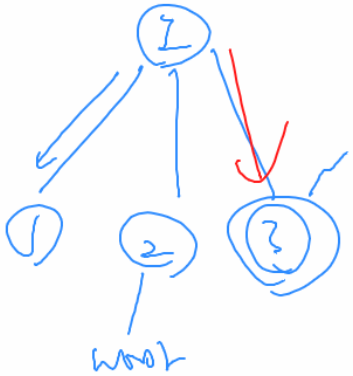
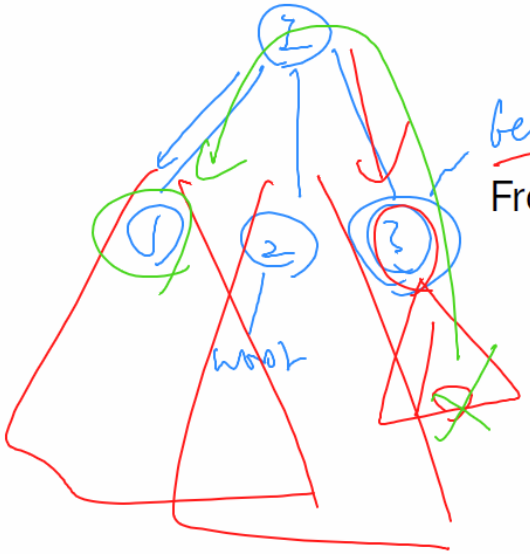


Informed Search

From domain knowledge, obtain an **evaluation function**.

- best-first search: order nodes according to the evaluation function value
- greedy search: minimize estimated cost $h(n)$ for reaching the goal: this is fast, but incomplete and non-optimal.
- A^*: minimize $f(n) = g(n) + h(n)$, where $g(n)$ is the current path cost from start to n, and $h(n)$ is the estimated cost from n to goal.

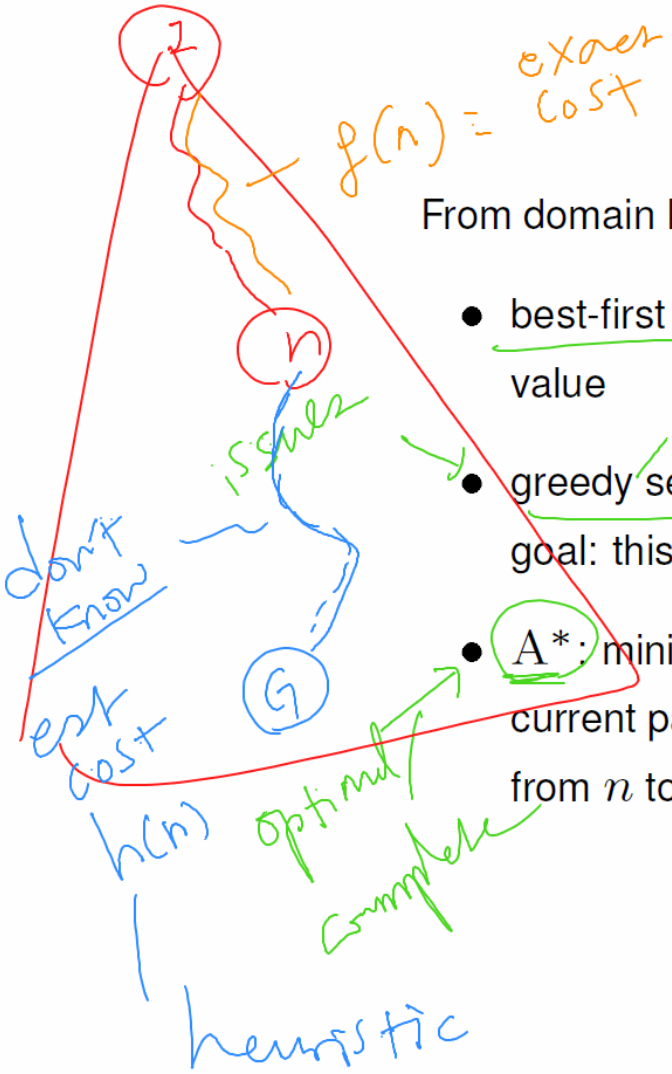


Informed Search

From domain knowledge, obtain an **evaluation function**.

- best-first search: order nodes according to the evaluation function value
- greedy search: minimize estimated cost $h(n)$ for reaching the goal: this is fast, but incomplete and non-optimal.
- A^*: minimize $f(n) = g(n) + h(n)$, where $g(n)$ is the current path cost from start to n, and $h(n)$ is the estimated cost from n to goal.

Informed Search



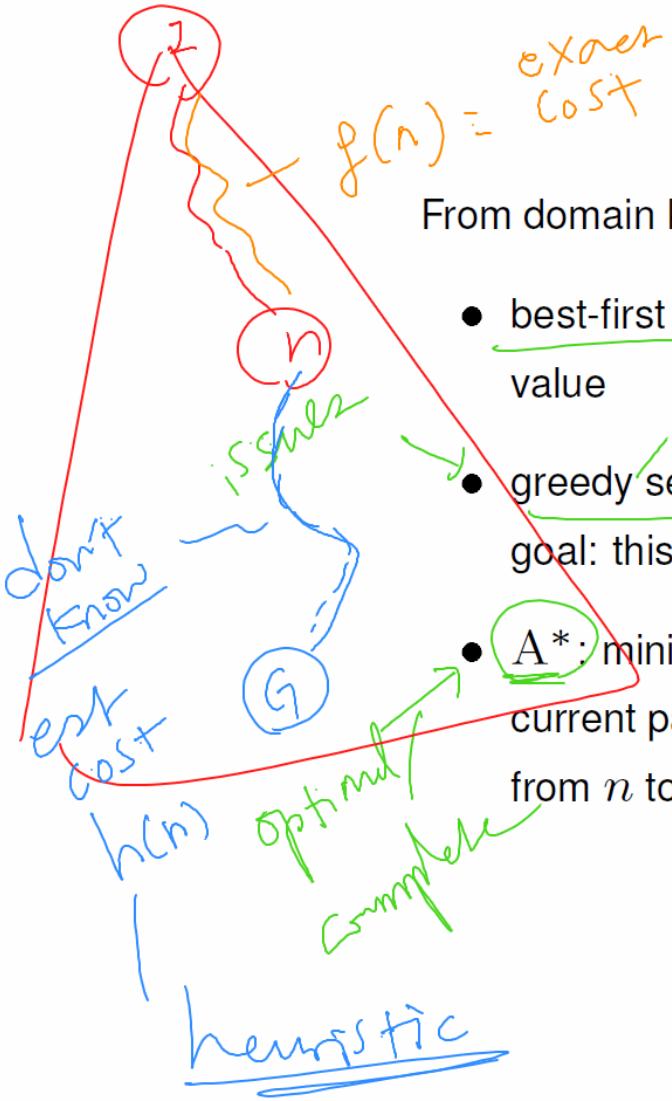
From domain knowledge, obtain an **evaluation function**.

- best-first search: order nodes according to the evaluation function value
- greedy search: minimize estimated cost $h(n)$ for reaching the goal: this is fast, but incomplete and non-optimal.
- A*: minimize $f(n) = g(n) + h(n)$, where $g(n)$ is the current path cost from start to n, and $h(n)$ is the estimated cost from n to goal.

h(n) - est. dist to goal

cost

Informed Search



From domain knowledge, obtain an **evaluation function**.

- best-first search: order nodes according to the evaluation function value
- greedy search: minimize estimated cost $h(n)$ for reaching the goal: this is fast, but incomplete and non-optimal.
- A*: minimize $f(n) = g(n) + h(n)$, where $g(n)$ is the current path cost from start to n, and $h(n)$ is the estimated cost from n to goal.

h(n) - est. dist to goal

cost

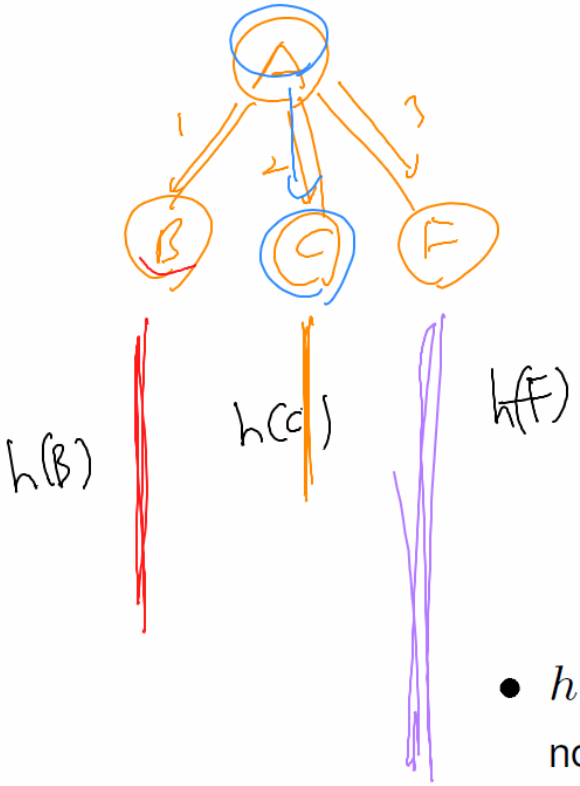
Best First Search

function Best-First-Search (*problem*, *Eval-Fn*)

Queuing-Fn ← sorted list by *Eval-Fn*(node)

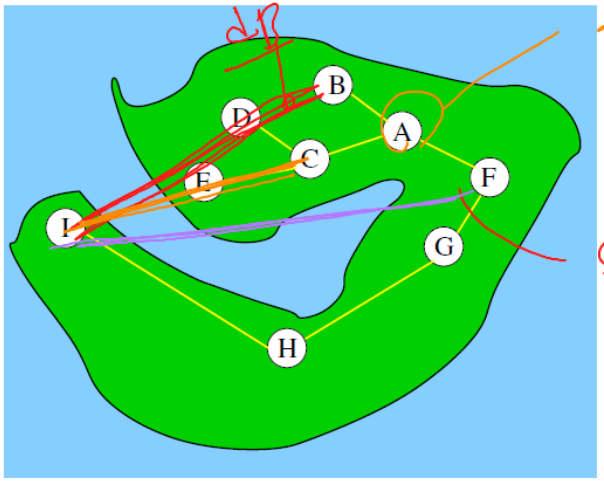
return General-Search(*problem*, *Queuing-Fn*)

- The queuing function queues the expanded nodes, and sorts it every time by the *Eval-Fn* value of each node.
- One of the simplest Eval-Fn: **estimated cost** to reach the goal.



A→C

Heuristic Function

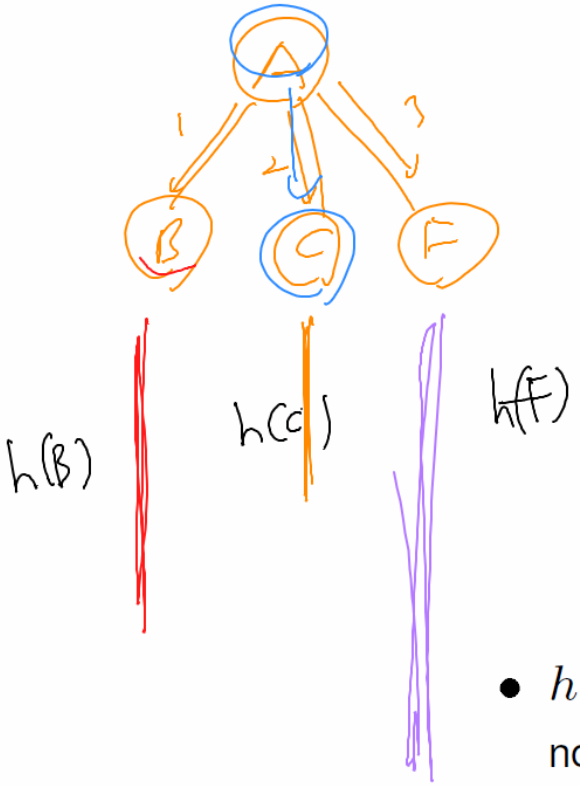


Init

straight line dist

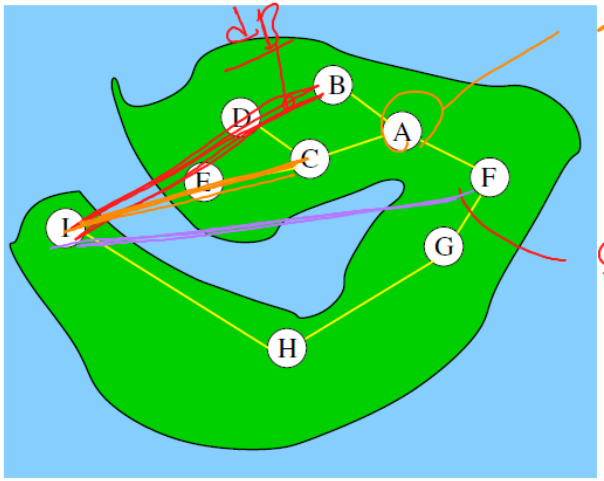
- $h(n)$ = estimated cost of the cheapest path from the state at node n to a goal state.
- The only requirement is the $h(n) = 0$ at the goal.
- **Heuristics** means "to find" or "to discover", or more technically, "how to solve problems" (Polya, 1957).

Right, given just this information, see it looks the most promising. You don't know the exact map.



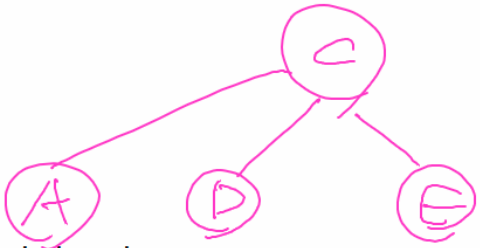
A→C

Heuristic Function



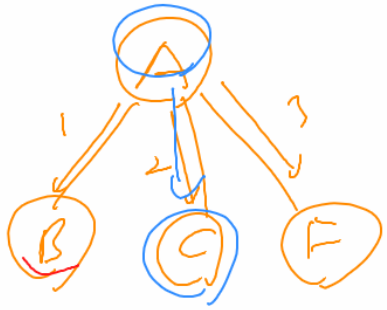
Init

straight line dist



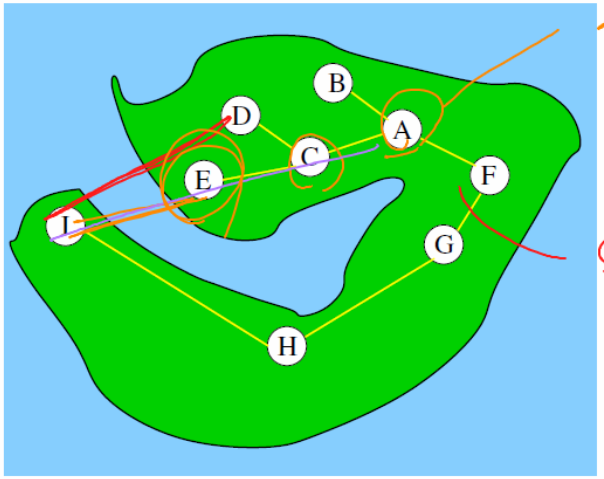
- $h(n)$ = estimated cost of the cheapest path from the state at node n to a goal state.
- The only requirement is the $h(n) = 0$ at the goal.
- **Heuristics** means “to find” or “to discover”, or more technically, “how to solve problems” (Polya, 1957).

So, now.



A→C

Heuristic Function



Init

A→C→E

straight line dist

h(B)

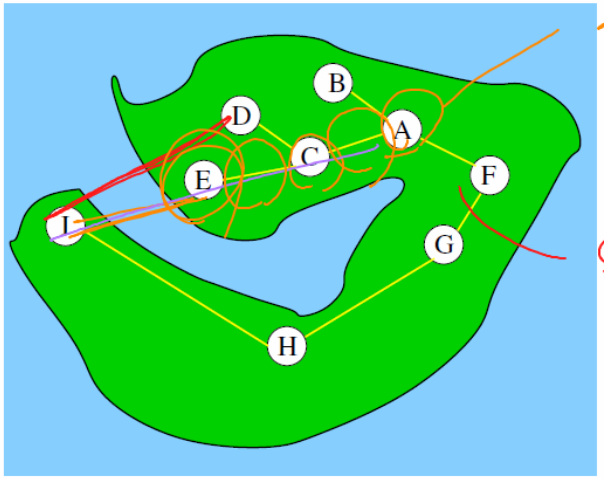
h(C)

h(F)

- $h(n)$ = estimated cost of the cheapest path from the state at node n to a goal state.
- The only requirement is the $h(n) = 0$ at the goal.
- **Heuristics** means "to find" or "to discover", or more technically, "how to solve problems" (Polya, 1957).

Right. And they have to go back. So, this case, there's use the function doesn't help you that much.

Heuristic Function



- $h(n)$ = estimated cost of the cheapest path from the state at node n to a goal state.
- The only requirement is the $h(n) = 0$ at the goal.
- **Heuristics** means “to find” or “to discover”, or more technically, “how to solve problems” (Polya, 1957).

Greedy Best-First Search

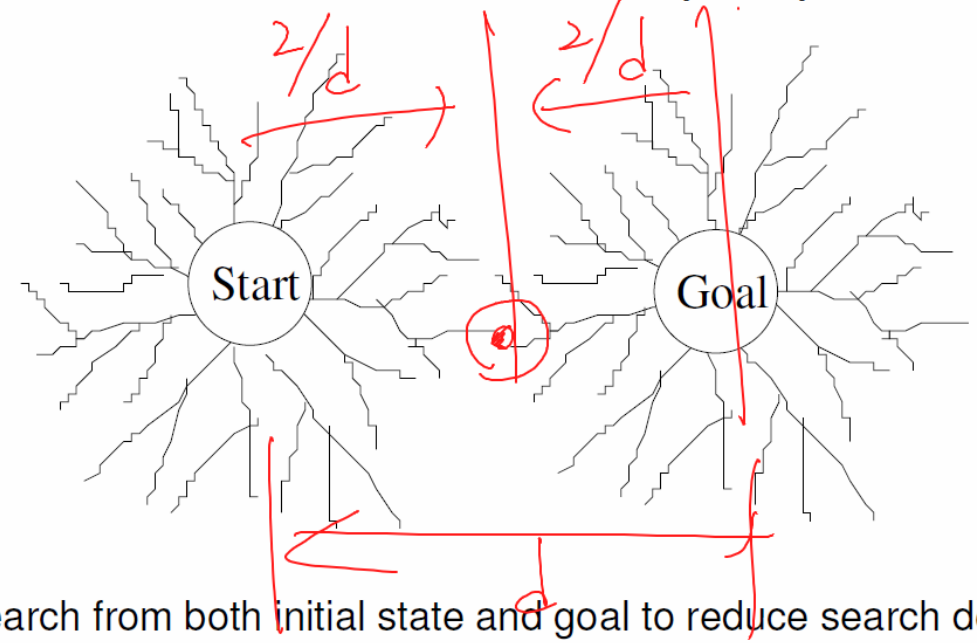
function Greedy-Best-First Search (*problem*)

$h(n)$: estimated cost from n to goal

return Best-First-Search(*problem*, h)

- Best-first with heuristic function $h(n)$ as evaluation function.

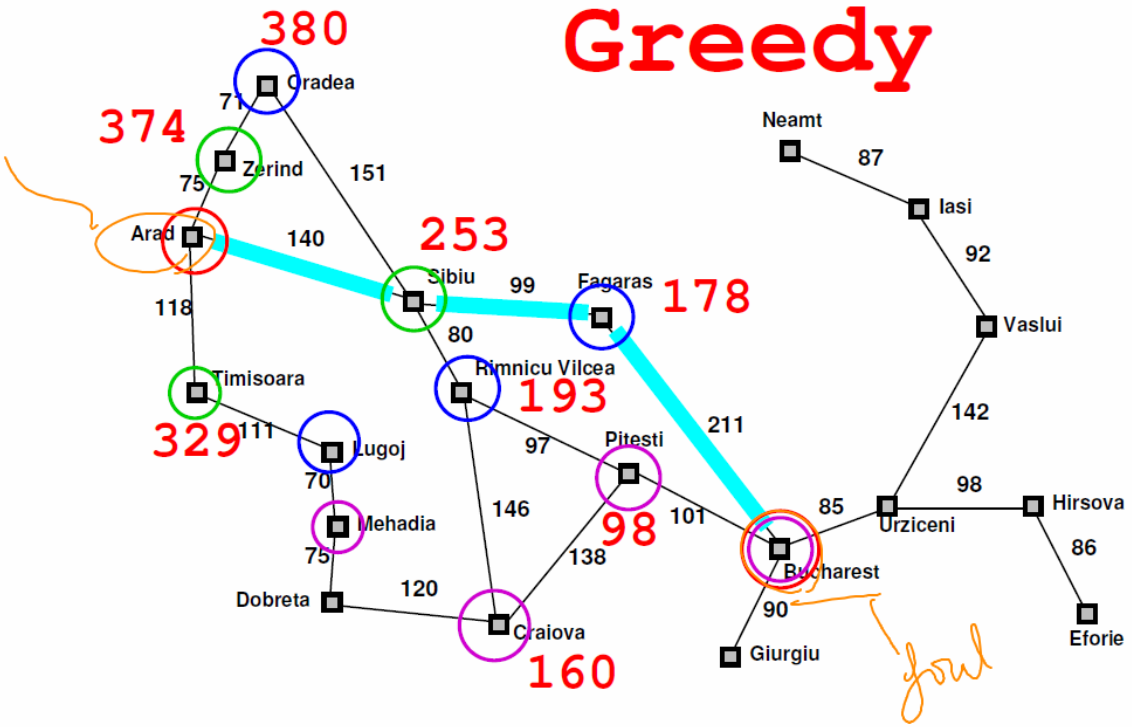
Bidirectional Search (BDS)



- Search from both initial state and goal to reduce search depth.
- $O(b^{d/2})$ of BDS vs. $O(b^{d+1})$ of BFS.

Romania with step costs in km

Greedy

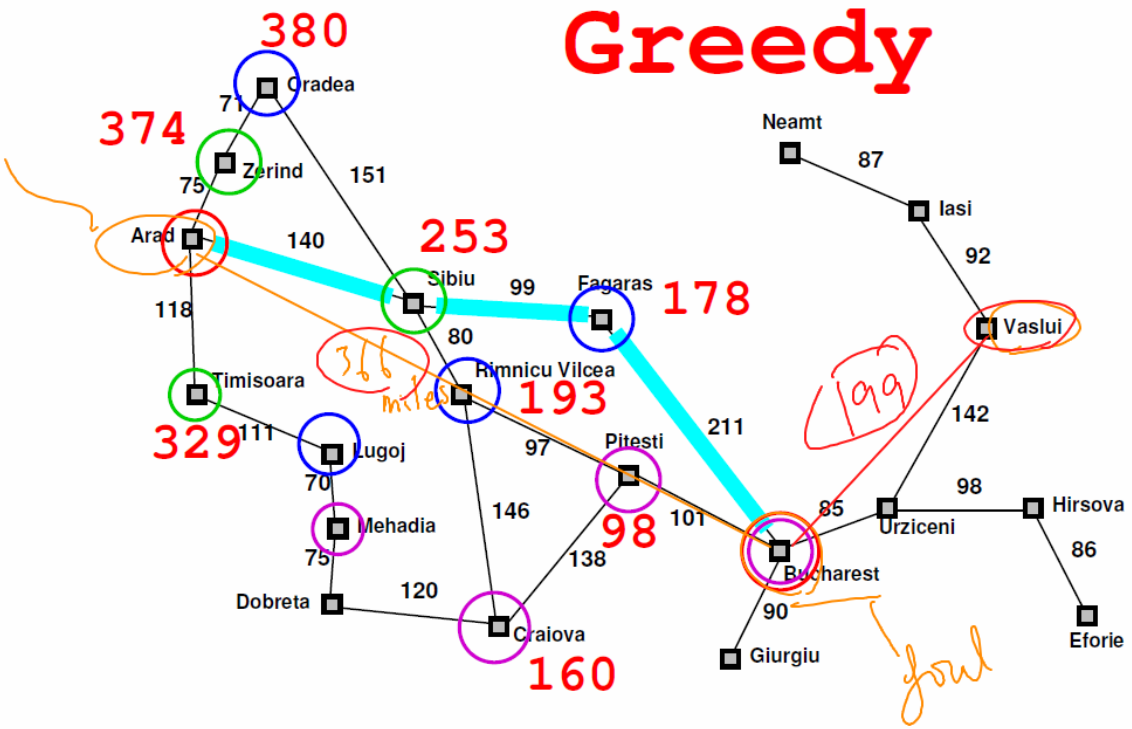


Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Total Path Cost = 450

This is the goal to all of these cities, so era, is the first city, that we can see this this is supposed to be the initial state.

Romania with step costs in km

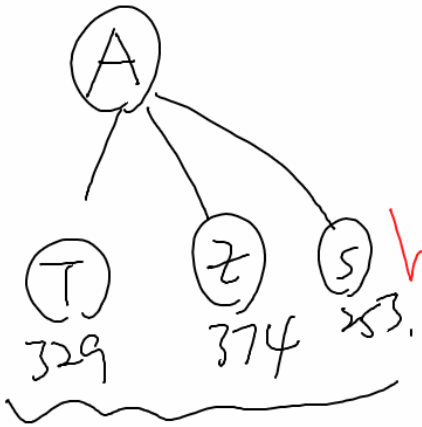


Straight–line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

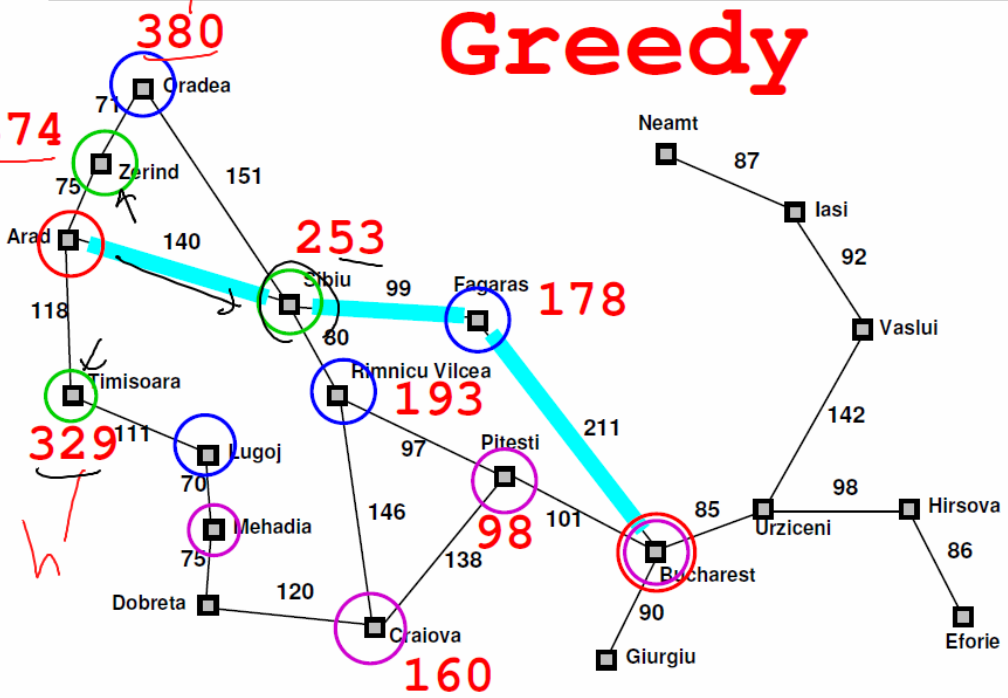
Total Path Cost = 450

So this is the HSLD of a particular.



Romania with step costs in km

Greedy



Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Total Path Cost = 450

So this. So basically, these nodes, those would be separated so if you look at the.

Mouse Select Text Draw Stamp Spotlight Eraser Format Undo Redo Clear Save

Who can see what you share here? Recording On

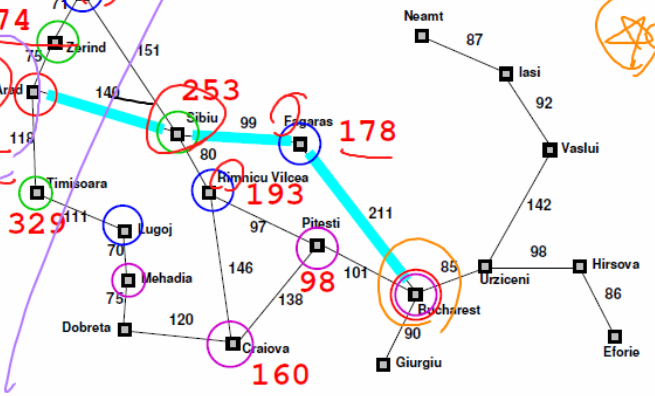
node list

expand

visit

Romania with step costs in km

Greedy



Straight-line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Total Path Cost = 450

So you're, you're done.

You are screen sharing Stop Share

Mouse Select Text Draw Stamp Spotlight Eraser Format Undo Redo Clear Save

Who can see what you share here? Recording On

node list

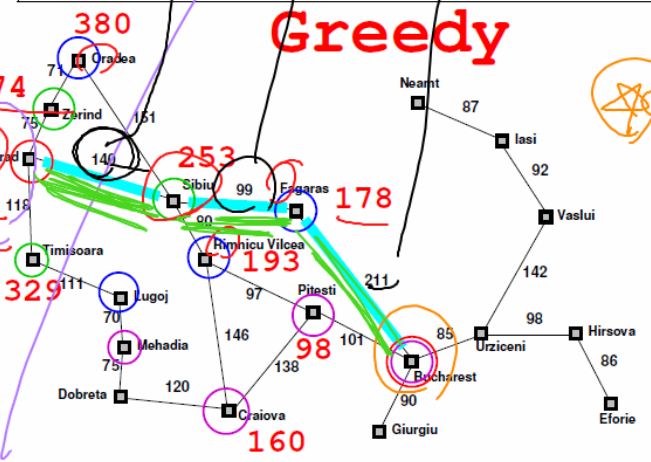
expand

A → S → F → B

140 99 211

Romania with step costs in km

Greedy



Straight-line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Total Path Cost = 450

So for a to s to F to be. So this was 140, which is that this is a 99, then this was to 11.

You are screen sharing Stop Share

Mouse Select Text Draw Stamp Spotlight Eraser Format Undo Redo Clear Save

Who can see what you share here? Recording On

node list

expand

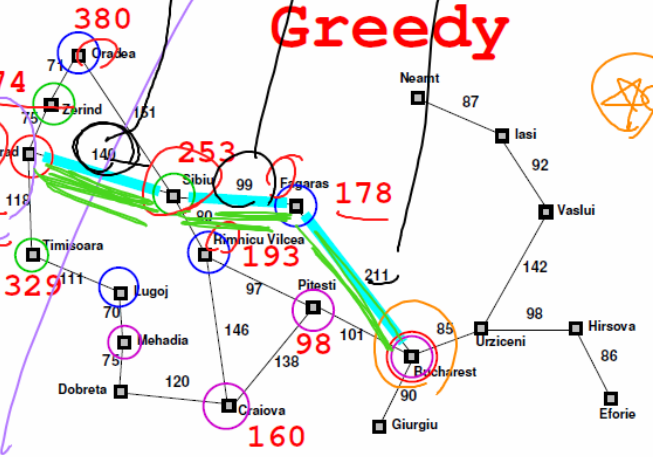
A → S → F → B

140 + 99 + 211

450

Romania with step costs in km

Greedy



Straight-line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Total Path Cost = 450

then you get for 15.

You are screen sharing Stop Share

Mouse Select Text Draw Stamp Spotlight Eraser Format Undo Redo Clear Save

Who can see what you share here? Recording On

Saved as PNG Show in Folder

node list

expand

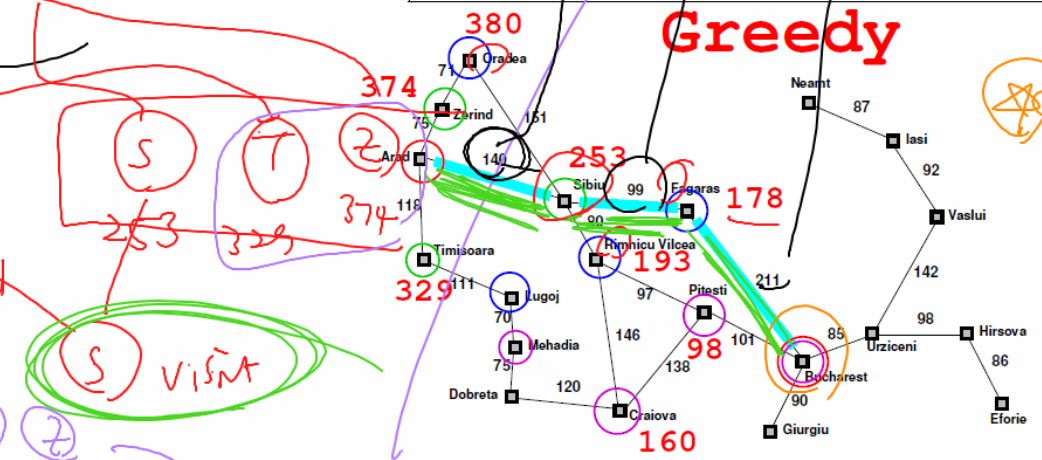
A → S → F → B

140 + 99 + 211

450

Romania with step costs in km

Greedy



Straight-line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Total Path Cost = 450

So that's.

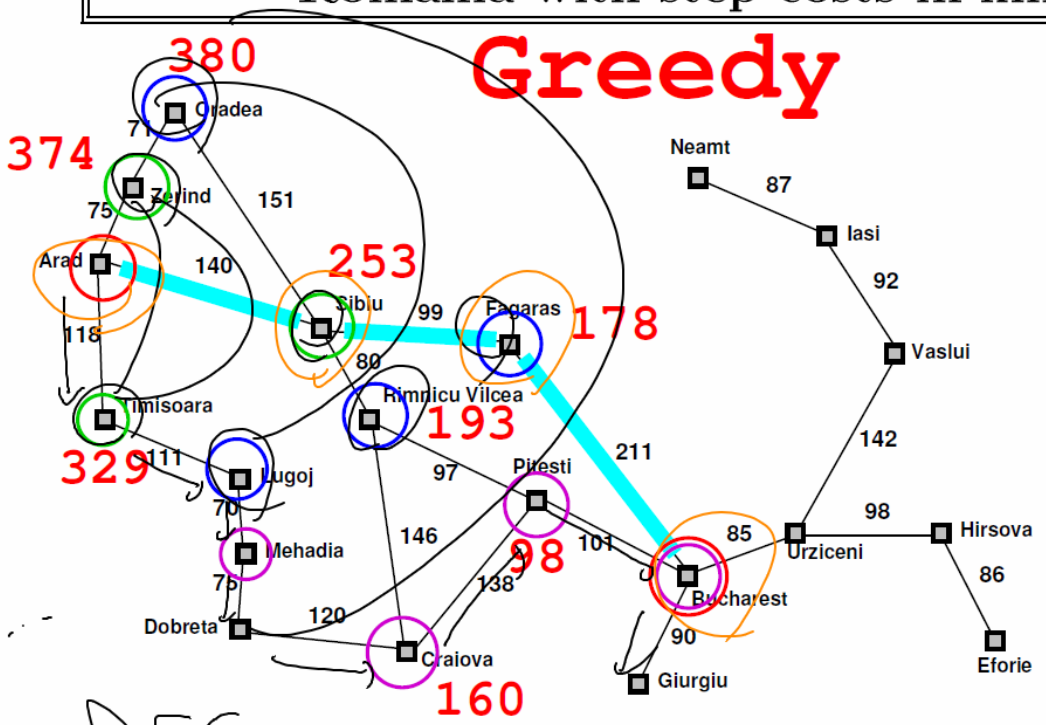
You are screen sharing Stop Share

BFS UCS

Romania with step costs in km

Greedy

380 374 253 178 193 329 160 98



Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

118+111
+70+75
+120
...
450

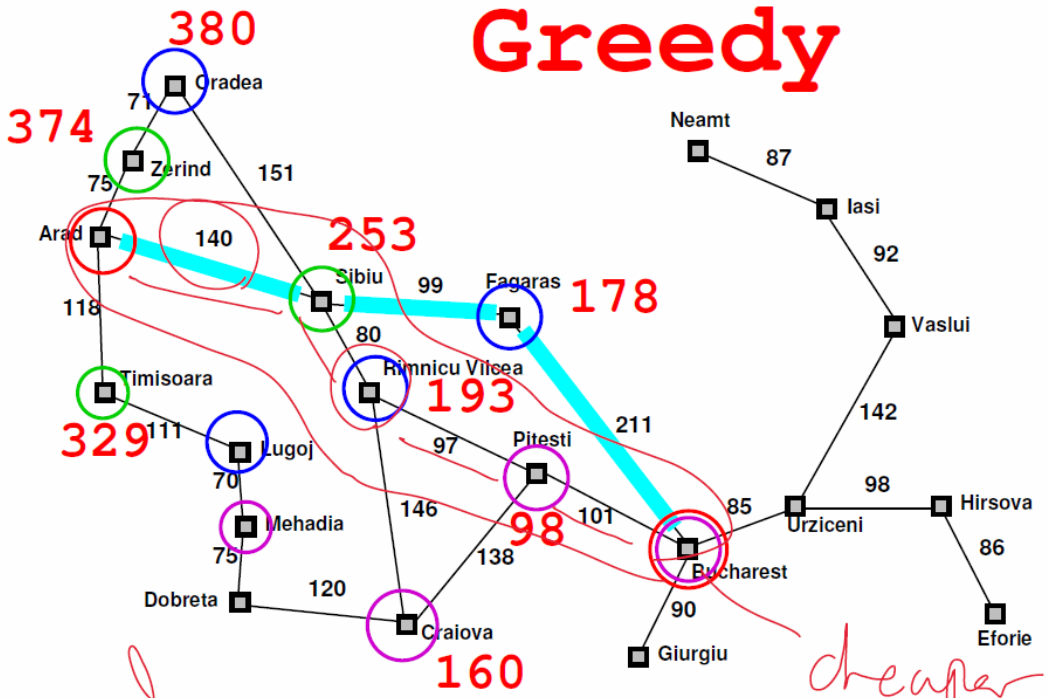
DFS

Total Path Cost = 450

So, in focus such can be.

Romania with step costs in km

Greedy



Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

not optimal.

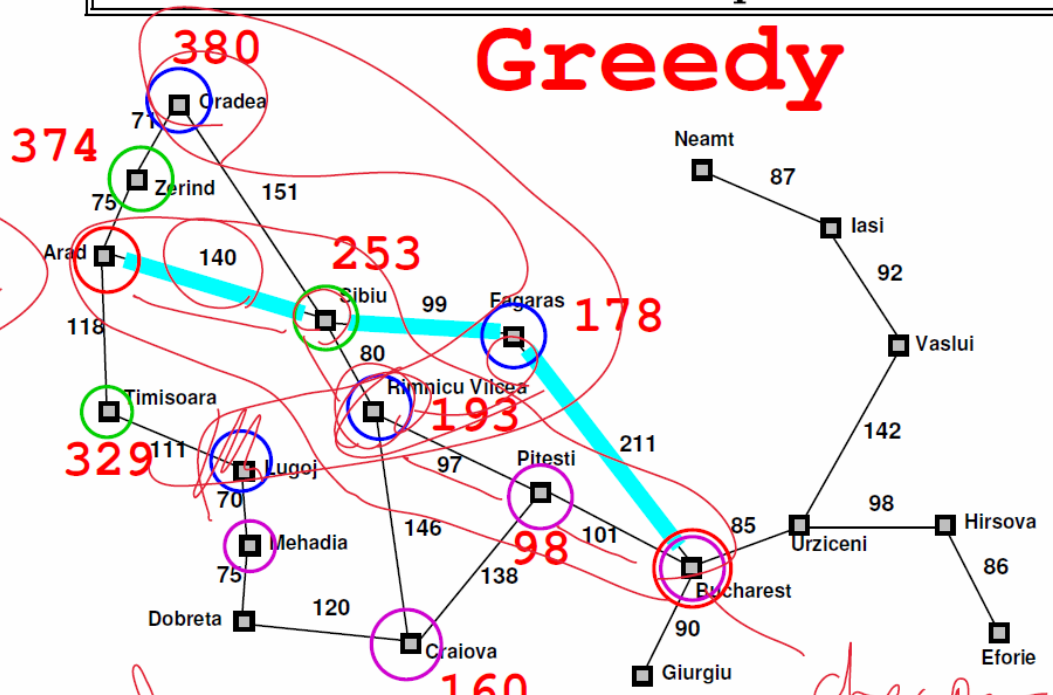
cheaper than 480

Total Path Cost = 450

for 15.

Romania with step costs in km

Greedy

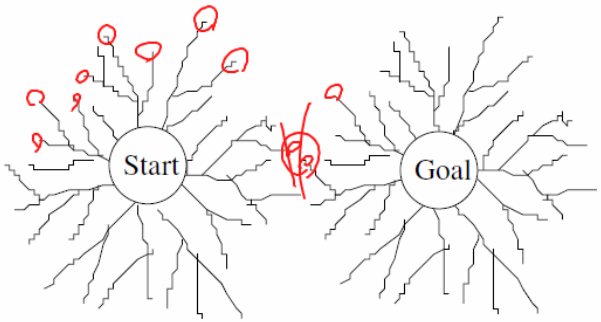


Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Total Path Cost = 450

So, in the worst case this can also grew up explanation.

BDS: Considerations



1. how to back trace from the goal?
2. successors and predecessors: are operations reversible?
3. are goals explicit?: need to know the goal to begin with
4. check overlap in two branches
5. BFS? DFS? which strategy to use? Same or different?

Greedy Best-First Search: Evaluation

Branching factor b and max depth m:

- Fast, just like Depth-First-Search: directed path toward the goal.
- Time: $O(b^m)$
- Space: same as time, in the worst case (bad heuristic function), **unlike DFS**
- Incomplete, just like DFS
- Non-optimal, just like DFS

infinite depth

finite

even when

So if it is a finite then it will be complete, somehow.

A*: Uniform Cost + Heuristic Search

Avoid visiting paths that are already found to be expensive:

- $f(n) = g(n) + h(n)$
- $f(n)$: estimated cost to goal that goes through node n
- **provably complete and optimal!**
- **restrictions**: $h(n)$ should be an **admissible heuristic**
- admissible heuristic: one that **never overestimate** the actual cost of the best solution through n

A* Search

exact

est.

function A*-Search (*problem*)

$g(n)$: current cost up till n

$h(n)$: estimated cost from n to goal

return Best-First-Search(*problem*, $g + h$)

eval-fn

- Condition: $h(n)$ must be an **admissible heuristic function**!
- A* is **complete and optimal!**

Behavior of A^* Search

- usually, the f value never decreases along a given path: **monotonicity**
- in case it is nonmonotonic, i.e. $f(Child) < f(Parent)$, make this adjustment:
  $f(Child) = max(f(Parent), g(Child) + h(Child))$.
- this is called **pathmax**

And there are some ways of making sure that this properties mean.

f(n)

f(Z) = 75 + 374 = 449

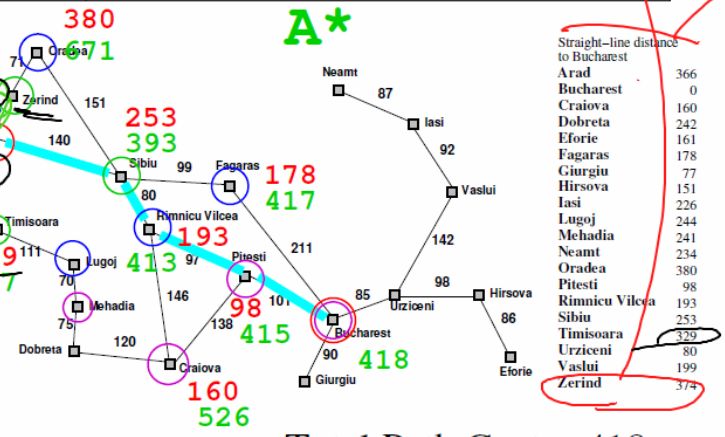
h

h_{SLD}

f(T) = 118 + 329 = 447

Romania with step costs in km

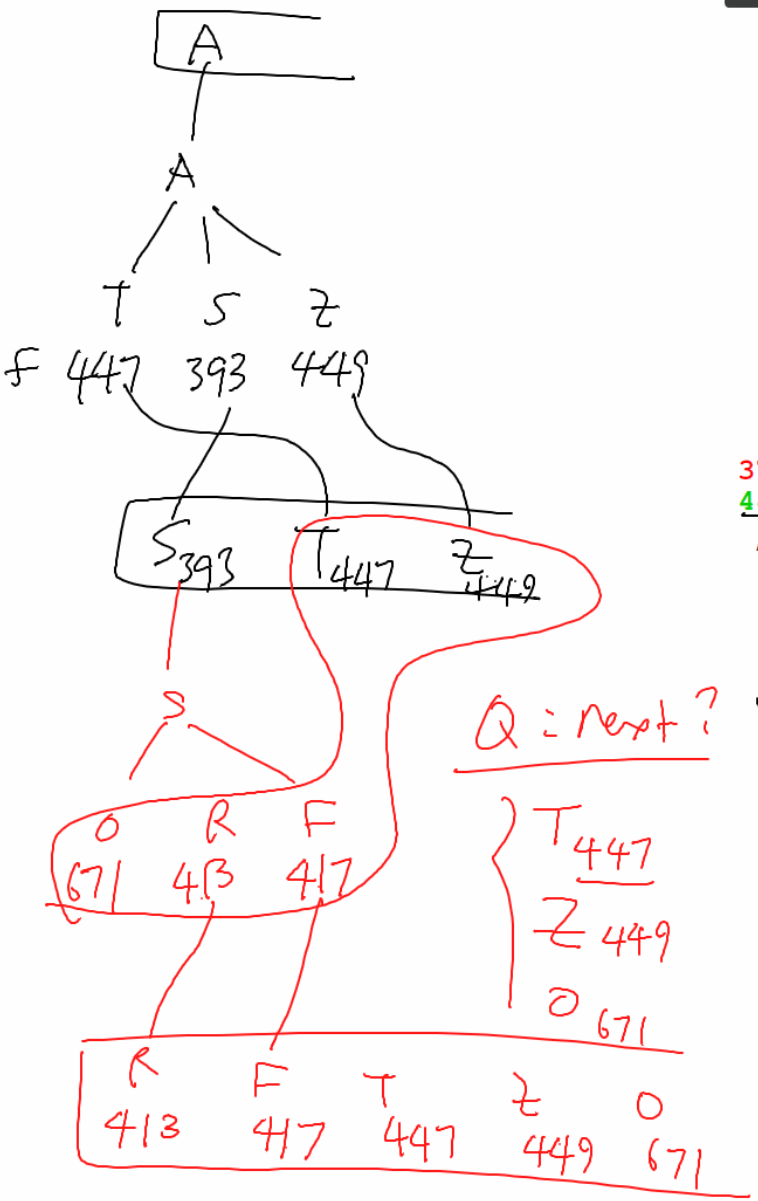
A*



Straight-line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Total Path Cost = 418

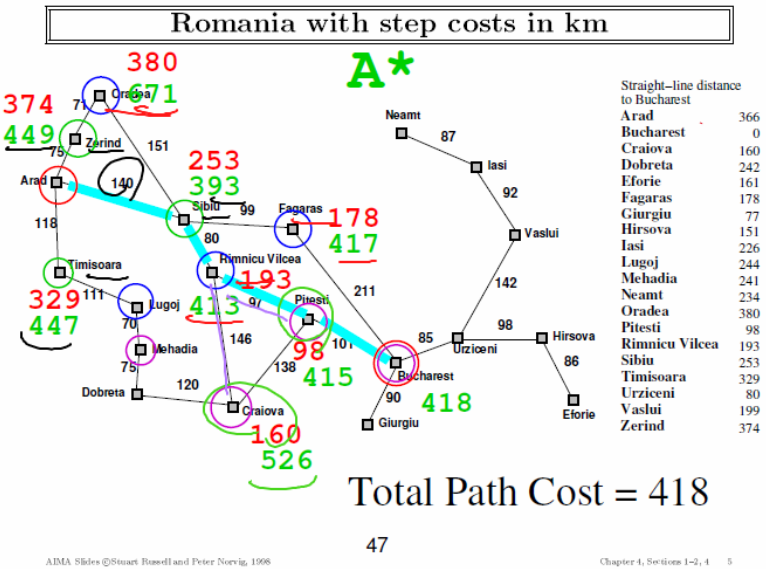
plus the uCLA function value of team, which is 329, which gives you, 447.



A → S
140

R F T Z O
417 447 449 671

R
C P
526 415

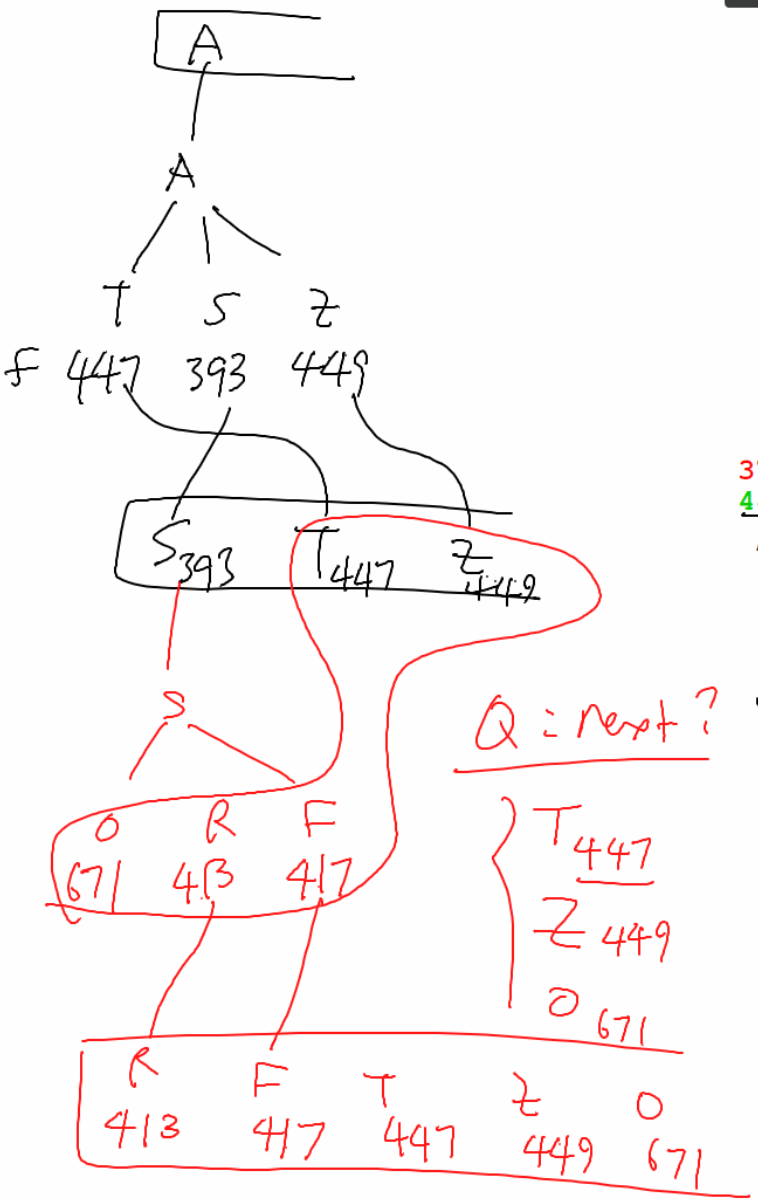


Straight-line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

P F T Z C O
415 417 447 449 526 671

P
C B

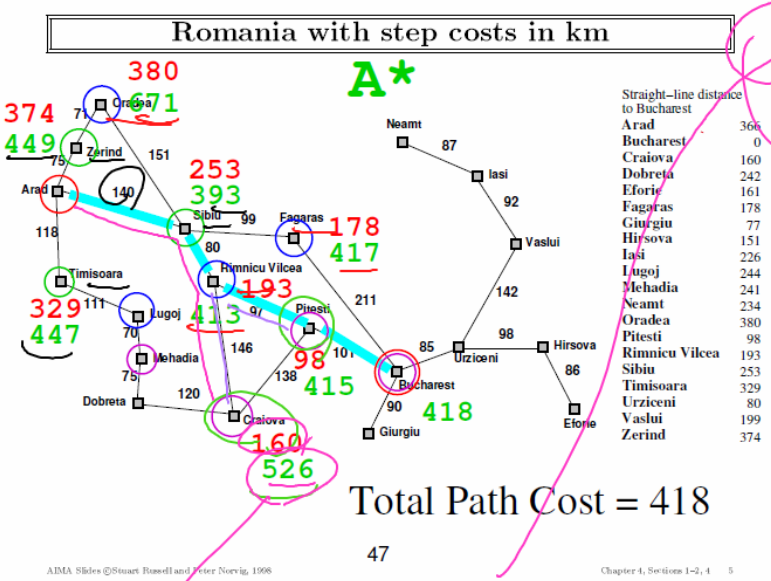
Actually this he has a different value.



A → S
140

R F T Z O
417 447 449 671

R
P
415
526



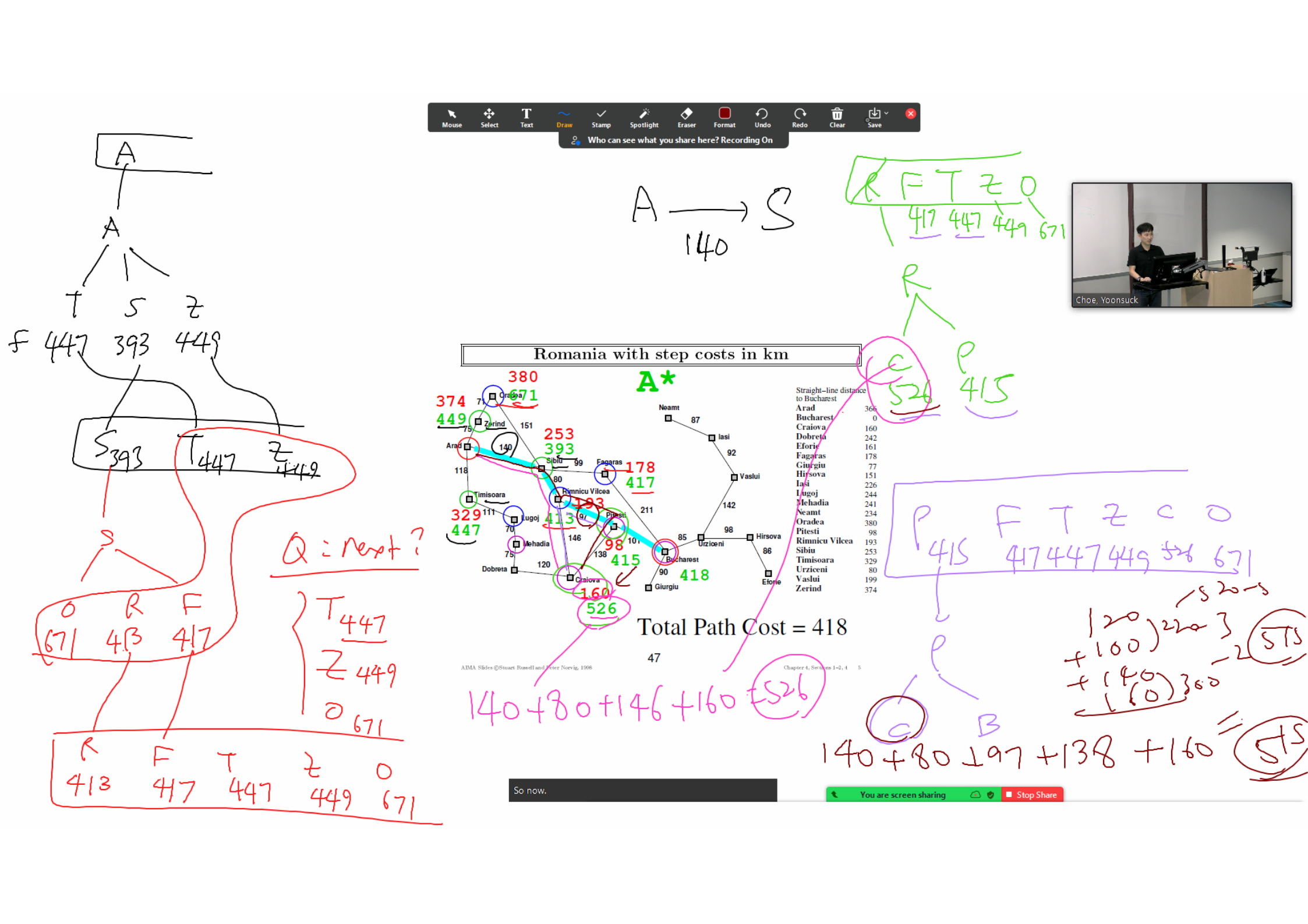
Straight-line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

140+80+146+160

P
415
P F T Z C O
415 417 447 449 526 671

C B

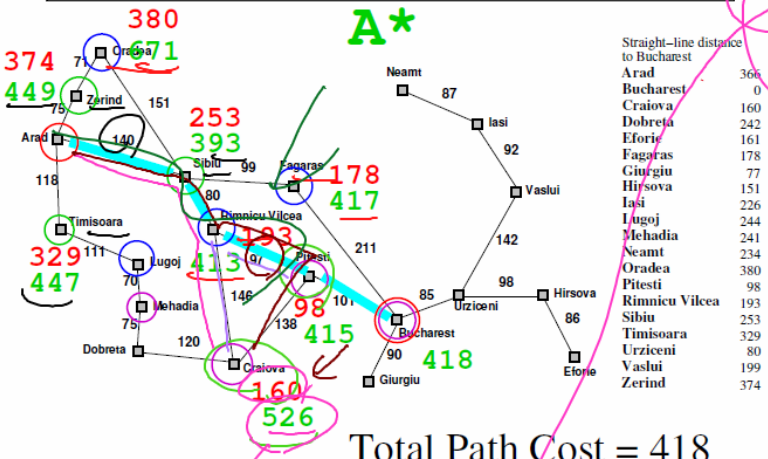
But we encounter see once again here.


Mouse
Select
Text
Draw
Stamp
Spotlight
Eraser
Format
Undo
Redo
Clear
Save
Who can see what you share here? Recording On
A
A
T S Z
F 447 393 449
S393 T447 Z449
S
O R F
671 413 417
R F T Z O
413 417 447 449 671
Q : Next?
T447
Z449
O671
A → S
140
R F T Z O
417 447 449 671
Choe, Yoonsuck
Romania with step costs in km
A*
Straight-line distance to Bucharest
Arad 366
Bucharest 0
Craiova 160
Dobreta 242
Eforie 161
Fagaras 178
Giurgiu 77
Hirsova 151
Iasi 226
Lugoj 244
Mehadia 241
Neamt 234
Oradea 380
Pitesti 98
Rimnicu Vilcea 193
Sibiu 253
Timisoara 329
Urziceni 80
Vaslui 199
Zerind 374
380
671
374
449
253
393
178
417
329
447
413
415
418
160
526
Total Path Cost = 418
47
AIMA Slides ©Stuart Russell and Peter Norvig, 1998
Chapter 4, Sections 1–2, 4 5
140+80+146+160=526
526
415
P F T Z C O
415 417 447 449 526 671
140+80+97+138+160=515
515
So now.
You are screen sharing
Stop Share

Mouse Select Text Draw Stamp Spotlight Eraser Format Undo Redo Clear Save

Who can see what you share here? Recording On

Romania with step costs in km



Straight-line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

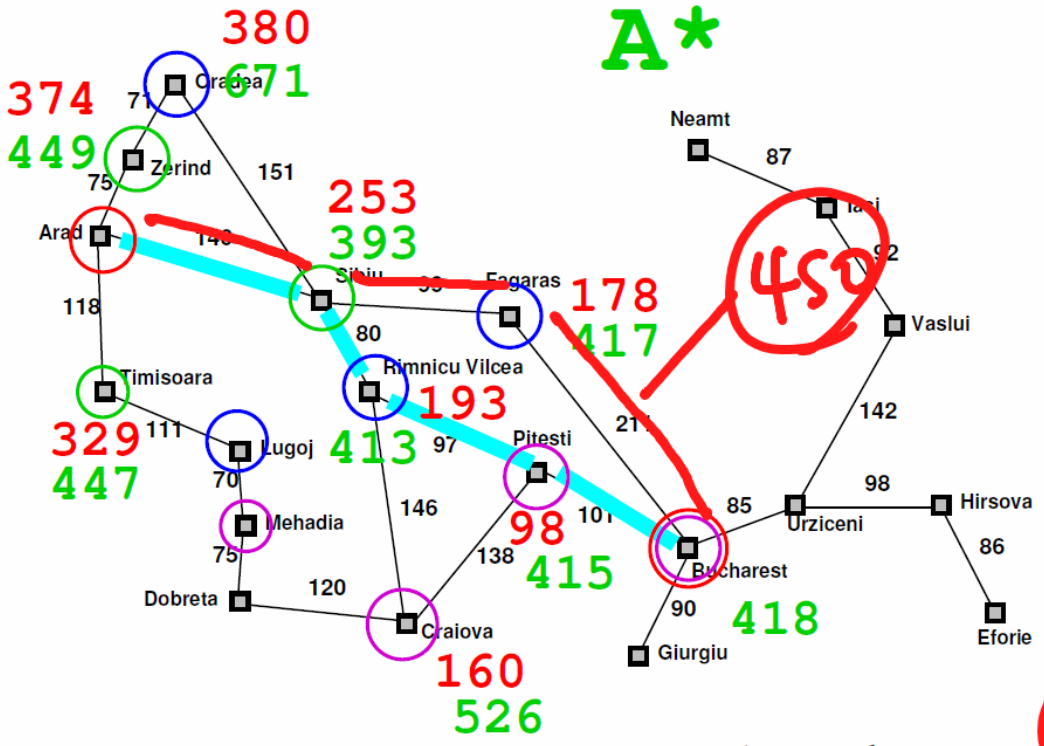
Right. So, the point is, I guess, conveyed.

You are screen sharing Stop Share

A → S → R → P → B

Romania with step costs in km

A*

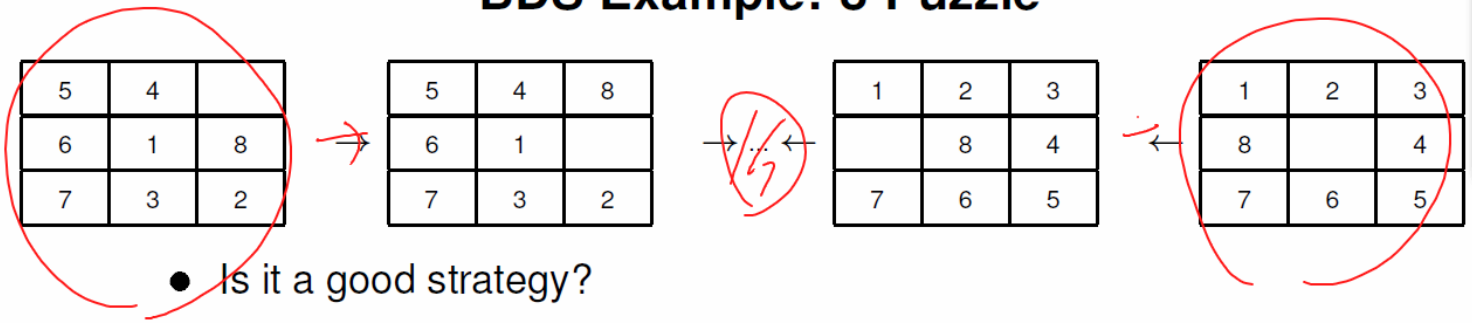


Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Total Path Cost = 418

But it gave you the optimal solution.

BDS Example: 8-Puzzle

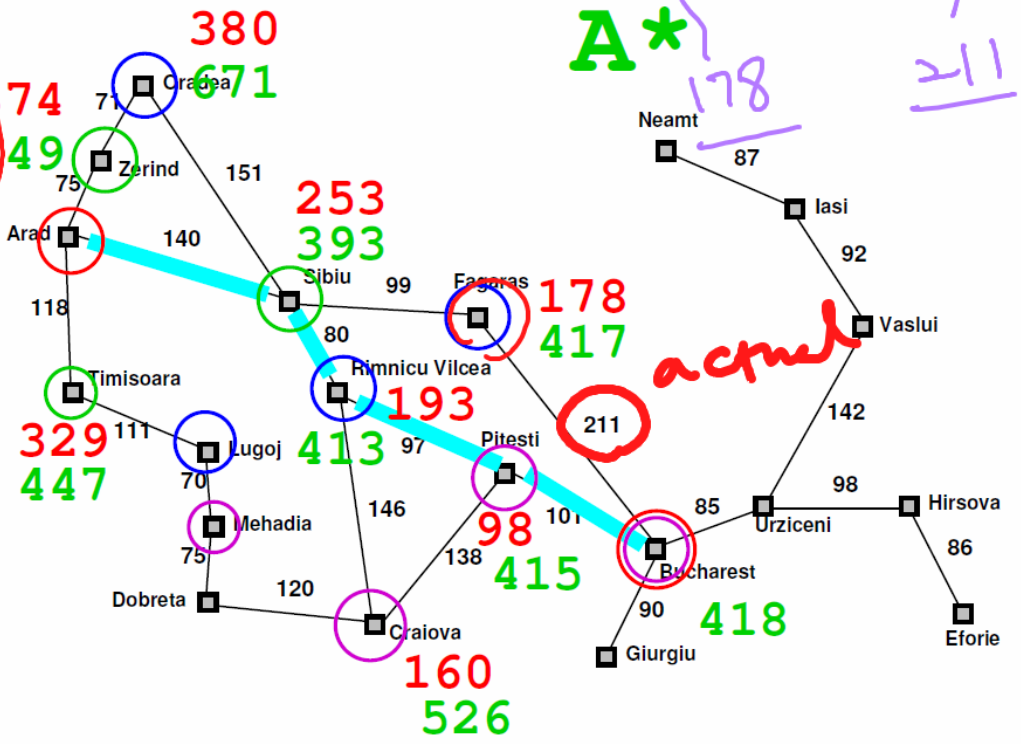


- Is it a good strategy?
- What about Chess? Would it be a good strategy?
- What kind of domains may be suitable for BDS?

And this is the goal state. So you start moving the tiles from the initial state, that way and then you start moving the times from the state backward and hope to meet some.

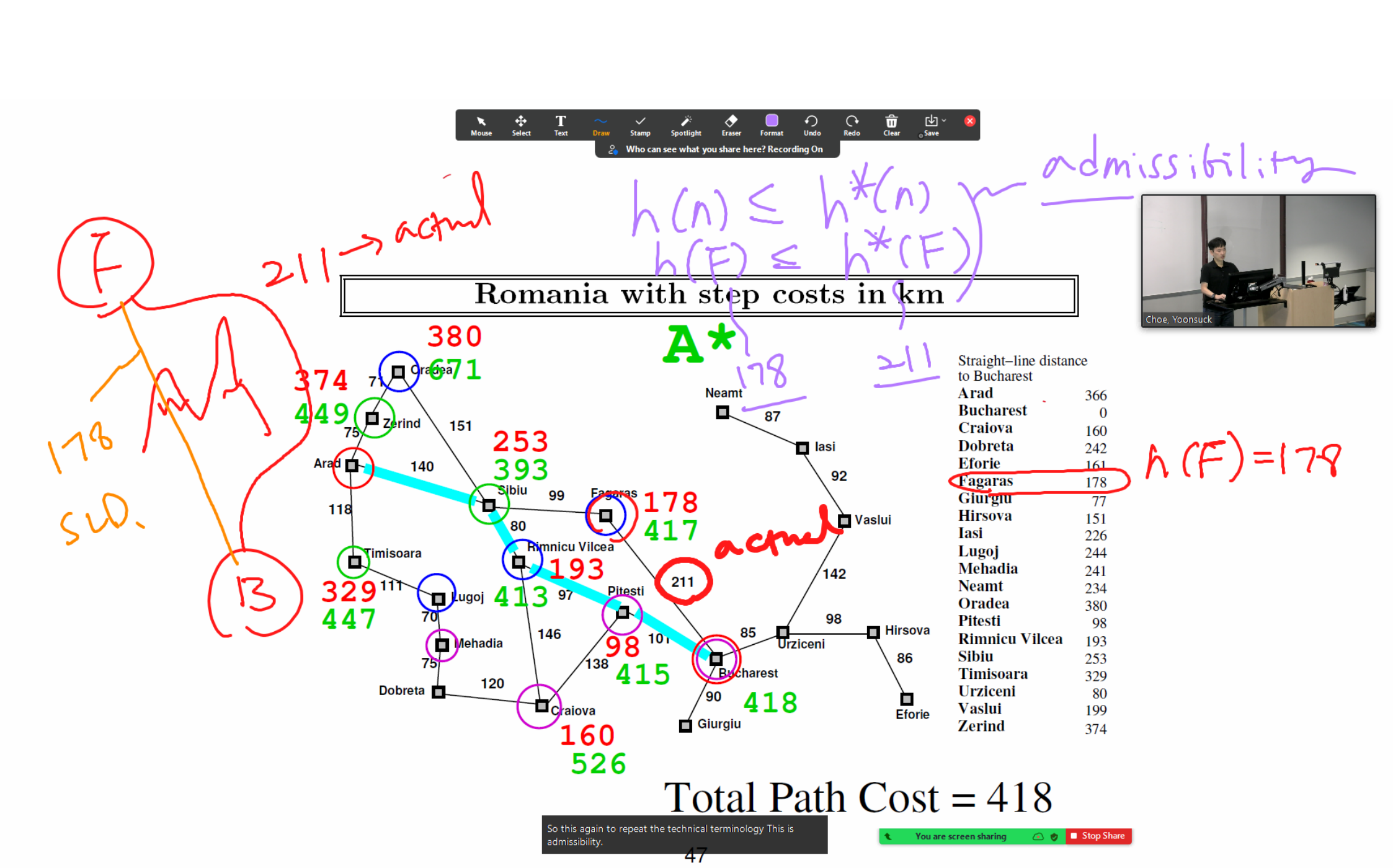
Romania with step costs in km

A*



Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Total Path Cost = 418


Mouse
Select
Text
Draw
Stamp
Spotlight
Eraser
Format
Undo
Redo
Clear
Save
Who can see what you share here? Recording On
admissibility
h(n) ≤ h*(n)
h(F) ≤ h*(F)
211 → actual
F
178
SLO.
B
Choe, Yoonsuck
Romania with step costs in km
A*
380
671
374
449
178
≥11
253
393
178
417
actual
211
193
413
98
415
160
526
329
447
418
Oradea
Neamt
Zerind
Iasi
Arad
Sibiu
Fagaras
Vaslui
Rimnicu Vilcea
Timisoara
Pitesti
Lugoj
Hirsova
Mehadia
Urziceni
Bucharest
Dobreta
Craiova
Giurgiu
Eforie
71
87
75
151
92
140
99
118
80
142
111
97
70
146
85
98
101
86
75
138
120
90
Straight–line distance to Bucharest
Arad 366
Bucharest 0
Craiova 160
Dobreta 242
Eforie 161
Fagaras 178
Giurgiu 77
Hirsova 151
Iasi 226
Lugoj 244
Mehadia 241
Neamt 234
Oradea 380
Pitesti 98
Rimnicu Vilcea 193
Sibiu 253
Timisoara 329
Urziceni 80
Vaslui 199
Zerind 374
h(F) = 178
Total Path Cost = 418
So this again to repeat the technical terminology This is admissibility.
You are screen sharing
Stop Share
47
AIMA Slides ©Stuart Russell and Peter Norvig, 1998
Chapter 4, Sections 1–2, 4 5

$h(p) = 98$

$\leq$

$h^*(p) = 101$

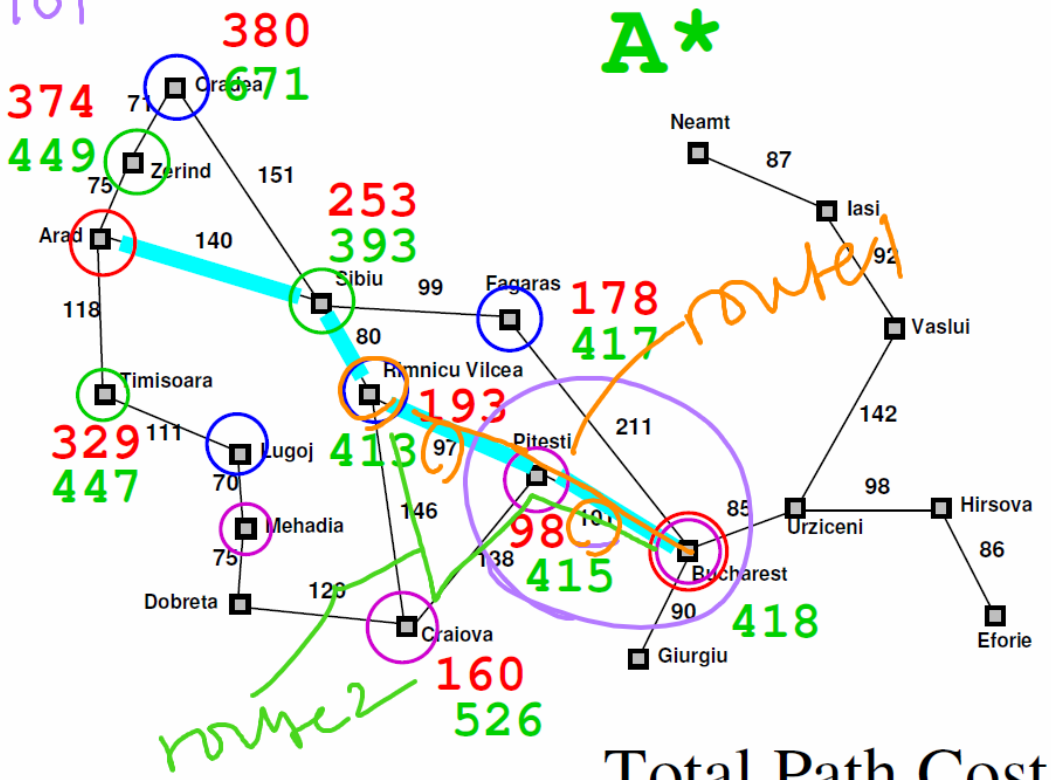
route 1 : 97+101

Romania with step costs in km

A*

$h(R) = 193$

$h^*(R)$



Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Total Path Cost = 418

course this 97 plus 101.

$h(p) = 98$

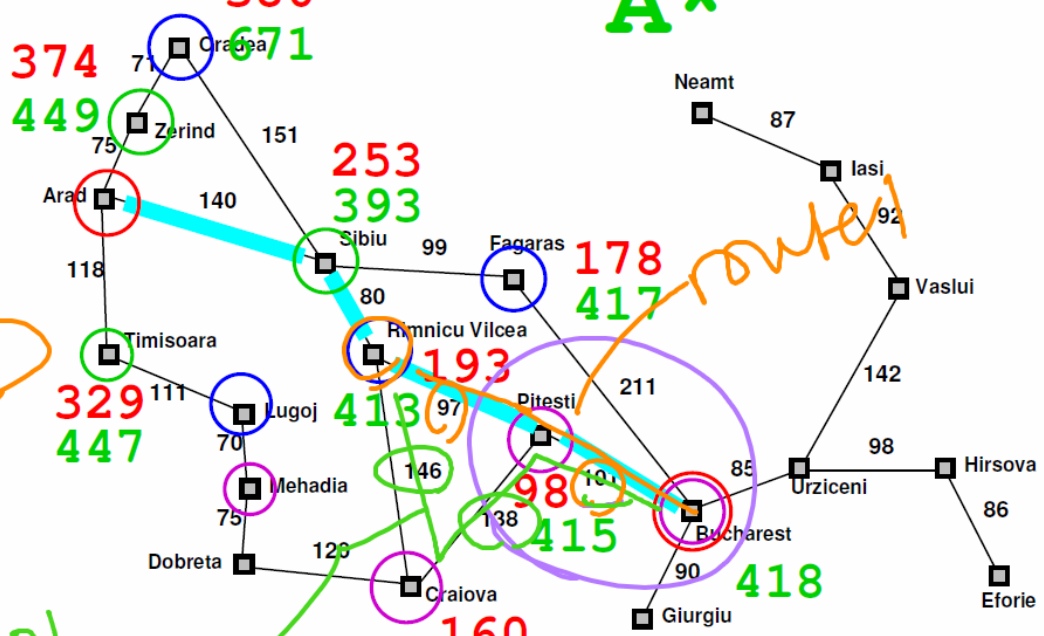
$\leq$

$h^*(p) = 101$

route 1 : 97+101 ≈ 200

Romania with step costs in km

A*



Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

$h(R) = 193$

$h^*(R) = 200$

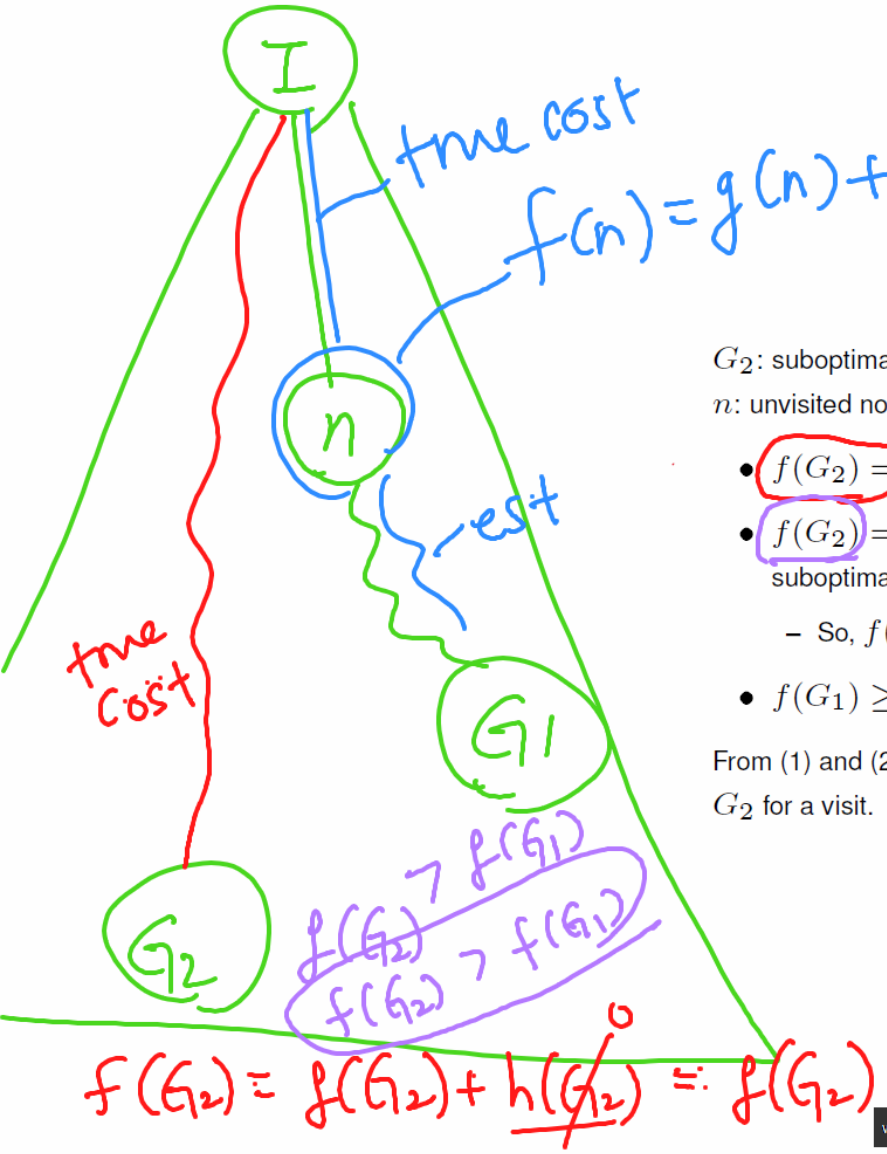
route 1

route 2 : 146 + 138 + 101 ≈ 380

route 2

Total Path Cost = 418

To find out the particular problem for that particular problem but here's the function values have this admissible property.



Optimality of A*

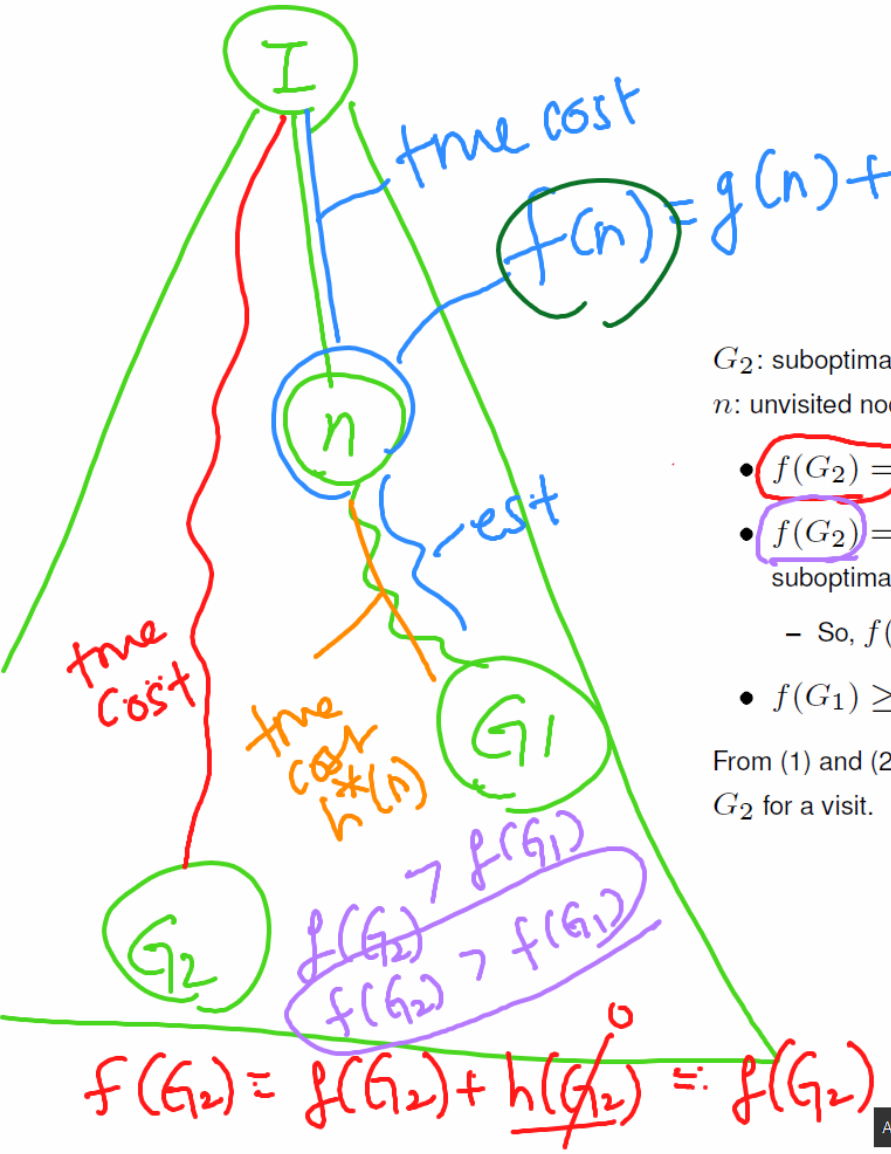
G_2: suboptimal goal in the node-list.

n: unvisited node on a shortest path to goal G_1

- $f(G_2) = h(G_2) + g(G_2) = g(G_2)$ since $h(G_2) = 0$
- $f(G_2) = g(G_2) > g(G_1) = f(G_1)$ since G_2 is suboptimal, and G_1 is a goal,
  - So, $f(G_2) > f(G_1)$: (1)
- $f(G_1) \geq f(n)$ since h is admissible : (2)

From (1) and (2), $f(G_2) > f(G_1) \geq f(n)$, so A* will never select G_2 for a visit.

which is H.



Optimality of A*

G_2: suboptimal goal in the node-list.

n: unvisited node on a shortest path to goal G_1

- $f(G_2) = h(G_2) + g(G_2) = g(G_2)$ since $h(G_2) = 0$
- $f(G_2) = g(G_2) > g(G_1) = f(G_1)$ since G_2 is suboptimal, and G_1 is a goal,
  - So, $f(G_2) > f(G_1)$: (1)
- $f(G_1) \geq f(n)$ since h is admissible : (2)

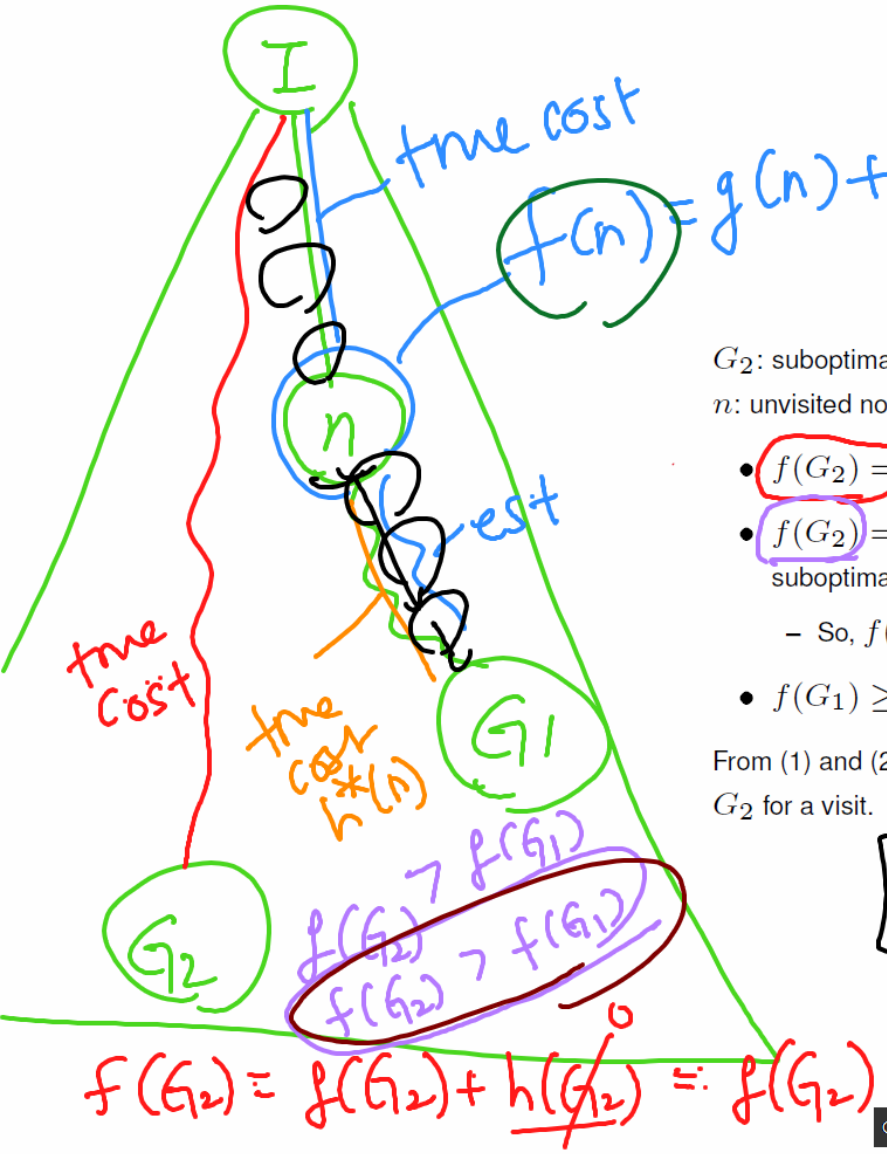
From (1) and (2), $f(G_2) > f(G_1) \geq f(n)$, so A* will never select G_2 for a visit.

h(n) is admissible

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

$f(n) \leq f(G_1)$

And then from, from here.

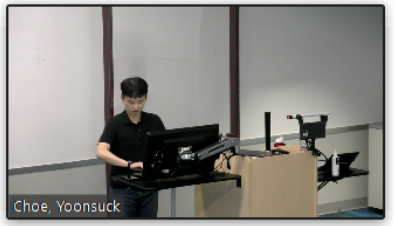


$f(n) = g(n) + h(n) \leq g(n) + h^*(n)$

$f(G_1)$

$h(n)$ is admissible

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



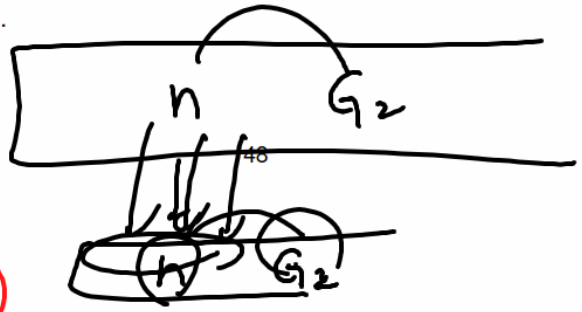
Optimality of A*

G_2: suboptimal goal in the node-list.

n: unvisited node on a shortest path to goal G_1

- $f(G_2) = h(G_2) + g(G_2) = g(G_2)$ since $h(G_2) = 0$
- $f(G_2) = g(G_2) > g(G_1) = f(G_1)$ since G_2 is suboptimal, and G_1 is a goal,
  - So, $f(G_2) > f(G_1)$: (1)
- $f(G_1) \geq f(n)$ since h is admissible : (2)

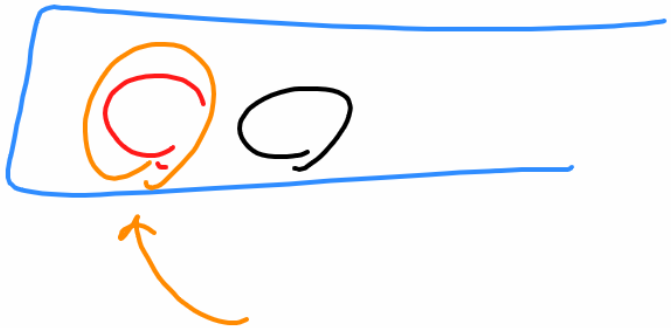
From (1) and (2), $f(G_2) > f(G_1) \geq f(n)$, so A* will never select G_2 for a visit.



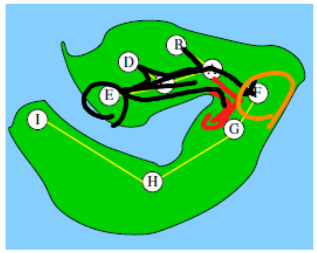
$f(n) \leq f(G_1) < f(G_2)$

$f(n) < f(G_2)$

Okay.



Optimality of A^*: Example



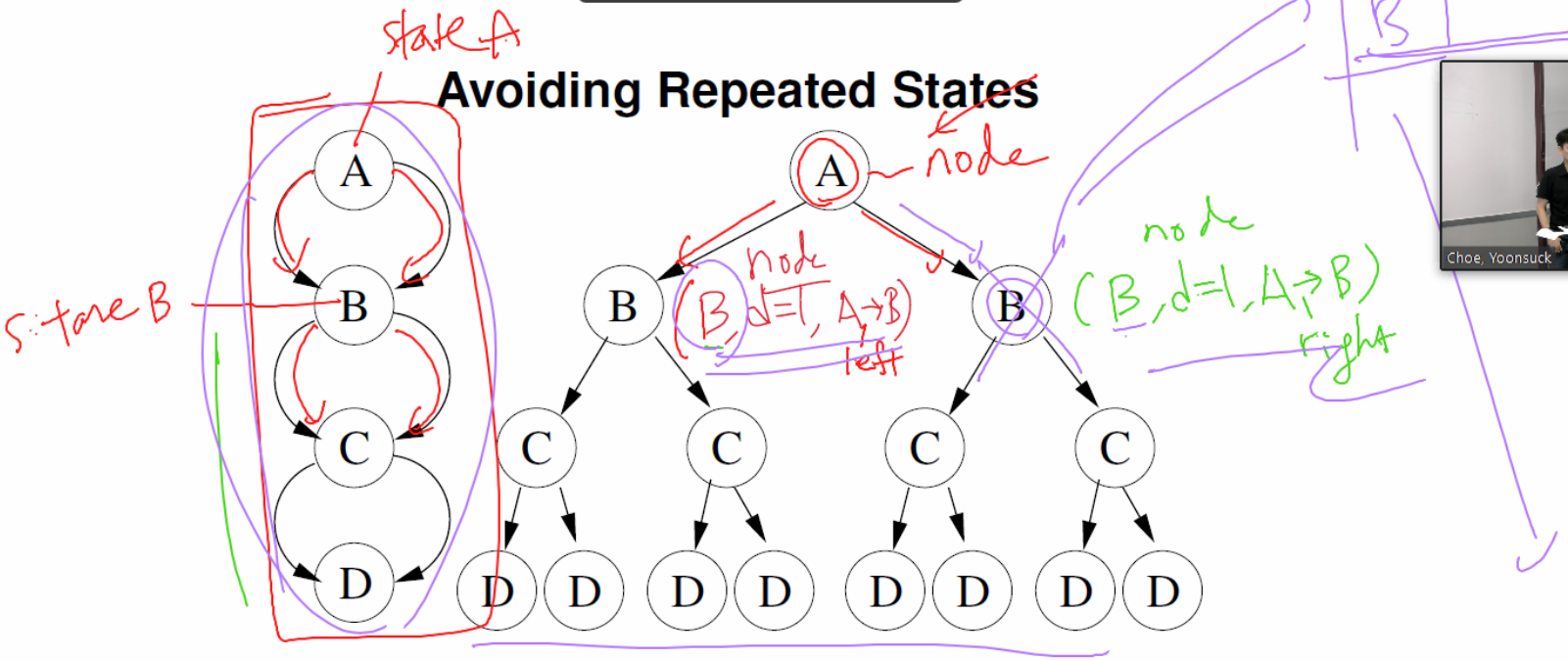
1. **Visit of parent disallowed**: search fails at nodes **B**, **D**, and **E**.
2. **Visit of parent allowed**: paths through nodes **B**, **D**, and **E** will have an inflated path cost $g(n)$, thus will become nonoptimal.

$$\underbrace{A \rightarrow C \rightarrow E \rightarrow C \rightarrow}_{\text{inflated path cost}} A \rightarrow F \rightarrow \ldots$$

B C D E C A F

So when you compare these two they're the same state, but this is much more efficient and.

Avoiding Repeated States



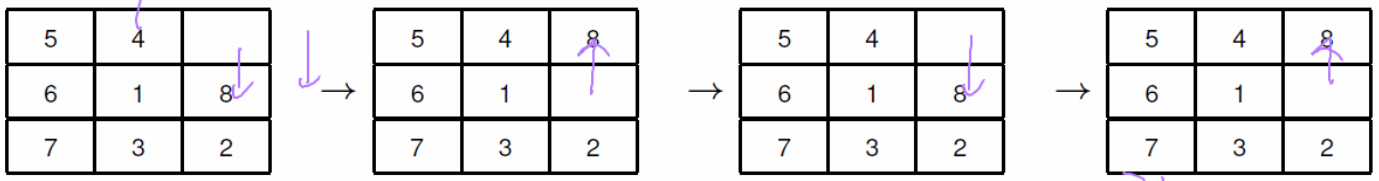
Same state can appear in different nodes.

Repeated states can be devastating in search problems.

- Common cases: problems with reversible operators → search space becomes infinite
- One approach: find a spanning tree of the graph

puzzle to the next, a puzzle state. They use your pride operating backward then you can go back and forth, back and forth, we can go in.

Avoiding Repeated States: Strategies

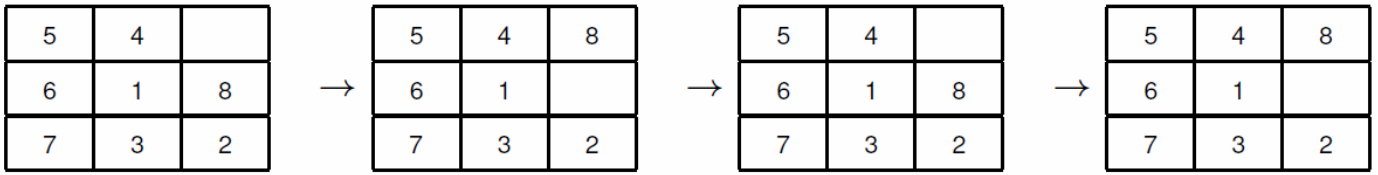


- Do not return to the node's parent
- Avoid cycles in the path (this is a huge theoretical problem in its own right)
- Do not generate states that you generated before: use a hash table to make checks efficient

How to avoid storing every state? Would using a short signature (or a checksum) of the full state description help?

So look these become an issue.

Avoiding Repeated States: Strategies



- Do not return to the node's parent
- Avoid cycles in the path (this is a huge theoretical problem in its own right)
- Do not generate states that you generated before: use a hash table to make checks efficient

How to avoid storing every state? Would using a short signature (or a checksum) of the full state description help?

get, get some kind of a shorter hash of that. And then just say the hash value, but.

Overview: Informed search, Iterative Improvement

- Best-first search
- Heuristic function
- Greedy best-first search
- A*
- Designing good heuristics
- *IDA**
- Iterative improvement algorithms
  1. Hill-climbing
  2. Simulated annealing

But the only thing now that is different, would be ida star was kind of similar in spirit, but Ida star is recursive Lee recursive Lee defined.

Overview: Informed search, Iterative Improvement

- Best-first search
- Heuristic function
- Greedy best-first search
- A*
- Designing good heuristics
- *IDA**
- Iterative improvement algorithms
  1. Hill-climbing
  2. Simulated annealing

So if you use this evaluation function to define King function, then that becomes informed such.

Overview: Informed search, Iterative Improvement

- Best-first search
- Heuristic function
- Greedy best-first search
- A*
- Designing good heuristics
- *IDA**
- Iterative improvement algorithms
  1. Hill-climbing
  2. Simulated annealing