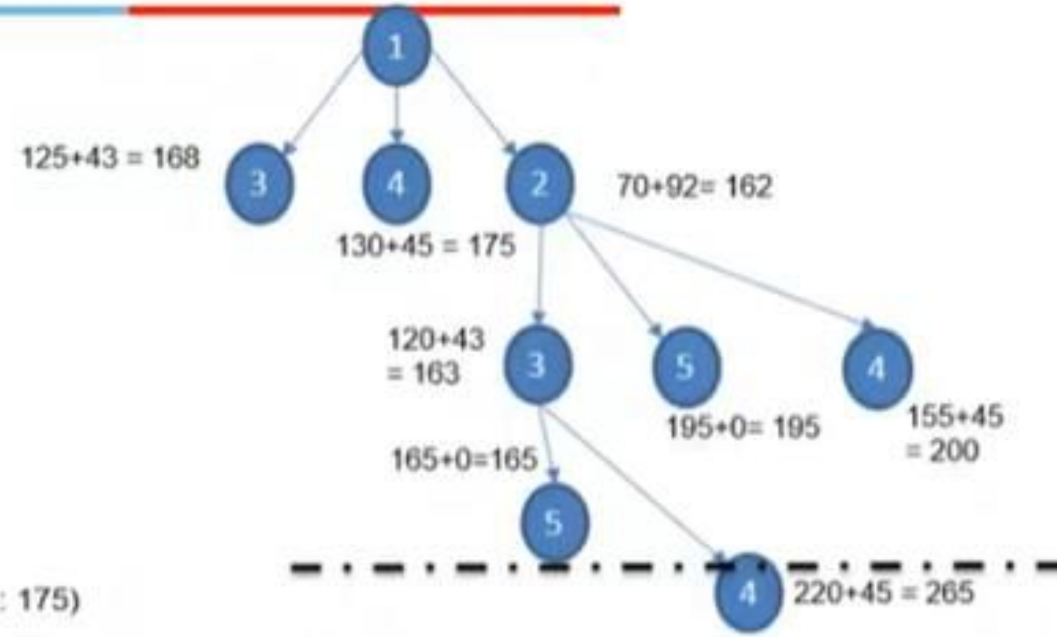
B=162
(1: 60)
TEST-F
(1 2: 162) (1 3: 168) (1 4: 175)
TEST-F
(1 2 3: 163) (1 2 4: 200) (1 2 5: 195) (1 3: 168) (1 4: 175)

B=163
(1: 60)
TEST-F
(1 2: 162) (1 3: 168) (1 4: 175)
TEST-F
(1 2 3: 163) (1 2 4: 200) (1 2 5: 195) (1 3: 168) (1 4: 175)
TEST-F

Iterative Deepening A*



Set limit for f(n)



n	h(n)
1	60
2	92
3	43
4	45
5	0

B=163

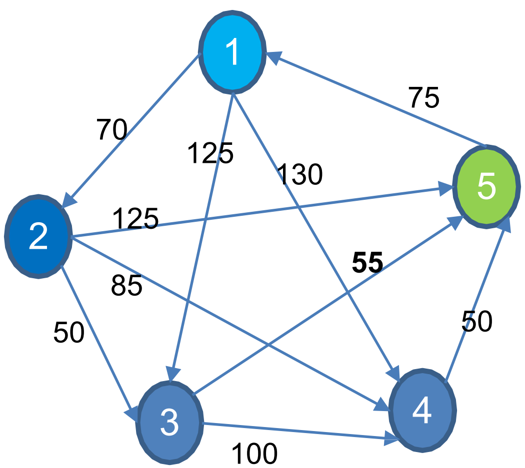
(1: 60)
TEST-F
(1 2: 162) (1 3: 168) (1 4: 175)
TEST-F
(1 2 3: 163) (1 2 4: 200) (1 2 5: 195) (1 3: 168) (1 4: 175)
TEST-F
(1 2 3 5: 165) (1 2 3 4: 265) (1 2 4: 200) (1 2 5: 195) (1 3: 168) (1 4: 175)

B=165

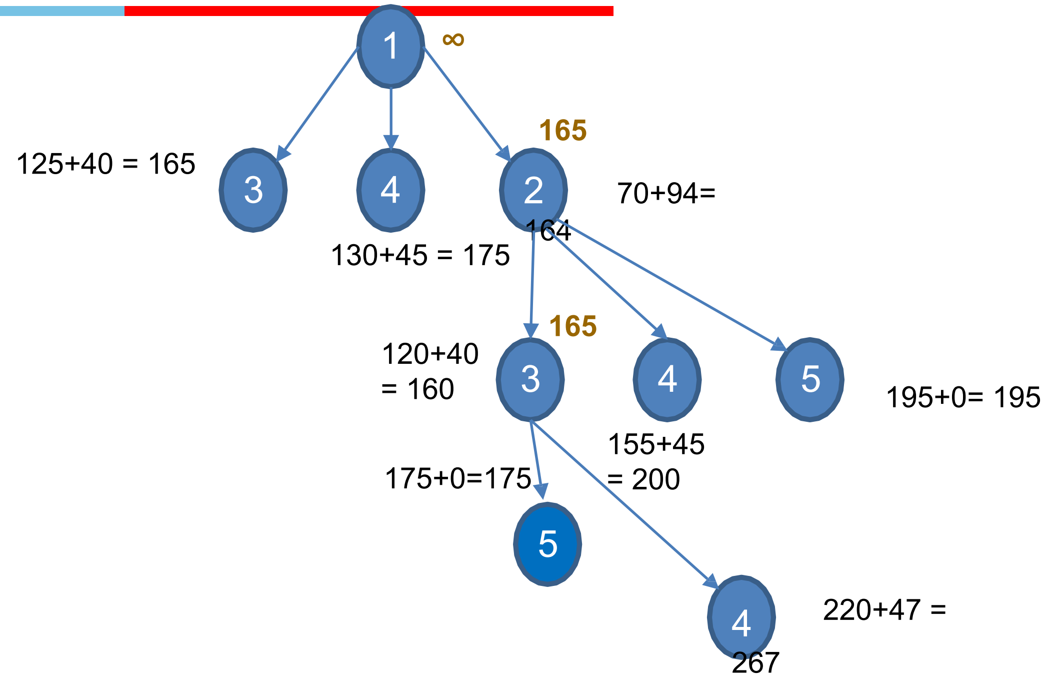
(1: 60)
TEST-F
(1 2: 162) (1 3: 168) (1 4: 175)
TEST-F
(1 2 3: 163) (1 2 4: 200) (1 2 5: 195) (1 3: 168) (1 4: 175)
TEST-F
(1 2 3 5: 165) (1 2 3 4: 265) (1 2 4: 200) (1 2 5: 195) (1 3: 168) (1 4: 175)

Recursive Best First Search A*

Remember the next best alternative f-Cost to regenerate

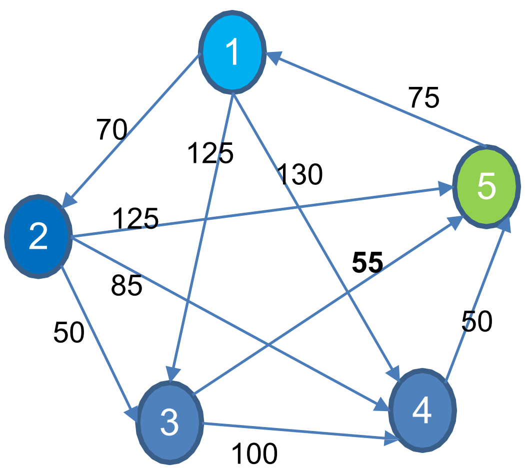


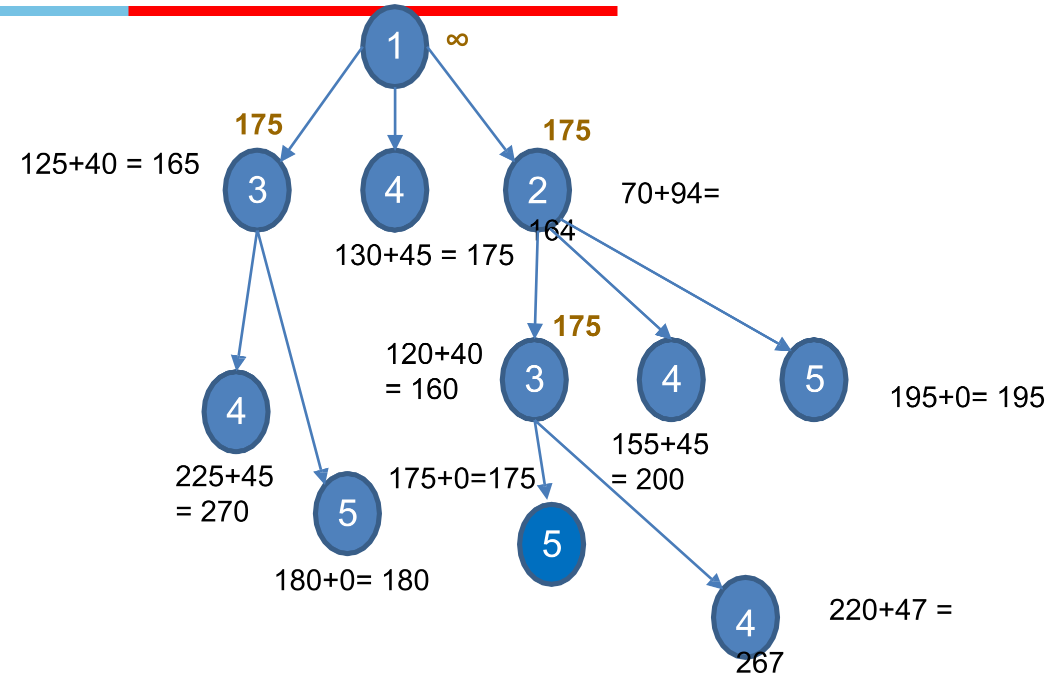
n	h(n)
1	60
2	94
3	40
4	45
5	0



Recursive Best First Search A*

Remember the next best alternative f-Cost to regenerate

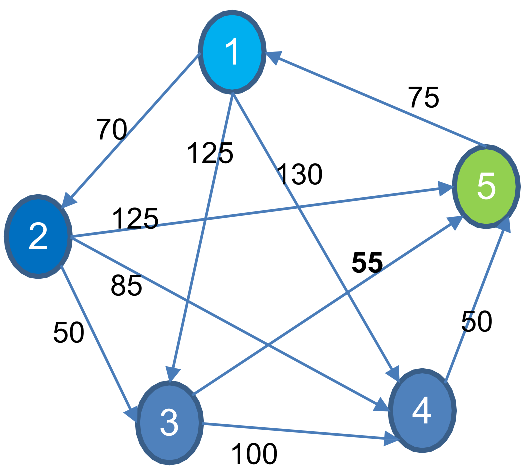


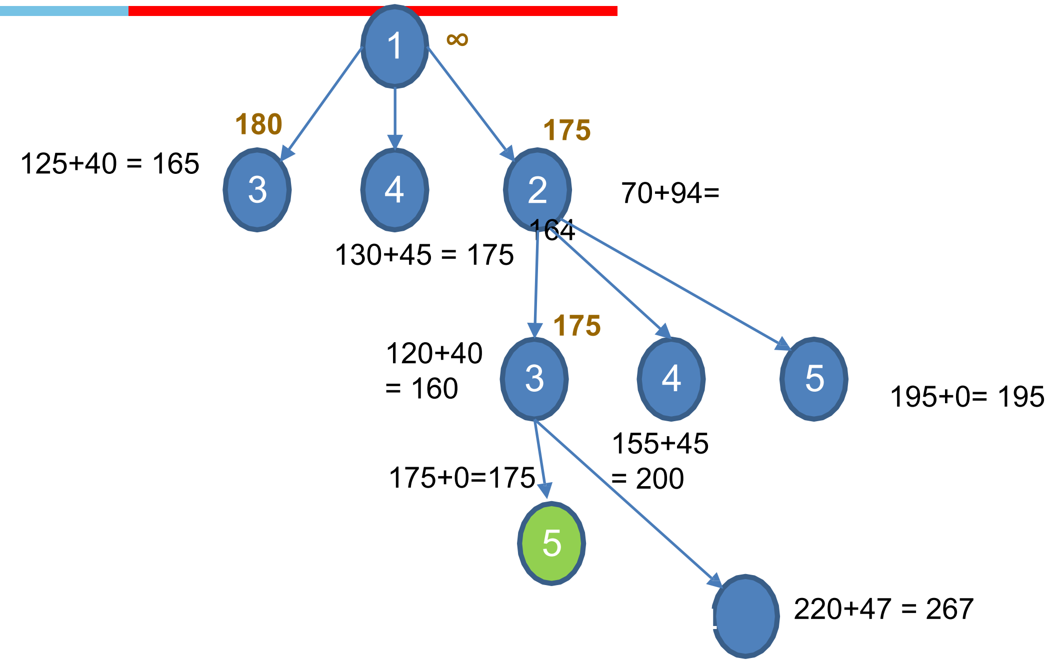


n	h(n)
1	60
2	94
3	40
4	45
5	0

Recursive Best First Search A*

Remember the next best alternative f-Cost to regenerate





n	h(n)
1	60
2	94
3	40
4	45
5	0

If the current best leaf value > best alternative path

Best leaf value of the forgotten subtree is backed up to the ancestors

Recursion unwinds

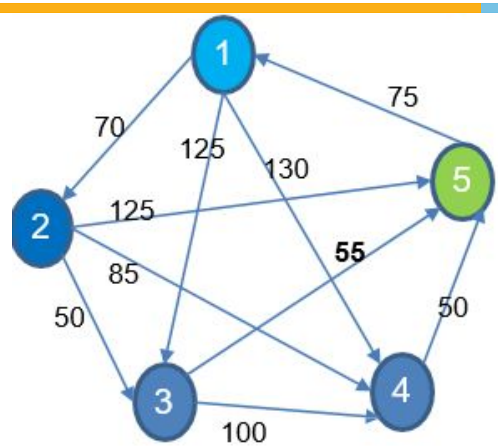
Else

Continue expansion

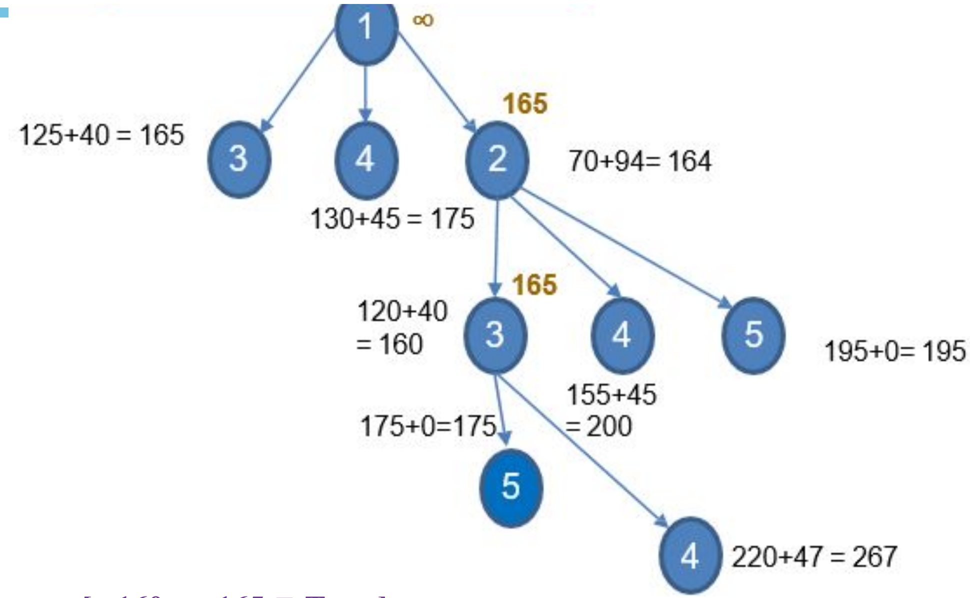
Space Usage = O(bd) very less

Recursive Best First Search A*

Remember the next best alternative f-Cost to regenerate



n	h(n)
1	60
2	94
3	40
4	45
5	0



(1, 60)

(1 2 | 164) (1 3 | 165) (1 4 | 175)

(1 2 3 | 160) (1 3 | 165) (1 4 | 175) (1 2 5 | 195) (1 2 4 | 200) [160 <= 165 □ True]

(1 2 3 5 | 175) (1 3 | 165) (1 4 | 175) (1 2 5 | 195) (1 2 4 | 200) (1 2 3 4 | 265) [175 <= 165 □ False]

(1 3 | 165) (1 2 | 175) (1 4 | 175) [165 <= 175 □ True]

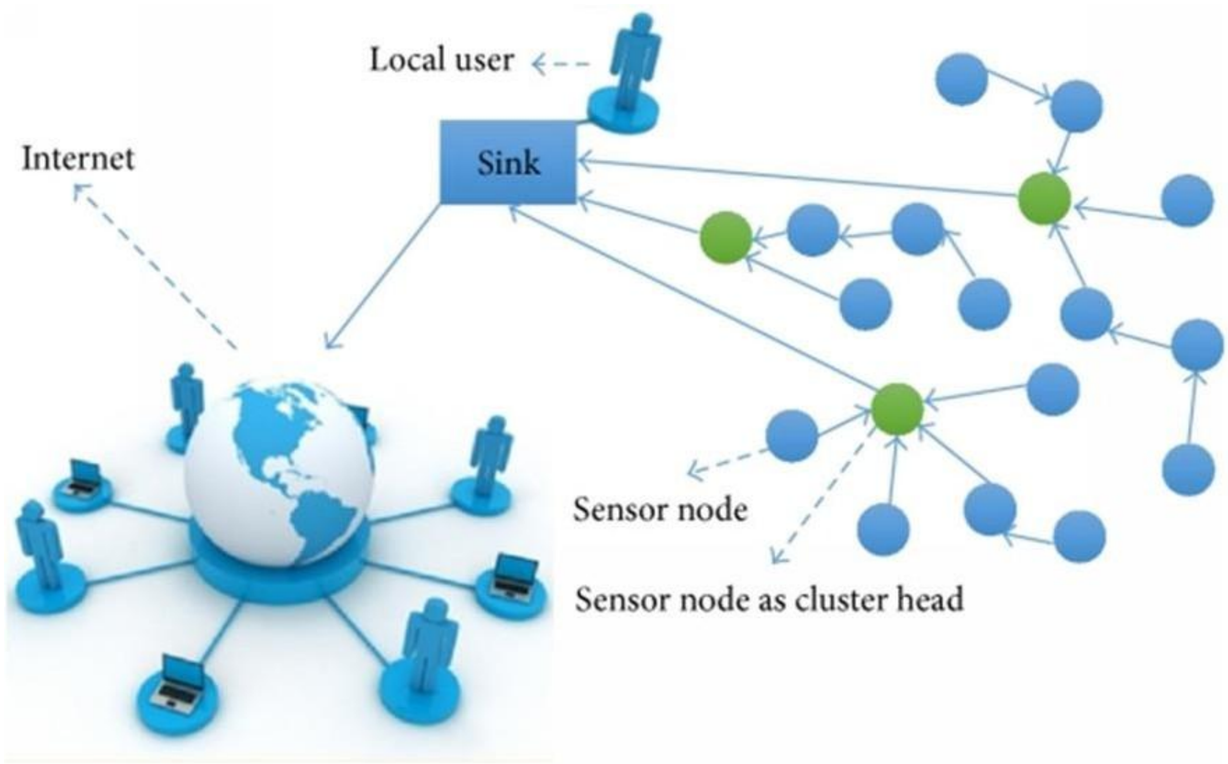
(1 3 5 | 180) (1 2 | 175) (1 4 | 175) (1 3 4 | 270) [185 <= 175 □ False]

(1 2 | 175) (1 4 | 175) (1 3 | 180)

(1 2 3 | 160) (1 4 | 175) (1 3 | 180) (1 2 5 | 195) (1 2 4 | 200)

(1 2 3 5 | 175) (1 4 | 175) (1 3 | 180) (1 2 5 | 195) (1 2 4 | 200) (1 2 3 4 | 267)

Case Study – Search in Network Routing



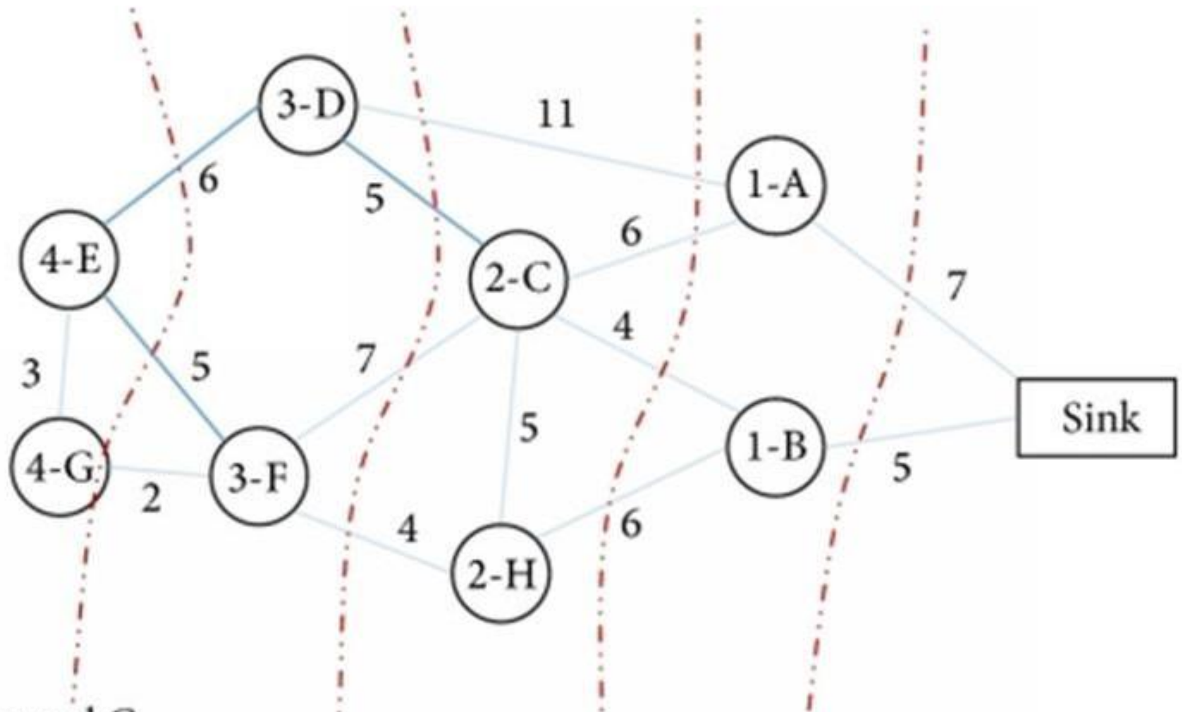
Source Credit :

AR-RBFS: Aware-Routing Protocol Based on Recursive Best-First Search Algorithm for Wireless Sensor Networks
https://doi.org/10.1155/2016/8743927

Case Study – Search in Network Routing

A	14.1
B	11.3
C	8.2
H	6.6
F	2
E	3
D	4.8

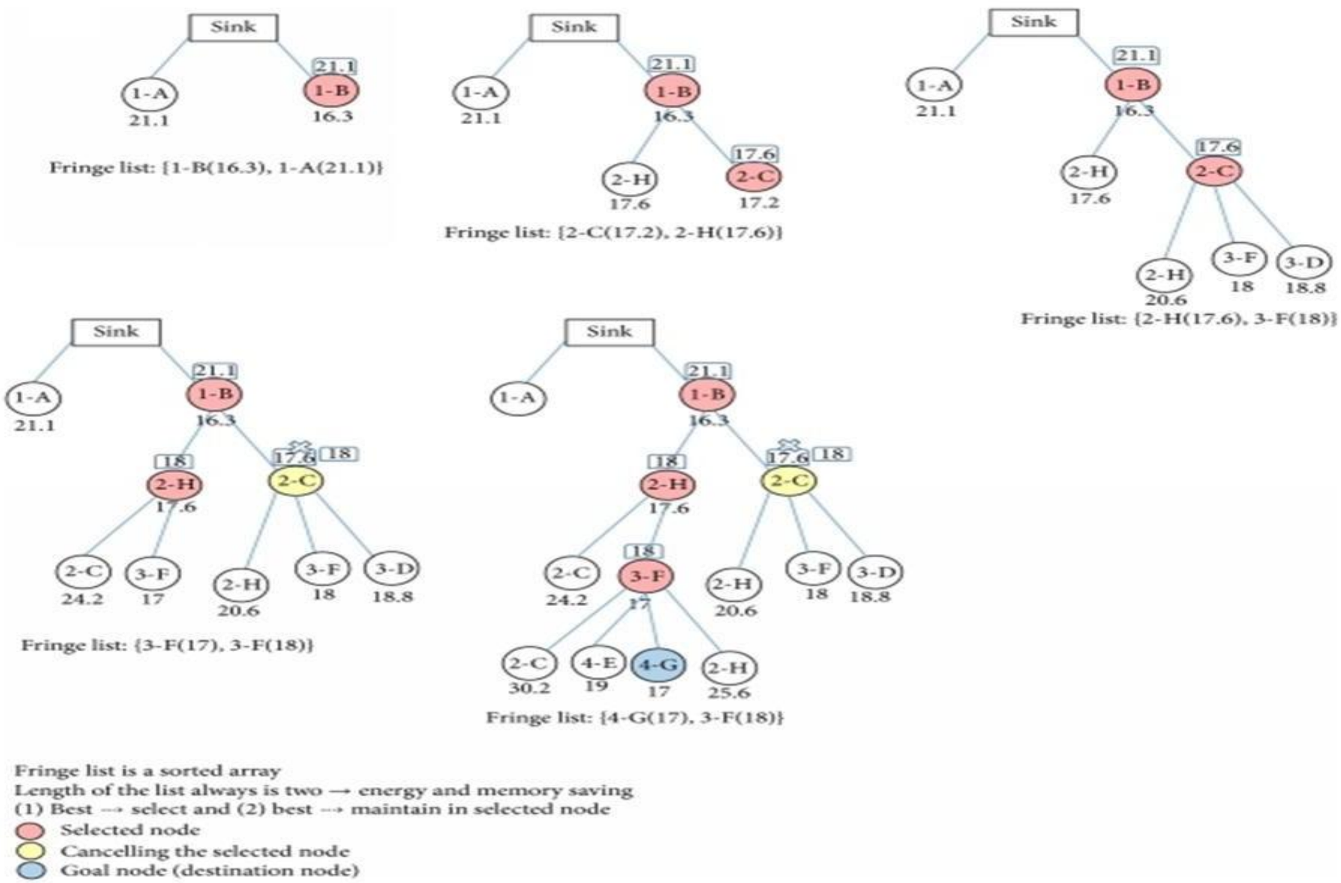
Heuristic values toward G



Source Credit :

AR-RBFS: Aware-Routing Protocol Based on Recursive Best-First Search Algorithm for Wireless Sensor Networks
https://doi.org/10.1155/2016/8743927

Case Study – Search in Network Routing



Source Credit :

AR-RBFS: Aware-Routing Protocol Based on Recursive Best-First Search Algorithm for Wireless Sensor Networks
https://doi.org/10.1155/2016/8743927

Required Reading: AIMA - Chapter #3: 3.1, 3.2, 3.3, 3.4,3.5

Next Class Plan :
Heuristic Design
Local Search Algorithm

Thank You for all your Attention

Note : Some of the slides are adopted from AIMA TB materials