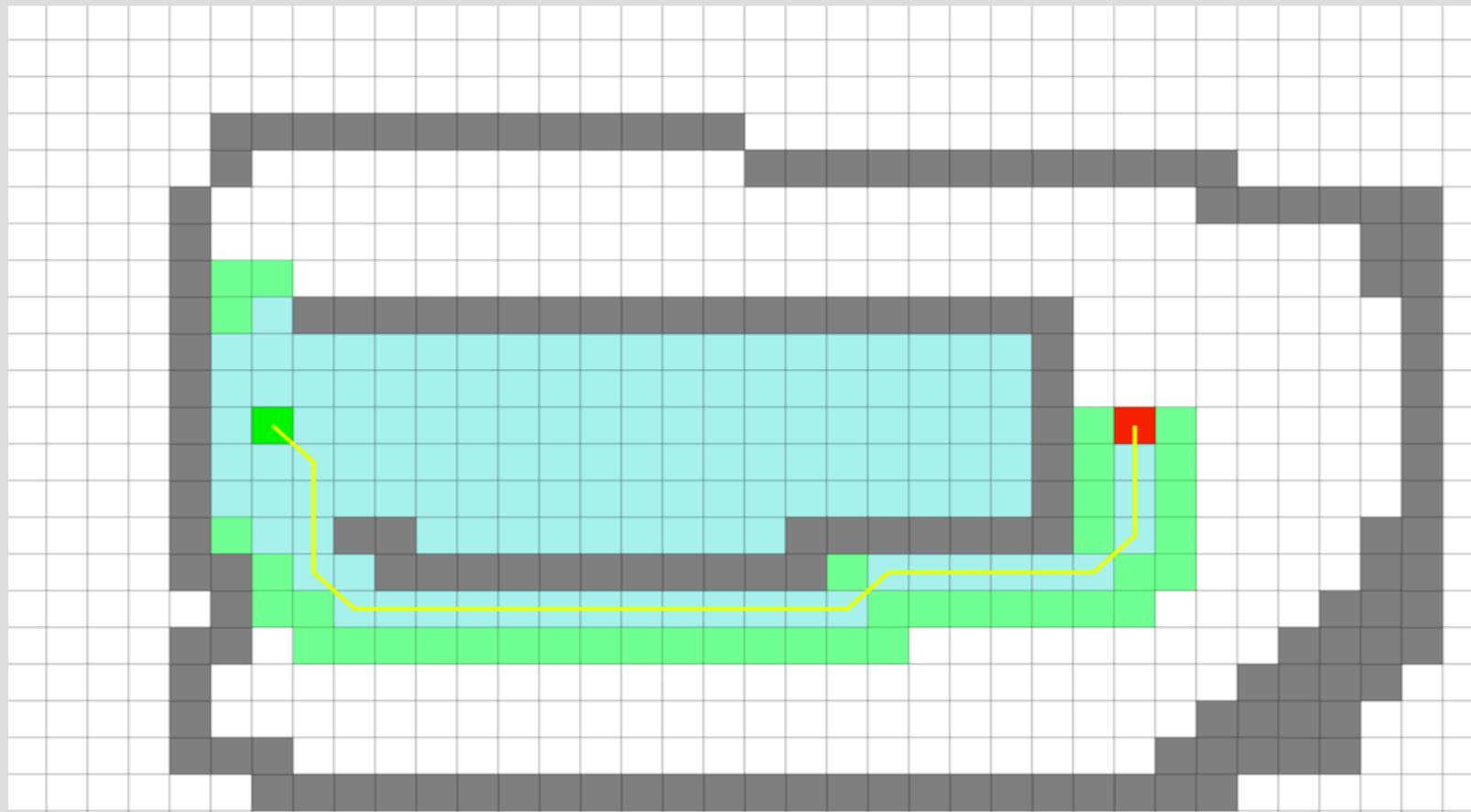


# Informed Search (Ch. 3.5-3.6)



length: 28.66  
time: 6.0000ms  
operations: 314

# Announcements

Writing 1 due tonight

HW 2 posted soon

My Thursday office hours are push back an hour (one time only): 1pm - 3pm

# Informed search

In uninformed search, we only had the node information (parent, children, cost of actions)

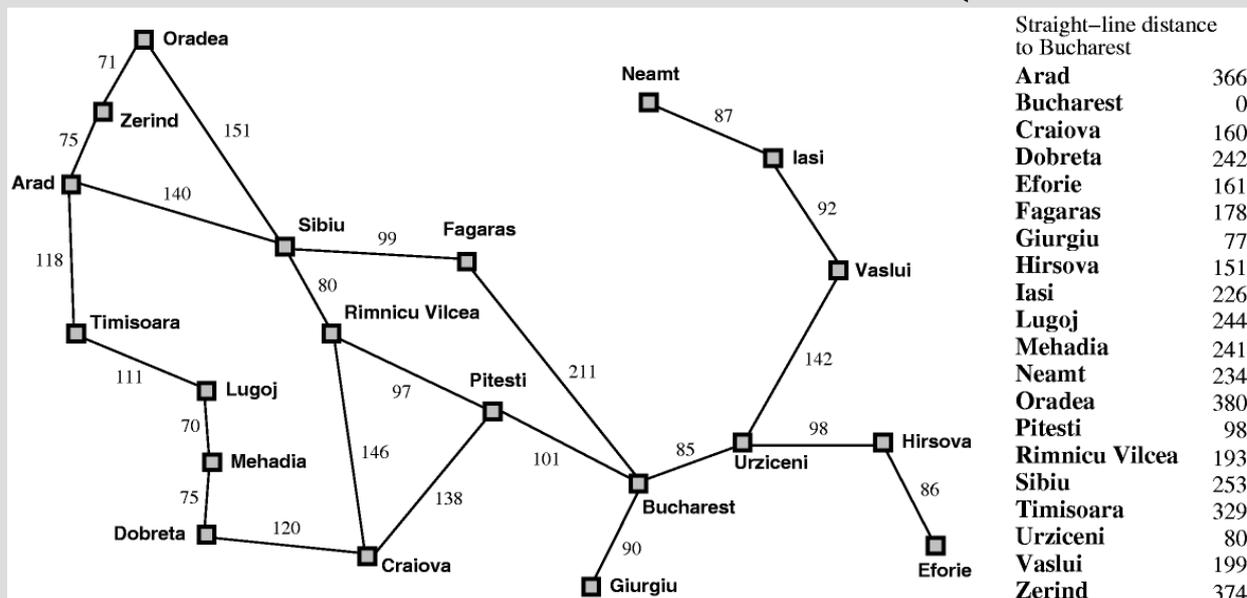
Now we will assume there is some additional information, we will call a heuristic that estimates the distance to the goal

Previously, we had no idea how close we were to goal, simply how far we had gone already

# Greedy best-first search

To introduce heuristics, let us look at the tree version of greedy best-first search

This search will simply repeatedly select the child with the lowest heuristic (cost to goal est.)



# Greedy best-first search

This finds the path: Arad -> Sibiu -> Fagaras -> Bucharest

However, this greedy approach is not optimal, as that is the path: Arad -> Sibiu -> Rimnicu Vilcea -> Pitesti -> Bucharest

In fact, it is not guaranteed to converge (if a path reaches a dead-end, it will loop infinitely)

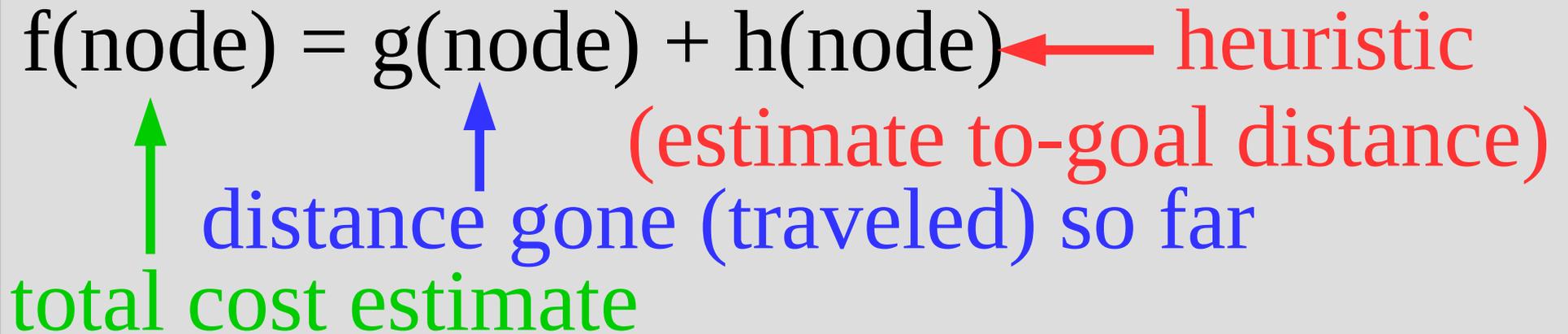
# A\*

We can combine the distance traveled and the estimate to the goal, which is called A\* (a star)

- The method goes: (red is for “graphs”)
- initialize **explored**= $\{\}$ , fringe= $\{[\text{start}, f(\text{start})]\}$
1. Choose  $C = \text{argmin}(f\text{-cost})$  in fringe
  2. Add **or update** C's children to fringe, with associated f-value, remove C from fringe
  3. **Add C to explored**
  4. Repeat 1. until  $C == \text{goal}$  or fringe empty

# A\*

$$f(\text{node}) = g(\text{node}) + h(\text{node})$$



total cost estimate

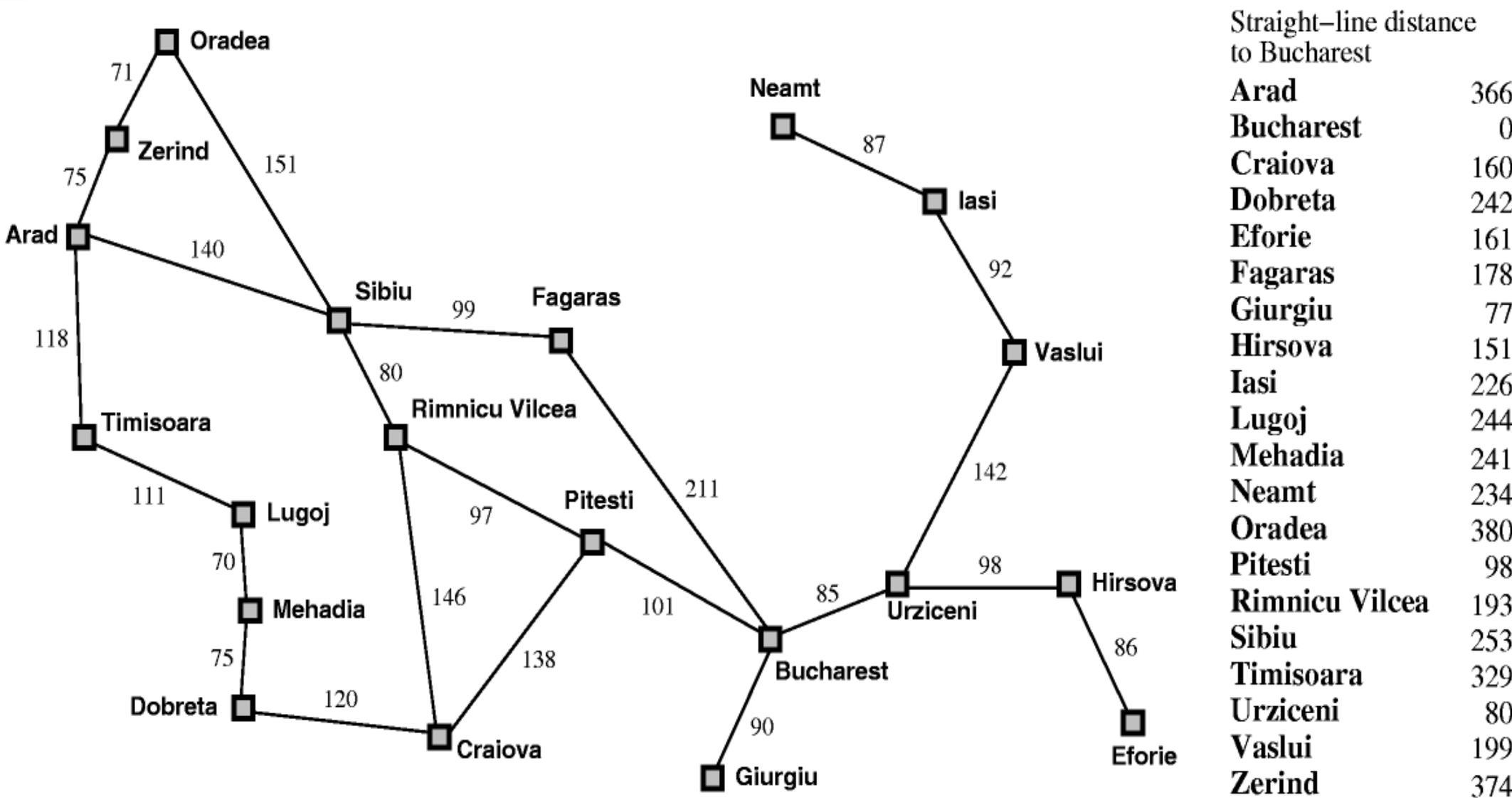
distance gone (traveled) so far

heuristic (estimate to-goal distance)

We will talk more about what heuristics are good or should be used later

Priority queues can be used to efficiently store and insert states and their f-values into the fringe

# A\*



# A\*

Step: Fringe (argmin)

0: [Arad, 366]

1: [Zerind, 75+374],[Sibu, 140+253],[Timisoara, 118+329]

1: [Zerind, 449], [Sibu, 393], [Timisoara, 447]

2: [Fagaras, 140+99+178], [Rimmicu Vilcea, 140+80+193],  
[Zerind, 449], [Timisoara, 447], [Oradea, 140+151+380]

2: [Fagaras, 417], [Rimmicu Vilcea, 413], [Zerind, 449],  
[Timisoara, 447], [Oradea, 671]

3: [Craiova, 140+80+146+160], [Pitesti, 140+80+97+98],  
[Fagaras, 417], [Zerind, 449], [Timisoara, 447], [Oradea, 671]

3: [Craiova, 526], [Pitesti, 415], [Fagaras, 417], [Zerind, 449],  
[Timisoara, 447], [Oradea, 671]

4: ... on next slide

# A\*

- 4: [Craiova from Rimnicu Vilcea, 526], [Fagaras, 417],  
[Zerind, 449], [Timisoara, 447], [Oradea, 671],  
[Craiova from Pitesti, 140+80+97+138+160],  
[Bucharest from Pitesti, 140+80+97+101+0]
- 4: [Craiova from Rimnicu Vilcea, 526], [Fagaras, 417],  
[Zerind, 449], [Timisoara, 447], [Oradea, 671],  
[Craiova from Pitesti, 615], [Bucharest from Pitesti, 418]
- 5: [Craiova from Rimnicu Vilcea, 526], [Zerind, 449],  
[Timisoara, 447], [Oradea, 671], [Craiova from Pitesti, 615]  
[Bucharest from Pitesti, 418],  
[Bucharest from Fagaras, 140+99+211+0 = 450]

Goal!

# A\*

You can choose multiple heuristics (more later) but good ones skew the search to the goal

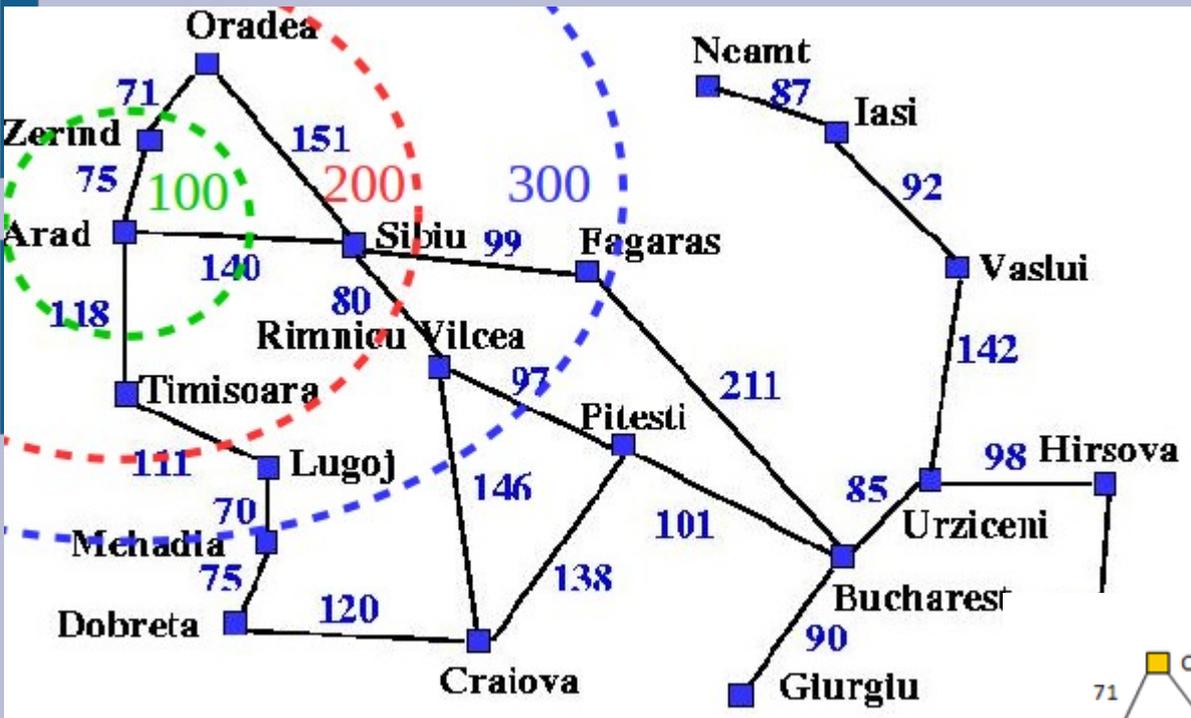
You can think circles based on f-cost:

-if  $h(\text{node}) = 0$ , f-cost are circles

-if  $h(\text{node}) = \text{very good}$ , f-cost long and thin ellipse

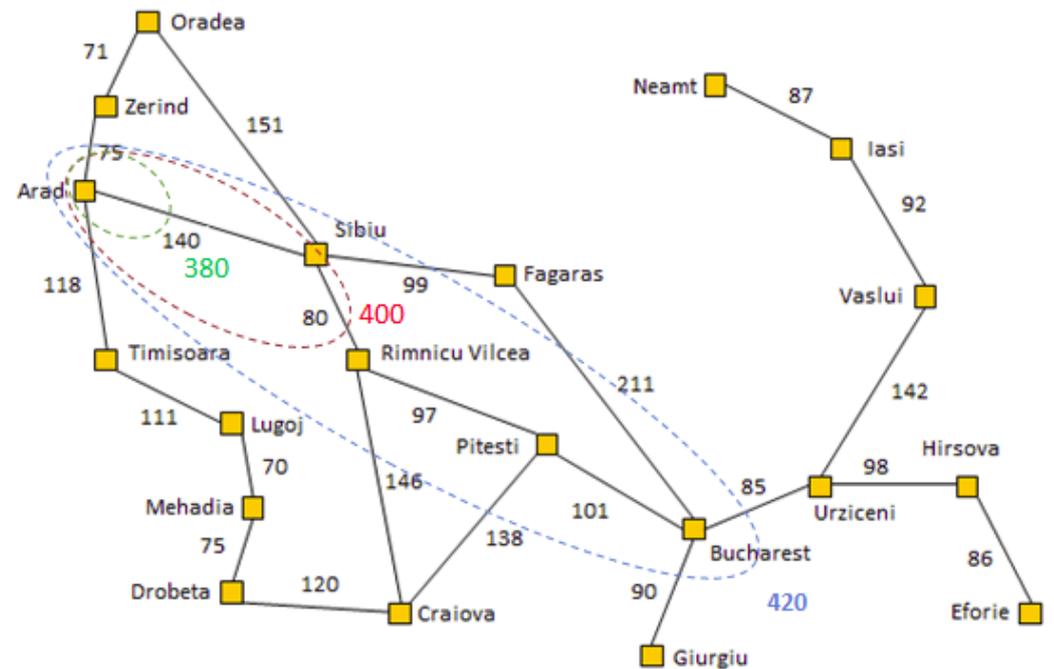
This can also be thought of as topographical maps (in a sense)

# A\*



$h(\text{node}) = 0$   
(bad heuristic, no goal guidance)

$h(\text{node}) = \text{straight line distance}$   
(good heuristic)



# A\*

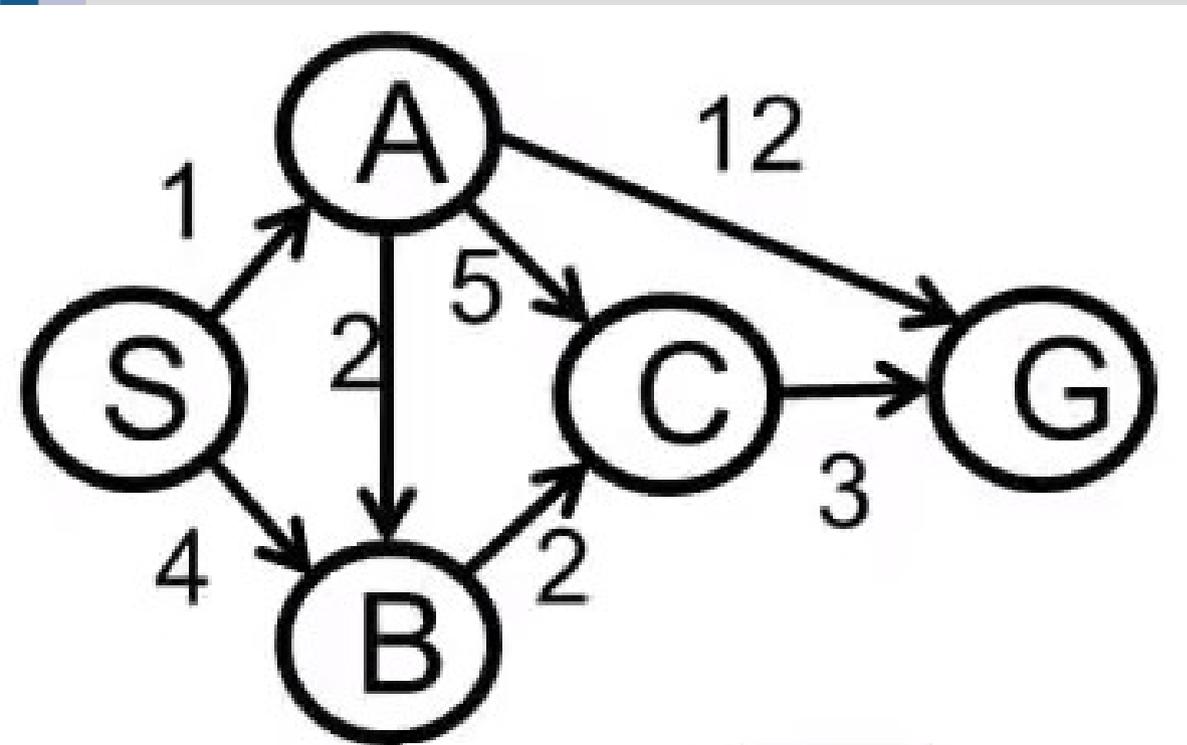
Good heuristics can remove “bad” sections of the search space that will not be on any optimal solution (called pruning)

A\* is optimal and in fact, no optimal algorithm could expand less nodes (optimally efficient)

However, the time and memory cost is still exponential (memory tighter constraint)

# A\*

You do it! Find path S → G



State	H
S	7
A	6
B	2
C	1
G	0

Arrows show children (easier for you)

(see: <https://www.youtube.com/watch?v=sAoBeujec74> )

# Iterative deepening A\*

You can combine iterative deepening with A\*

Idea:

1. Run DFS in IDS, but instead of using depth as cutoff, use f-cost
2. If search fails to find goal, increase f-cost to next smallest seen value (above old cost)

Pros: Efficient on memory

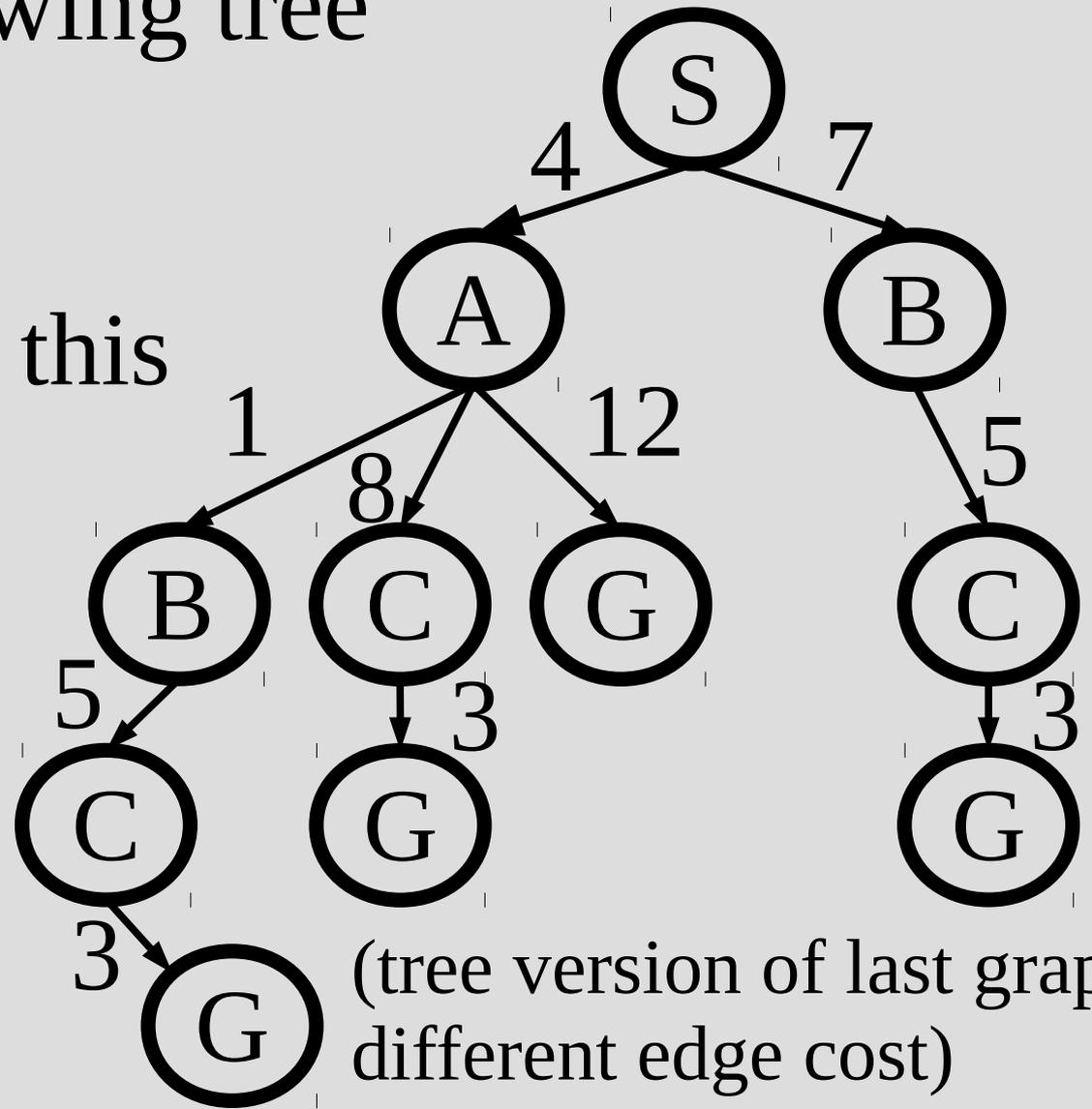
Cons: Large (LARGE) amount of re-searching

# Iterative deepening A\*

Consider the following tree and heuristic

Let's run IDA\* on this

State	H
S	7
A	6
B	2
C	1
G	0



# Iterative deepening A\*

State	H
S	7
A	6
B	2
C	1
G	0

Iterative deepening, round 1:  
Limit =  $h(s) = 7$

Run DFS expanding nodes  
less (or =) limit

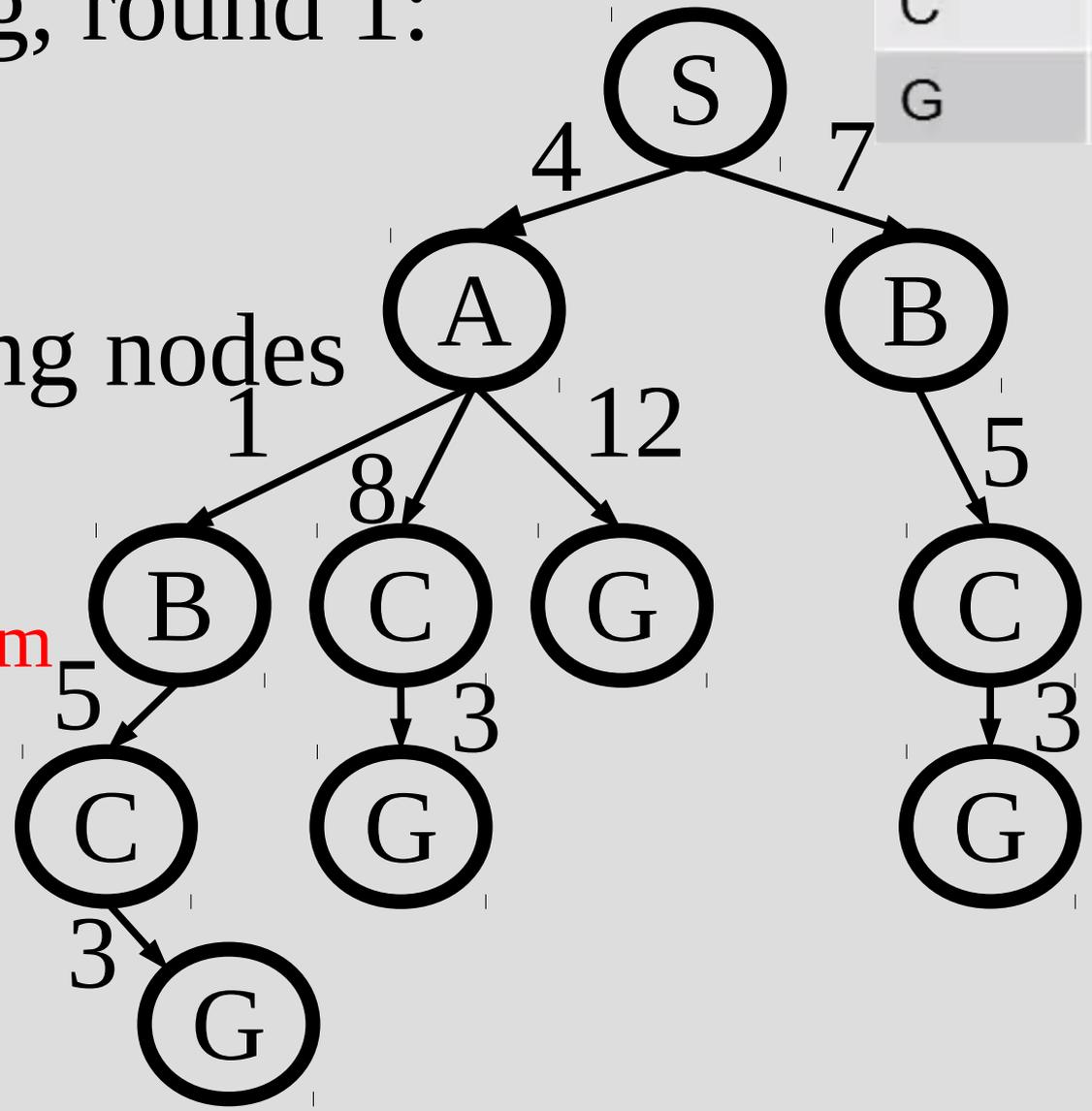
This is DFS FILO  
not finding minimum

Fringe:

1: (S,7)

2: (A,10), ~~(B,9)~~

3: ~~(A,10)~~



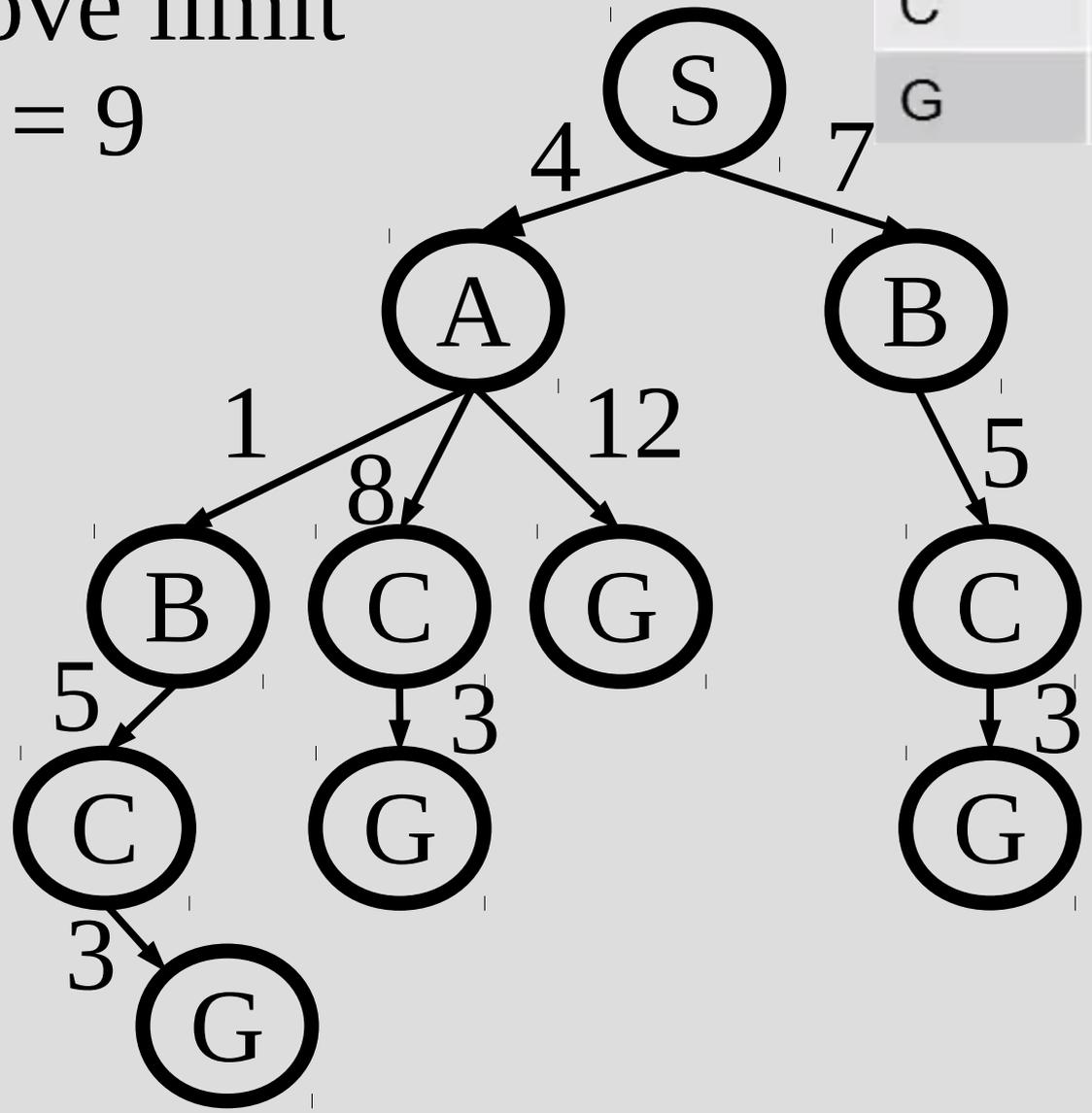
# Iterative deepening A\*

State	H
S	7
A	6
B	2
C	1
G	0

Smallest f-cost above limit  
in previous search = 9

New limit = 9

- 1: (S,7)
- 2: (A,10), (B,9)
- 3: (A,10), ~~(C,14)~~
- 4: ~~(A,10)~~



# Iterative deepening A\*

State	H
S	7
A	6
B	2
C	1
G	0

Smallest f-cost above limit  
in previous search = 10 = limit

1: (S,7)

2: (A,10), (B,9)

3: (A,10), ~~(C,14)~~

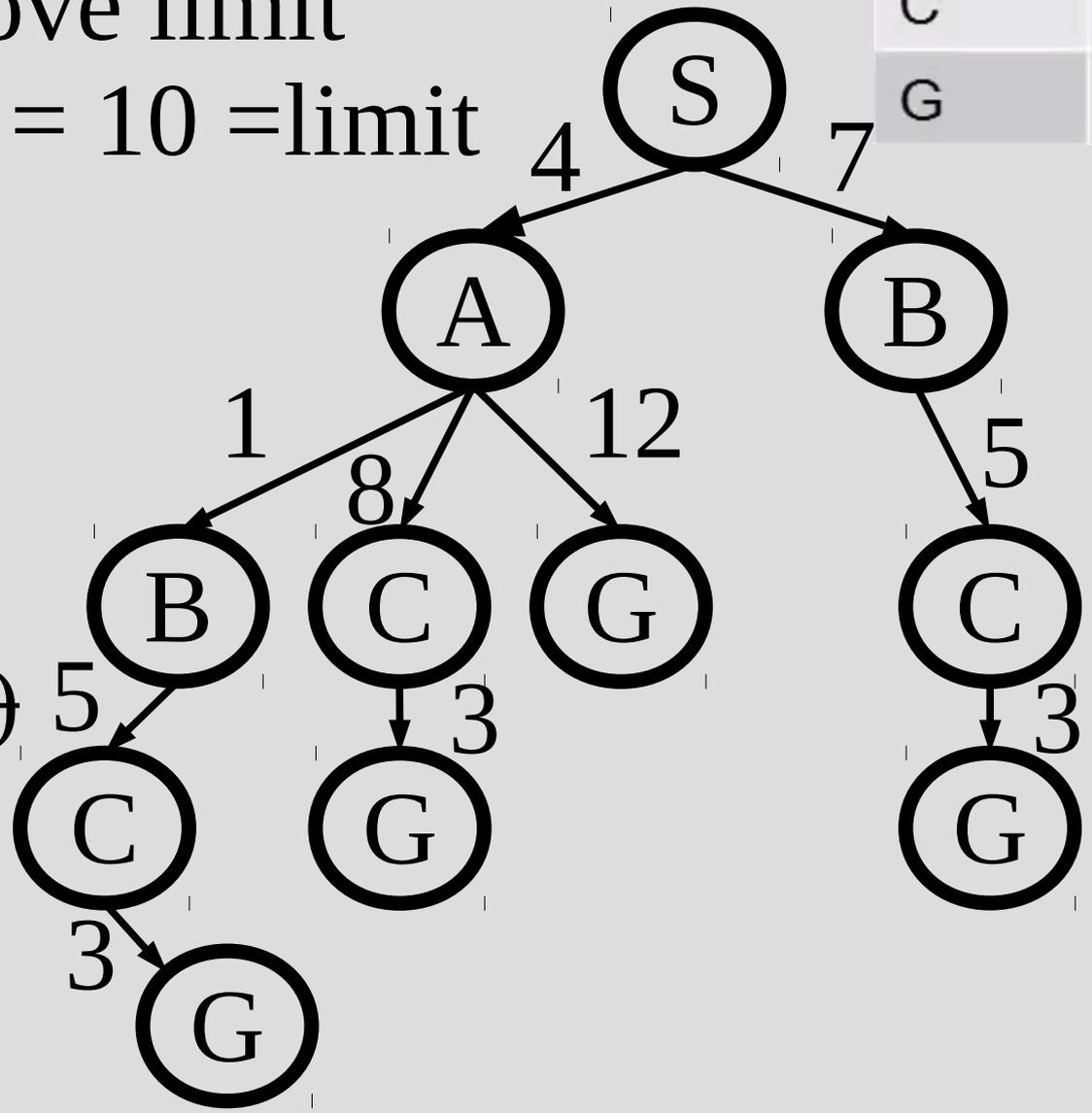
4: (A,10)

5: (B,7), (C,13), ~~(G,16)~~

6: (B,7), ~~(C,13)~~

7: (B,7)

8: ~~(C,11)~~



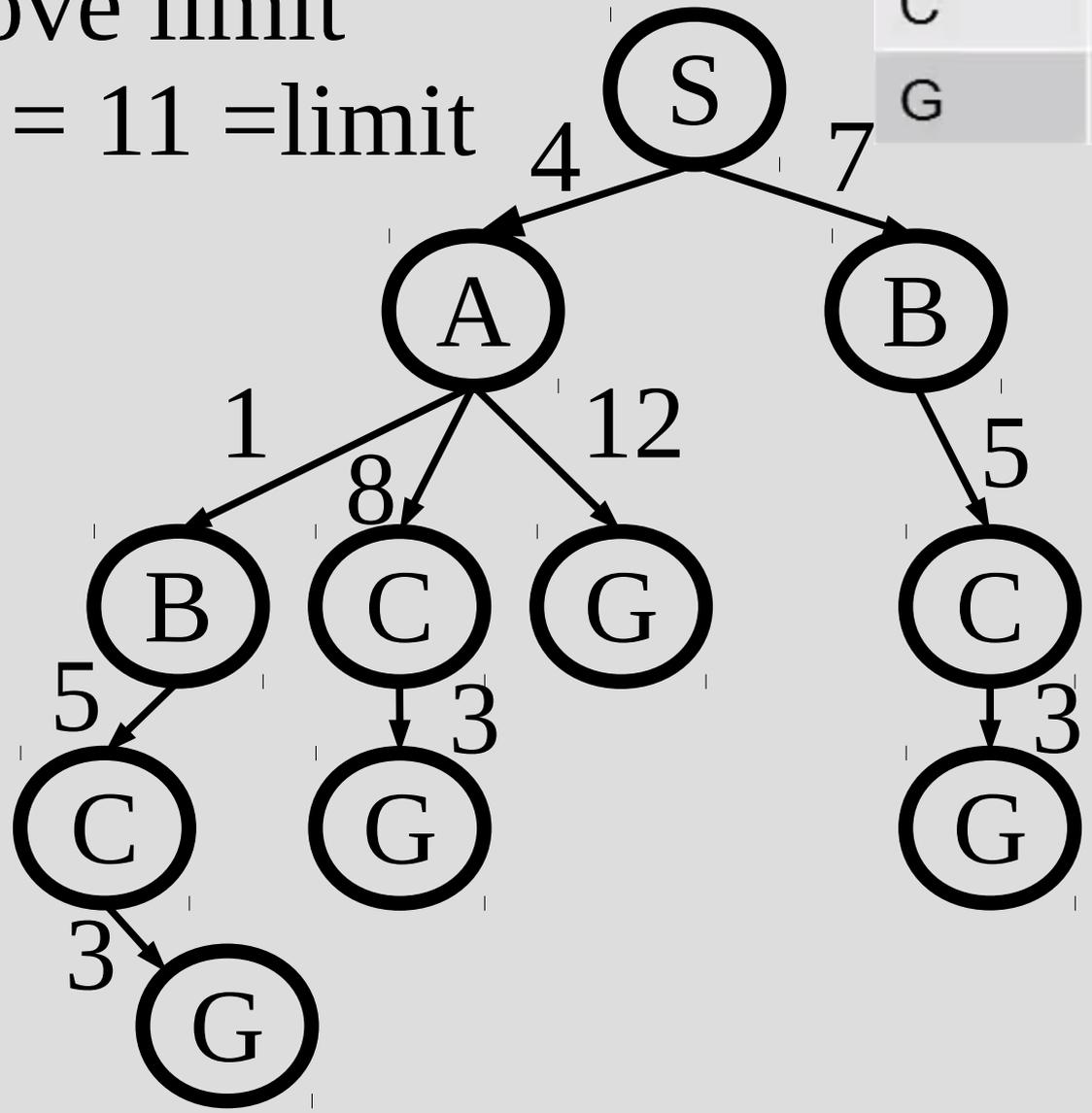
# Iterative deepening A\*

State	H
S	7
A	6
B	2
C	1
G	0

Smallest f-cost above limit  
in previous search = 11 = limit

... and repeat this  
process until goal  
is found

Since search is  
DFS, memory  
efficient

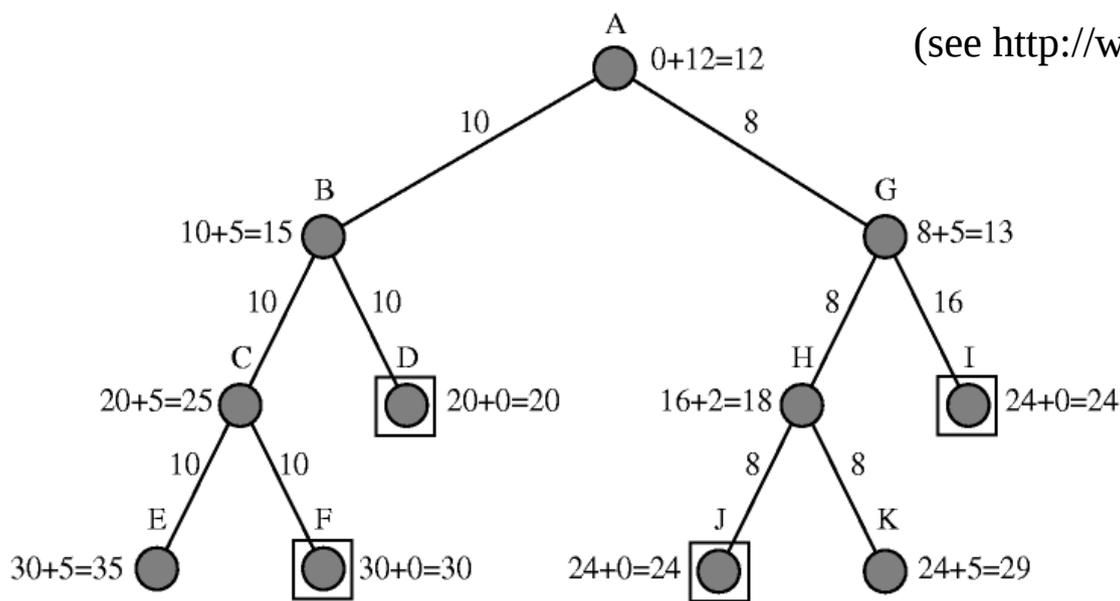


# SMA\*

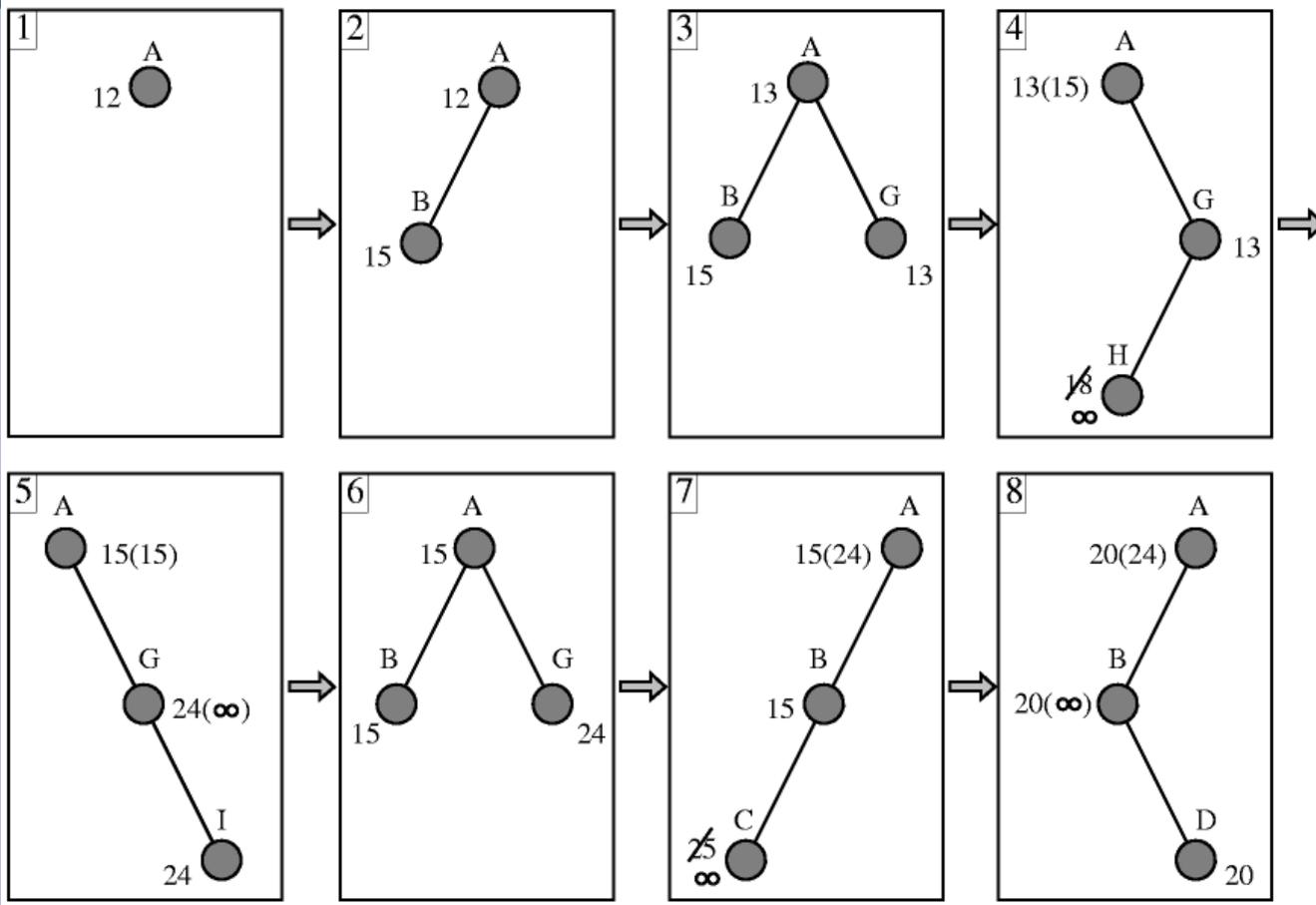
One fairly straight-forward modification to  $A^*$  is simplified memory-bounded  $A^*$  (SMA\*)

Idea:

1. Run  $A^*$  normally until out of memory
2. Let  $C = \operatorname{argmax}(\text{f-cost})$  in the leaves
3. Remove  $C$  but store its value in the parent  
(for re-searching)
4. Goto 1



Here assume you can only hold at most 3 nodes in memory



# SMA\*

SMA\* is nice as it (like A\*) find the optimal solution while keeping re-searching low (given your memory size)

IDA\* only keeps a single number in memory, and thus re-searches many times (inefficient use of memory)

Typically there is some time to memory trade-off