Informed Search (Ch. 3.5-3.6)

SHOPPING TEAMS
BAD:
TWO NON-NERDS
LET'S GET THAT ONE.
OKAY.
GOOD:
NON-NERD + NERD
LET'S GET THAT ONE.
WAIT. I THINK THE OTHER ONE MIGHT BE A BETTER DEAL.
OKAY, THAT ONE.
VERY BAD:
TWO NERDS
HOW ABOUT THAT ONE?
I THINK THE OTHER ONE MIGHT BE A BETTER DEAL...
HMM, I'M NOT SURE...
TWO HOURS LATER
I THINK OUR MAIN PROBLEM IS OUR UNCLEAR DEFINITION OF VALUE.
THAT IS NOT YOUR MAIN PROBLEM!

A*

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

heuristic

(estimate to-goal distance)

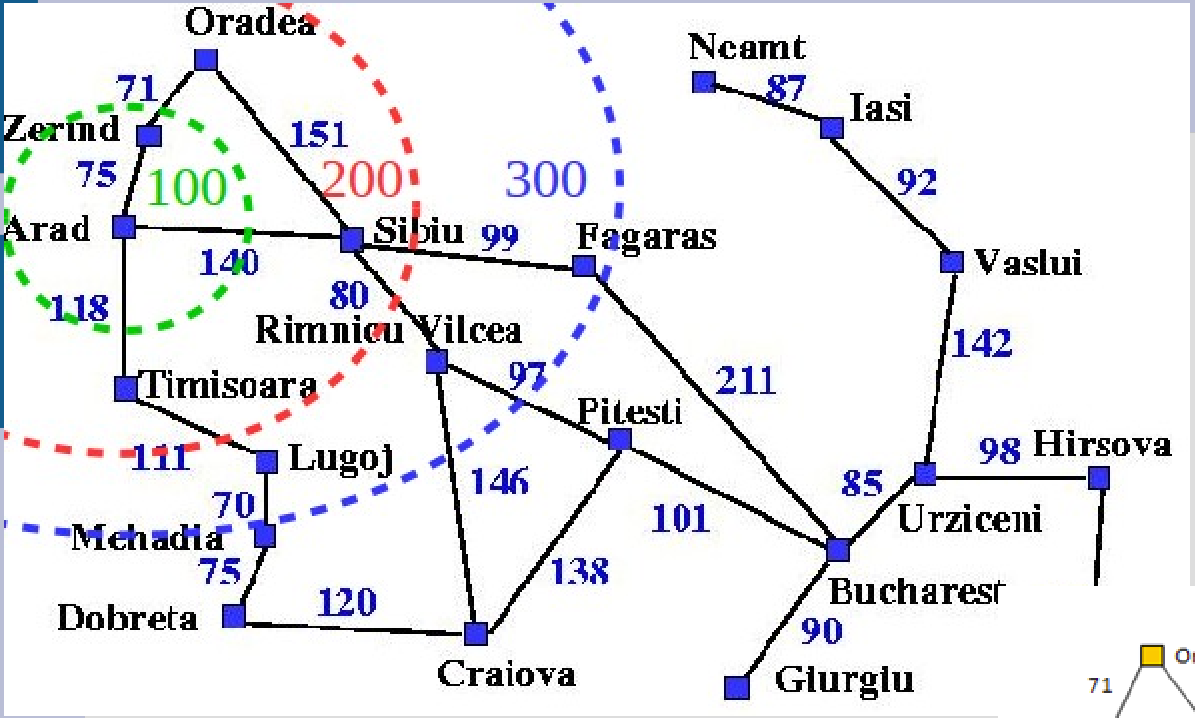
distance gone (traveled) so far

total cost estimate

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

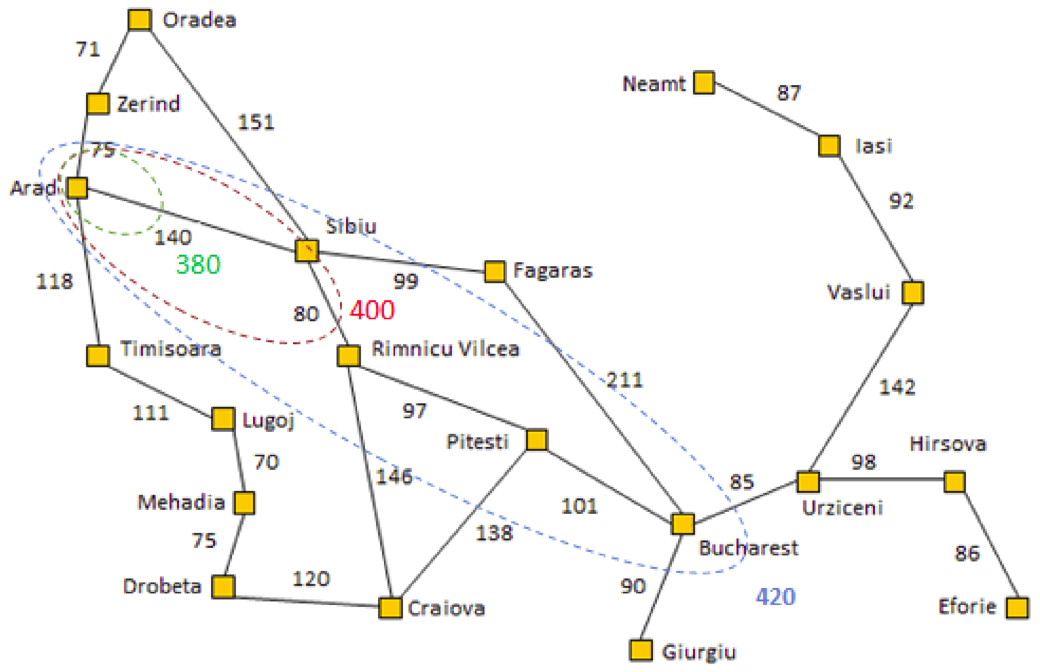
Priority queues can be used to efficiently store and insert states and their f-values into the fringe

A*



h(node) = 0
(bad heuristic, no goal guidance)

h(node) = 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)

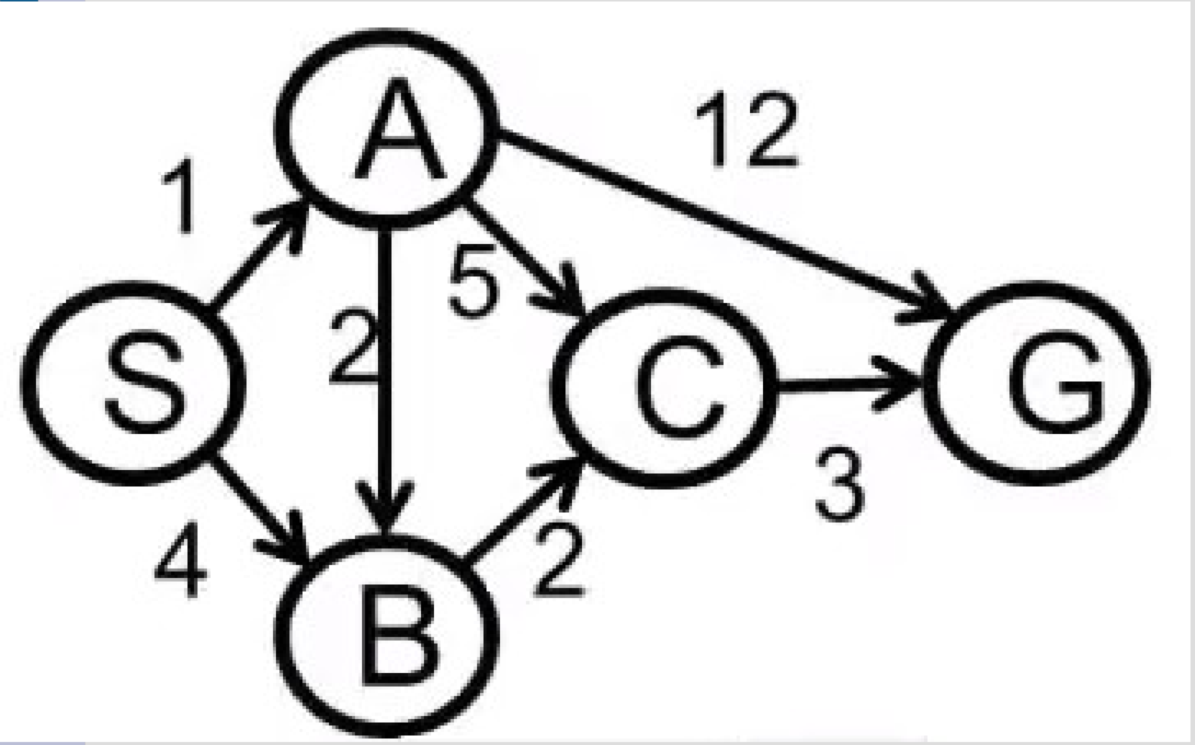
"some restrictions may apply"

A* can be optimal and in fact, no optimal alg. 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)

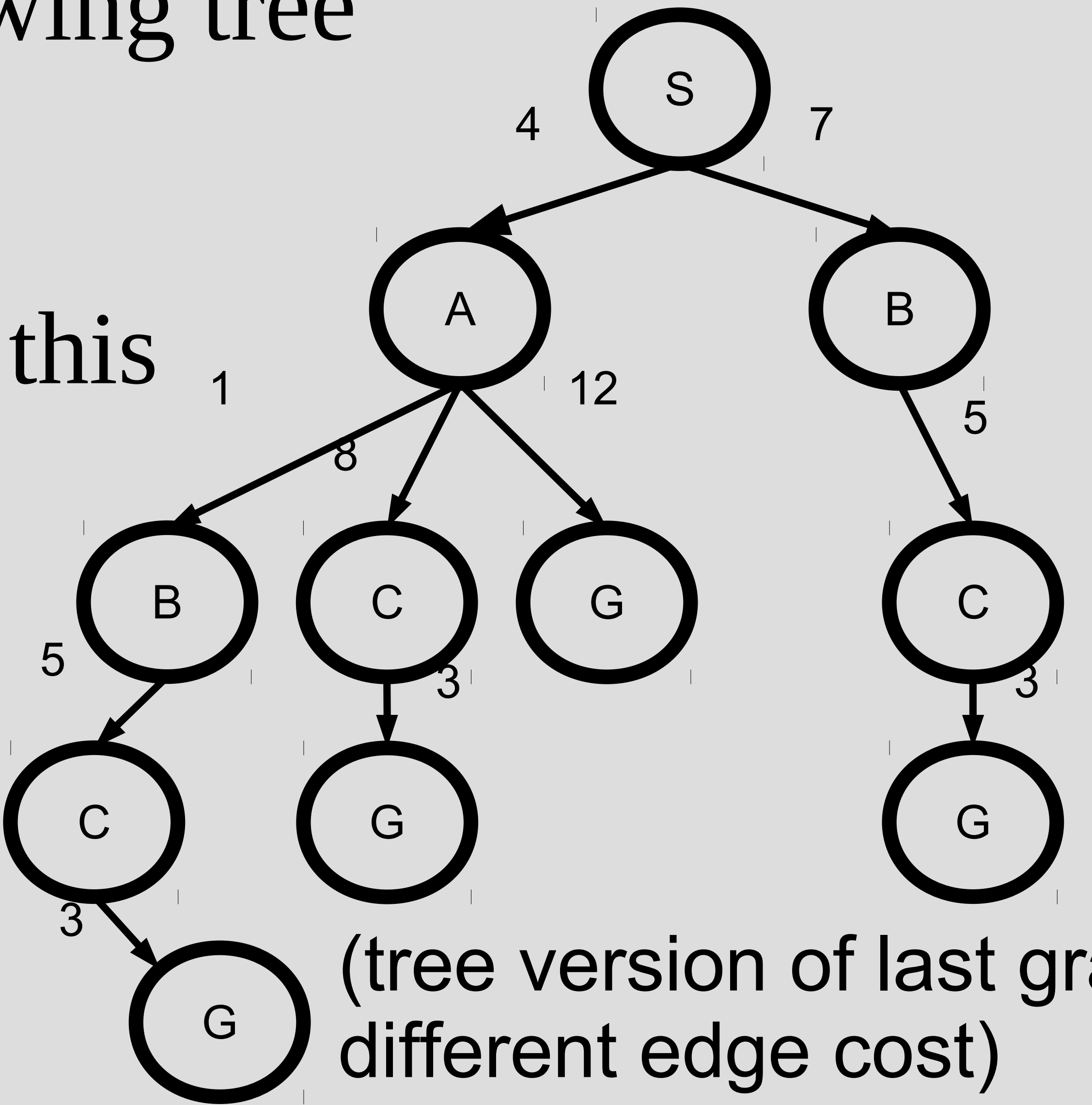
Pros: Efficient on memory
Cons: Large (LARGE) amount of re-searching

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



(tree version of last grap
different edge cost)

Iterative deepening A*

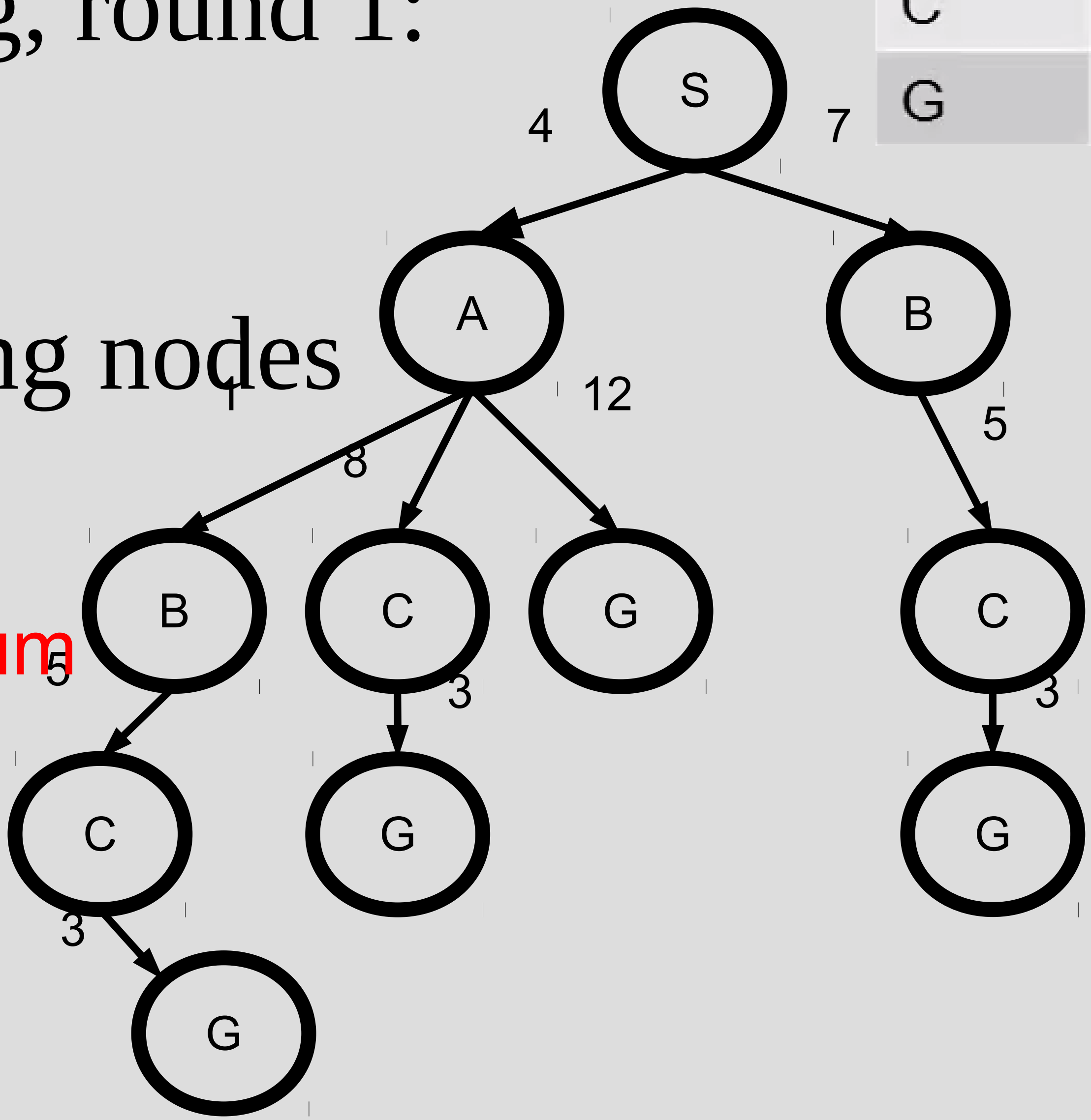
State	H
S	7
A	6
B	2
C	1
G	0

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)~~

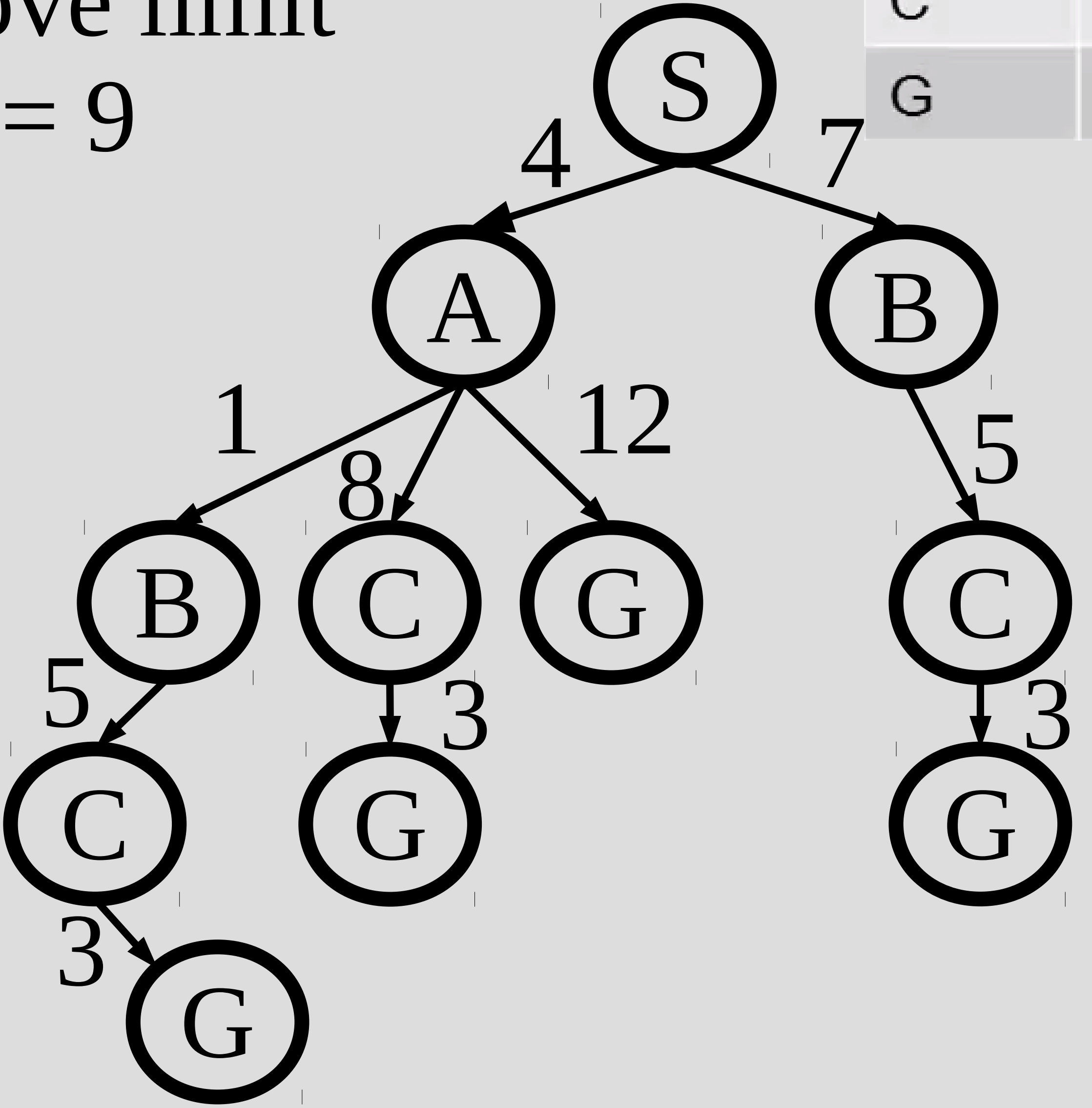


Iterative deepening A*

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: <u>(S,7)</u>

2: (A,10), <u>(B,9)</u>

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

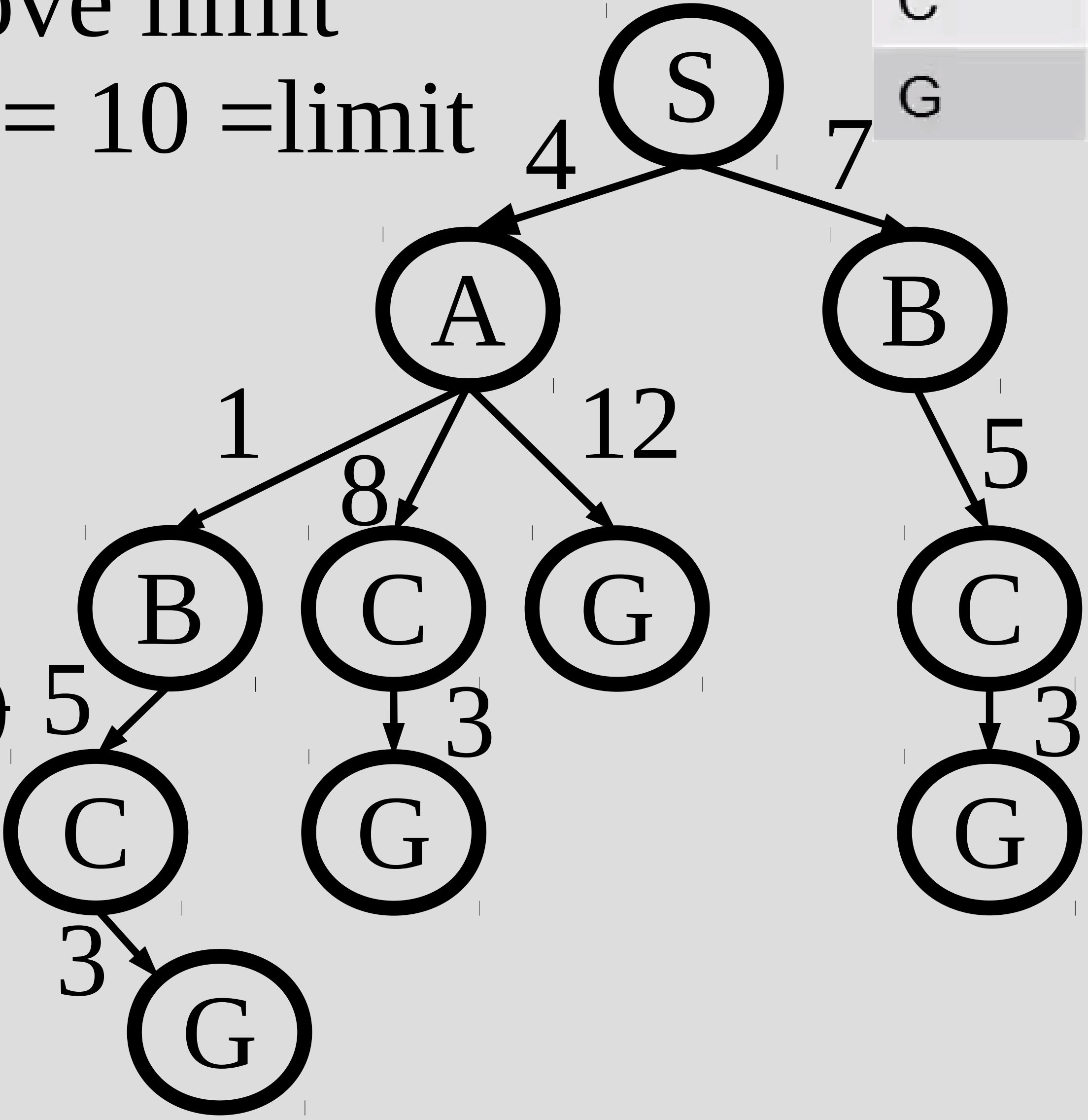
4: <u>(A,10)</u>

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

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

7: <u>(B,7)</u>

8: ~~(C,11)~~



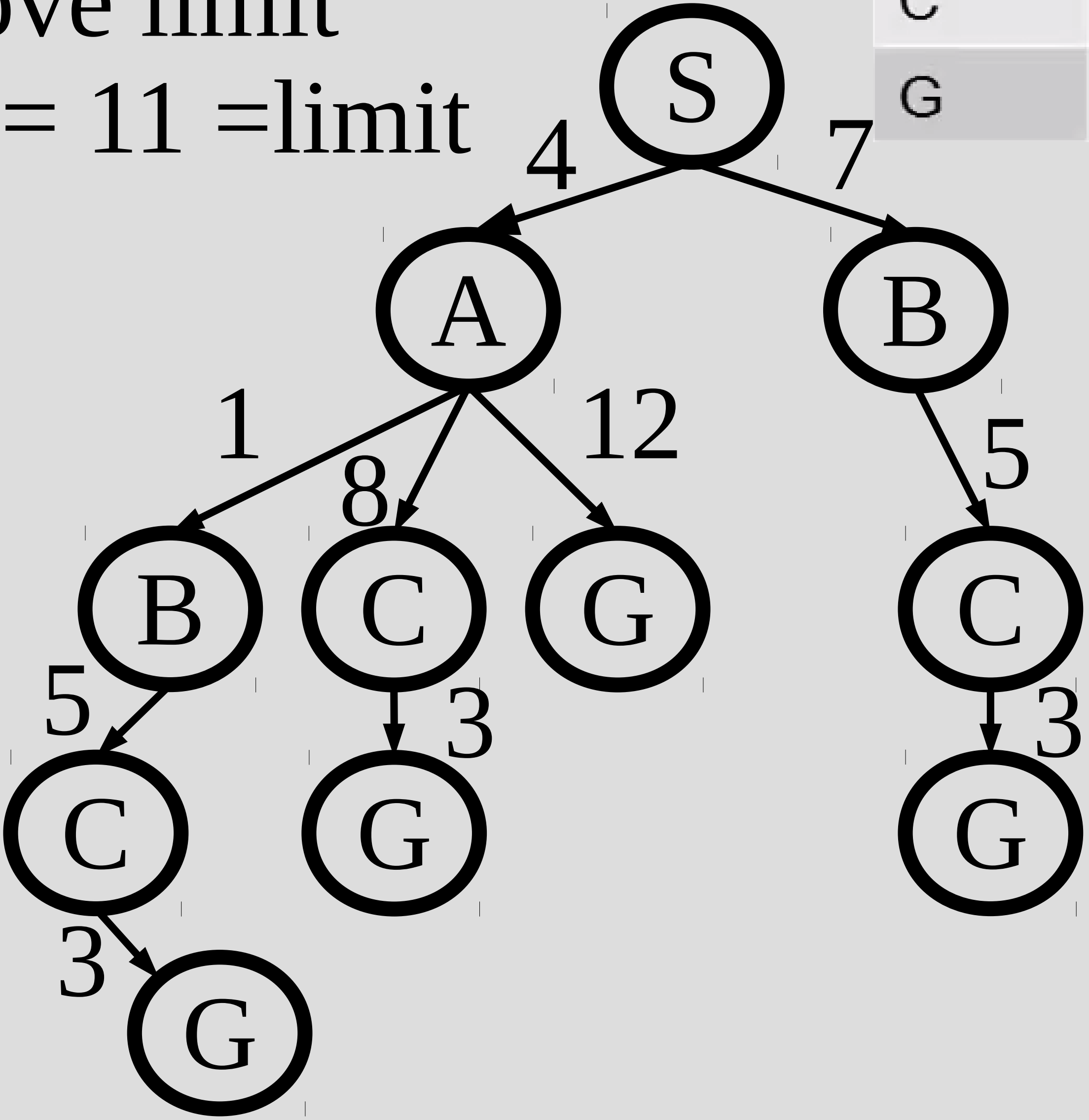
Iterative deepening A*

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



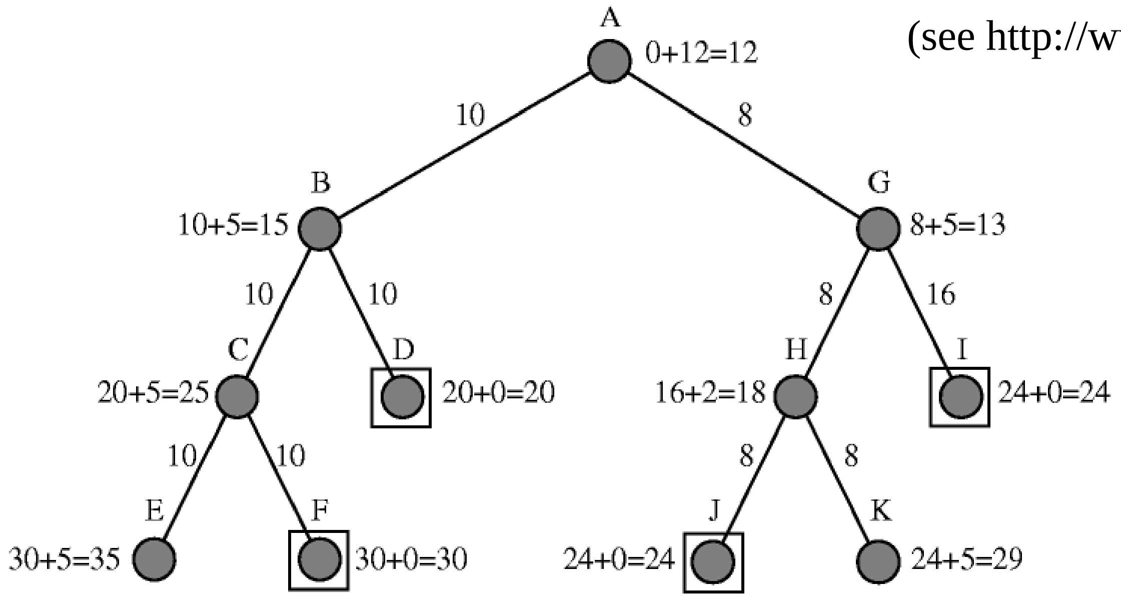
SMA*

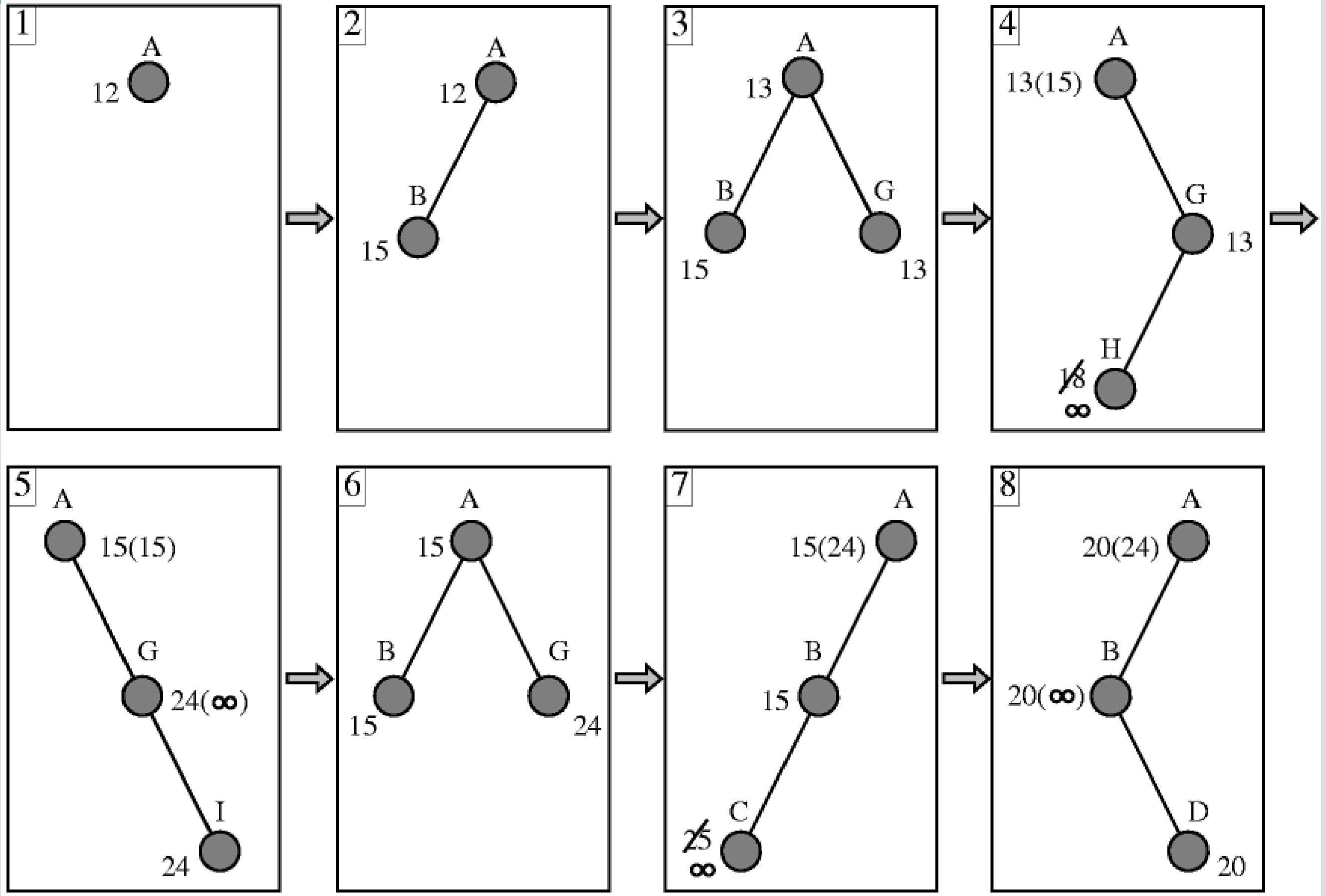
One fairly straight-forward modification to A* is <u>simplified memory-bounded A*</u> (SMA*)

Idea:

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

(see http://www.massey.ac.nz/~mjjohnso/notes/59302/l04.html)





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

Heuristics

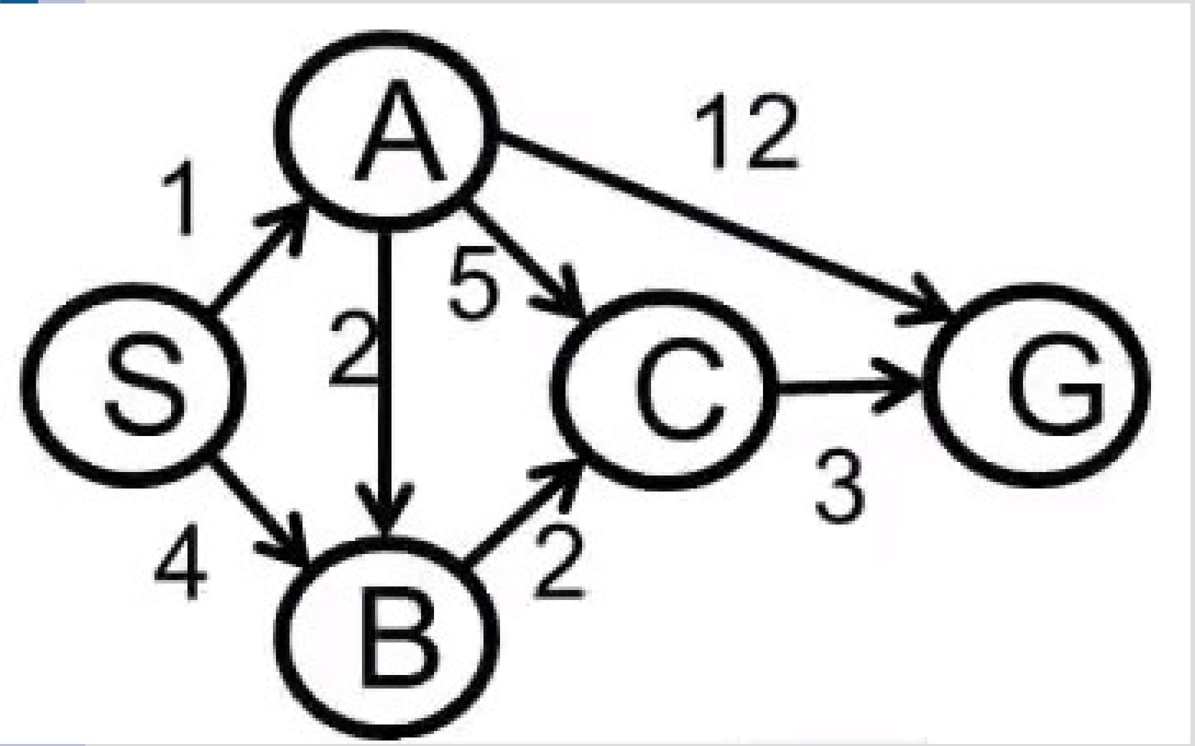
However, for A* to be optimal the heuristic h(node) needs to be...

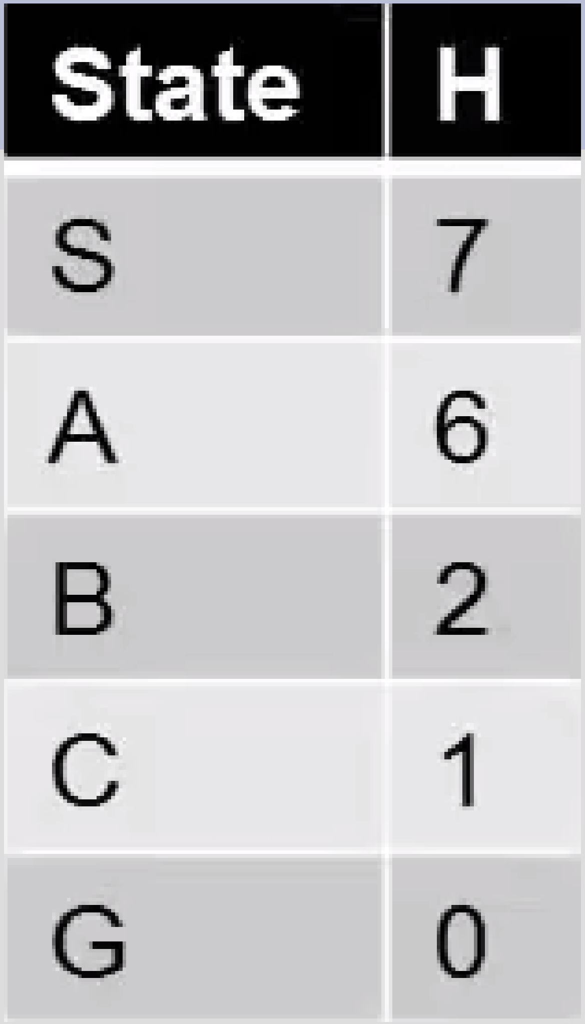
For trees: <u>admissible</u> which means:
- h(node) $\underline{<}$ optimal path from h to goal
- (i.e. h(node) is an underestimate of cost)

For graphs: <u>consistent</u> which means:
- h(node) $\underline{<}$ cost(node to child) + h(child)
- (i.e. triangle inequality holds true)
- (i.e. along any path, f-cost increases)

A*

Remember this?





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

Not consistent (between S and B)

Heuristics

Consistent heuristics are always admissible
-Requirement: h(goal) = 0

Consistency is a strong requirement, so an admissible heuristics **might not** be consistent, but a consistent heuristic is always admissible

A* is guaranteed to find optimal solution if the heuristic is admissible for trees (consistent for graphs)

Heuristics

In our example, the h(node) was the straight line distance from node to goal

This is an underestimate as physical roads cannot be shorter than this
(it also satisfies the triangle inequality)

Thus this heuristic is admissible
(and consistent)

Relaxation

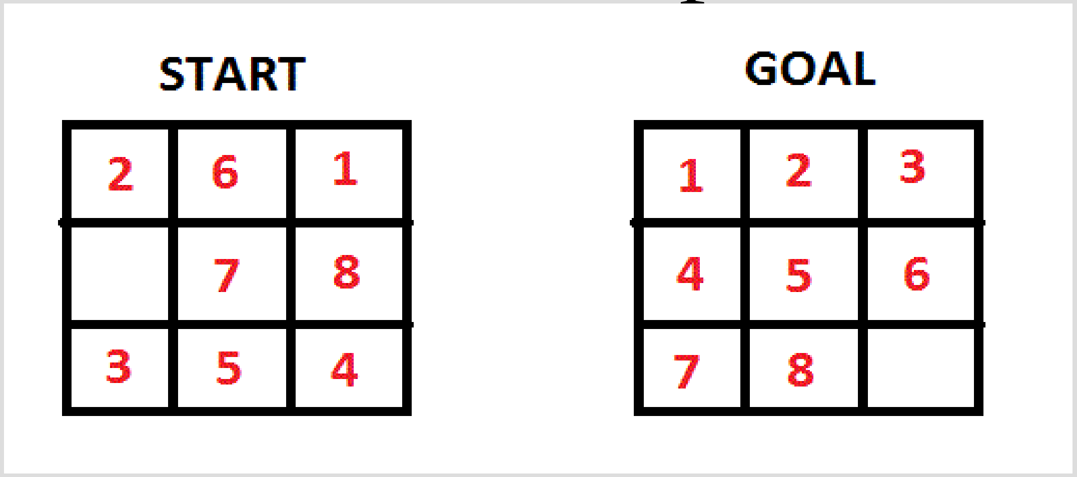
The straight line cost works for distances in the physical world, do any others exist?

One way to make heuristics is to <u>relax</u> the problem (i.e. **simplify** in a useful way)

The optimal path cost in the relaxed problem can be a heuristic for the original problem (i.e. if we were not constrained to driving on roads, we could take the straight line path)

Relaxation

Let us look at 8-puzzle heuristics:



The rules of the game are:

- You can swap any square with the blank

Relaxed rules:

1. Teleport any square to any destination
2. Move any square 1 space (overlapping ok)

Relaxation

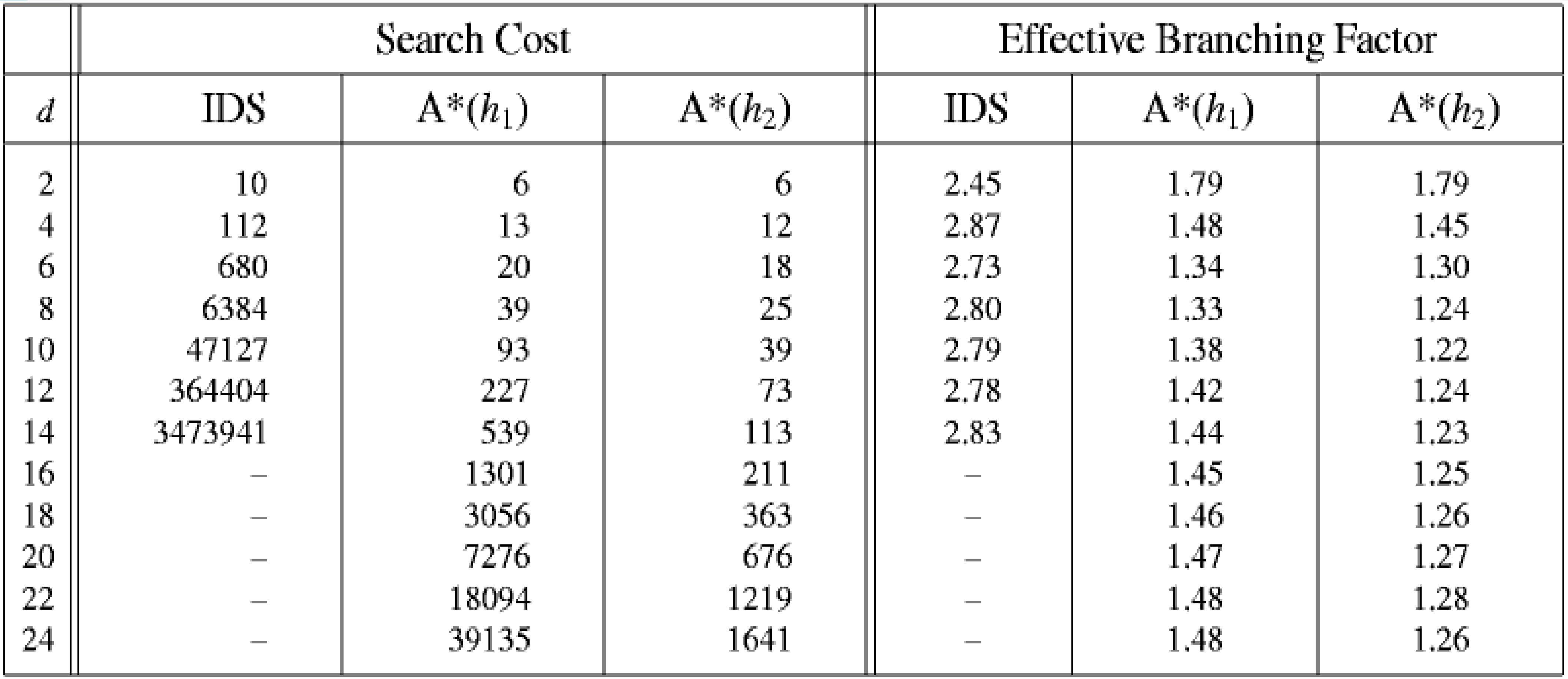
1. Teleport any square to any destination
Optimal path cost is the number of mismatched squares (blank included)

2. Move any square 1 space (overlapping ok)
Optimal path cost is Manhattan distance for each square to goal summed up

Which ones is better? (Note: these optimal solutions in relaxed need to be computed fast)

Heuristics & Branching Factor

h1 = mismatch count
h2 = number to goal difference sum



	Search Cost			Effective Branching Factor		
d	IDS	A*(h_1)	A*(h_2)	IDS	A*(h_1)	A*(h_2)
2	10	6	6	2.45	1.79	1.79
4	112	13	12	2.87	1.48	1.45
6	680	20	18	2.73	1.34	1.30
8	6384	39	25	2.80	1.33	1.24
10	47127	93	39	2.79	1.38	1.22
12	364404	227	73	2.78	1.42	1.24
14	3473941	539	113	2.83	1.44	1.23
16	–	1301	211	–	1.45	1.25
18	–	3056	363	–	1.46	1.26
20	–	7276	676	–	1.47	1.27
22	–	18094	1219	–	1.48	1.28
24	–	39135	1641	–	1.48	1.26

Heuristics & Branching Factor

The real branching factor in the 8-puzzle:

- 2 if in a corner
- 3 if on a side
- 4 if in the center
- (Thus larger “8-puzzles” tend to 4)

An <u>effective branching factor</u> finds the “average” branching factor of a tree (smaller branching = less searching)

Heuristics & Branching Factor

The effective branching factor is defined as:

$$N = b^* + (b^*)^2 + (b^*)^3 + ... + (b^*)^d$$

... where:

N = the number of nodes (i.e. size of fringe + size of explored)

b^* = effective branching factor (to find)

d = depth of solution

No easy formula, but can approximate:

$$N^{1/(d+1)} \leq b^* \leq N^{1/d}$$

Heuristics & Branching Factor

A* search then has the following properties:

If b* is the effective branching factor:

1. Completeness: Complete
2. Optimality: Optimal (if tree & admissible or graph & consistent)
3. Time complexity: $(b^*)^d$ (Note: $b^* < b$)
4. Space compexity: $(b^*)^d$

Combining Heuristics

h2 has a better branching factor than h1, and this is not a coincidence...

h2(node) $\underline{>}$ h1(node) for all nodes, thus we say h2 <u>dominates</u> h1 (and will thus perform better)

If there are multiple non-dominating heuristics: h1, h2... Then h* = max(h1, h2, ...) will dominate h1, h2, ... and will also be admissible /consistent if h1, h2 ... are as well

Combining Heuristics

If larger is better, why do we not just set h(node) = 9001?

Combining Heuristics

If larger is better, why do we not just set h(node) = 9001?

This would (probably) not be admissible...

If h(node) = 0, then you are doing the uninformed uniform cost search

If h(node) = optimal_cost(node to goal) then will ONLY explore nodes on an optimal path

Combining Heuristics

You cannot add two heuristics (h* = h1 + h2), unless there is no overlap (i.e. h1 cost is independent of h2 cost)

For example, in the 8-puzzles:

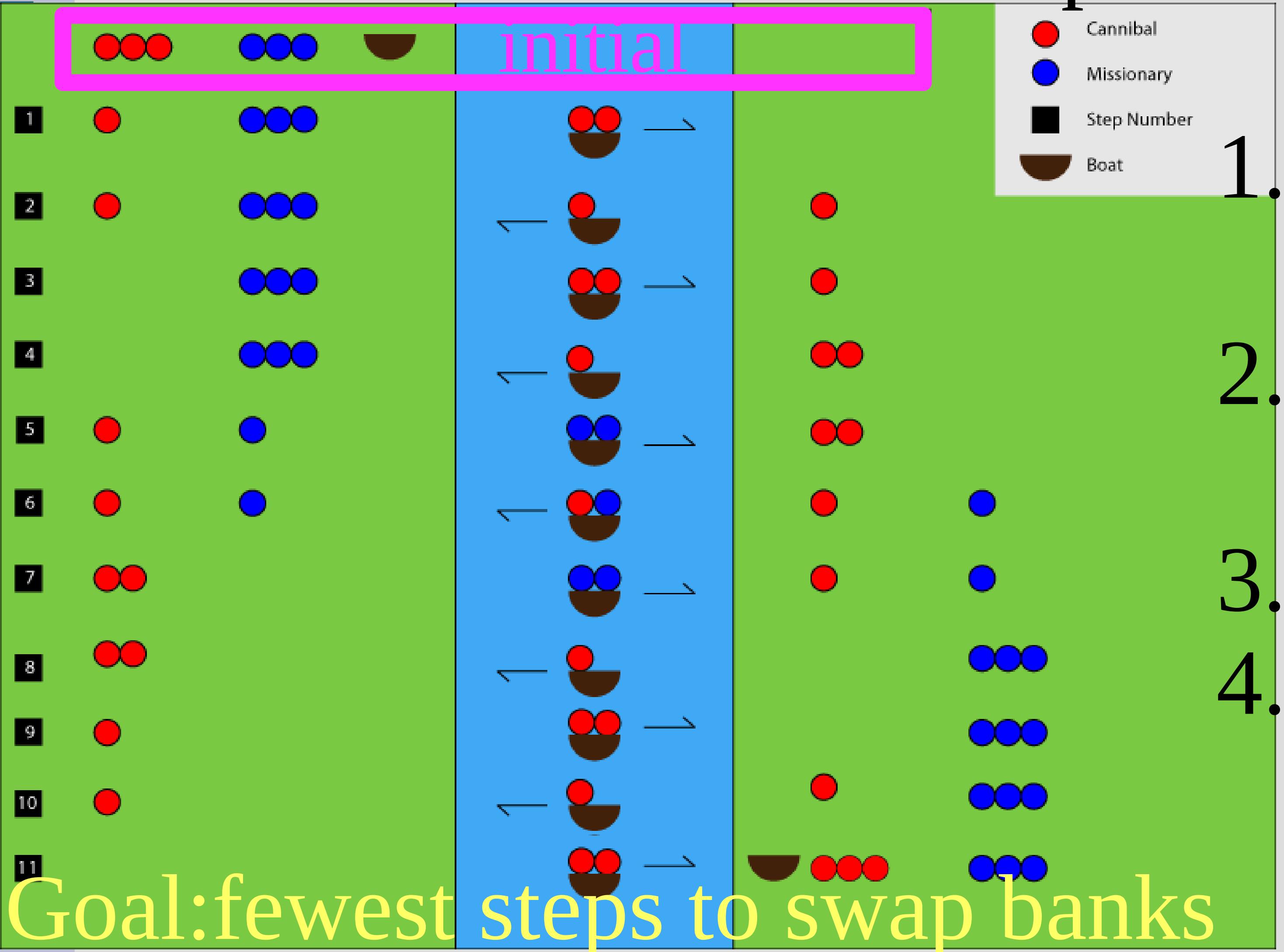
- h3: number of 1, 2, 3, 4 that are misplaced
- h4: number of 5, 6, 7, 8 that are misplaced

There is no overlap, and in fact:

- h3 + h4 = h1 (as defined earlier)

Heuristics Exercise

Cannibals & missionaries problem:



Goal:fewest steps to swap banks

Rules:

1. Either bank: m>=c, if m>0
2. 2 ppl in boat max
3. Start: 3m & 3c
4. Need 1 in boat to move

Heuristics Exercise

What relaxation did you use? (sample)

Make a heuristic for this problem

Is the heuristic admissible/consistent?

Heuristics Exercise

What relaxation did you use? (sample)

- Remove needing person in boat to move and the different types of people

Make a heuristic for this problem

- h1 = $2 * \lceil \frac{\text{[number people on left]}}{2} \rceil - 1(+1 \text{ if boat on right})$
- as you can move 2 people across in 2 steps

Is the heuristic admissible/consistent?

- YES! The point of relaxing guarantees admissibility!

Heuristics Exercise 2

UPS needs to send packages from a depot to houses using a fixed number of trucks

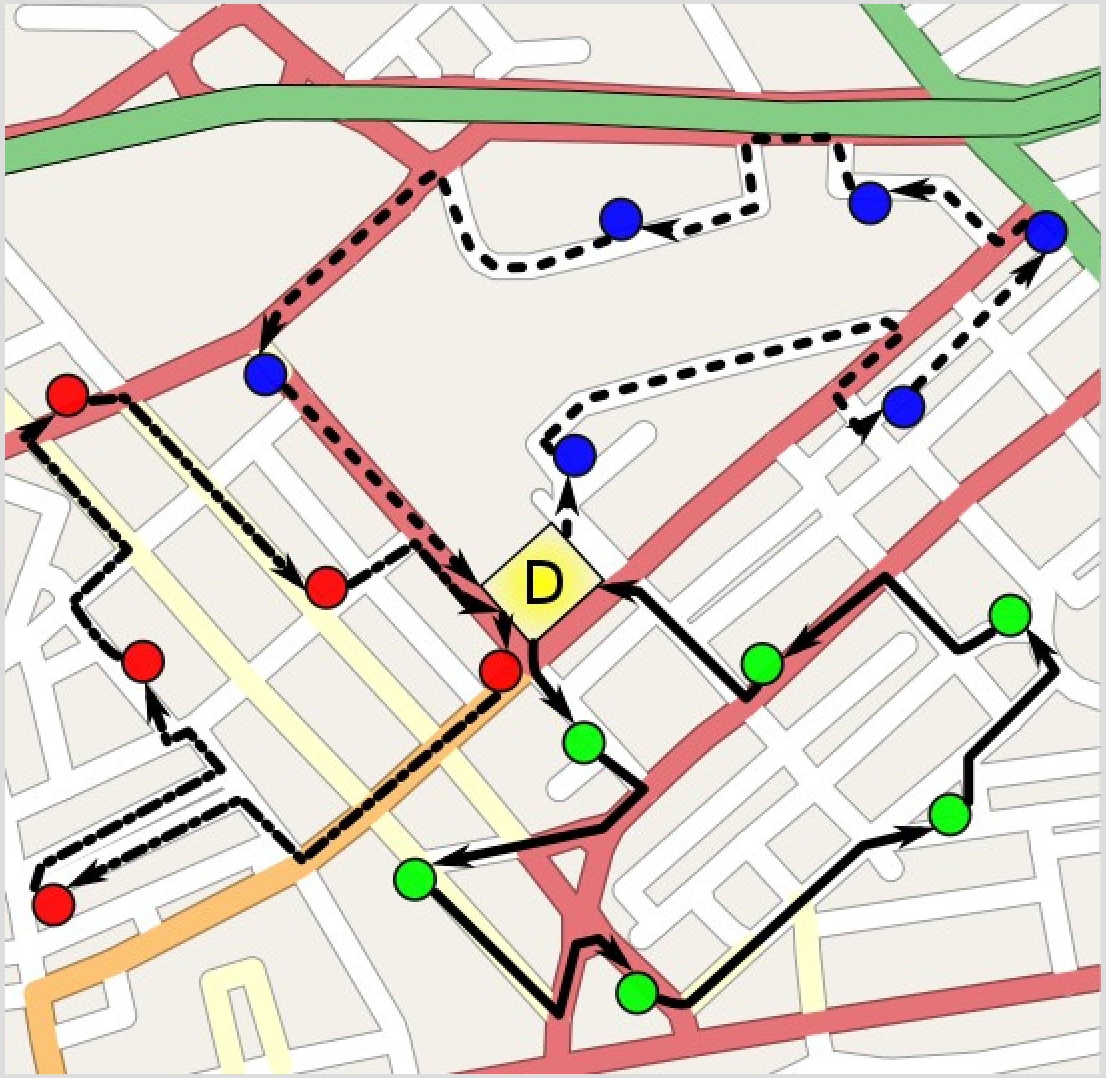
The trucks need to choose which houses and in which order they are going to visit. After delivering to all the houses, the trucks must return to the depot

The goal is to minimize the distance traveled by all the trucks

Heuristics Exercise 2

This is a sample answer

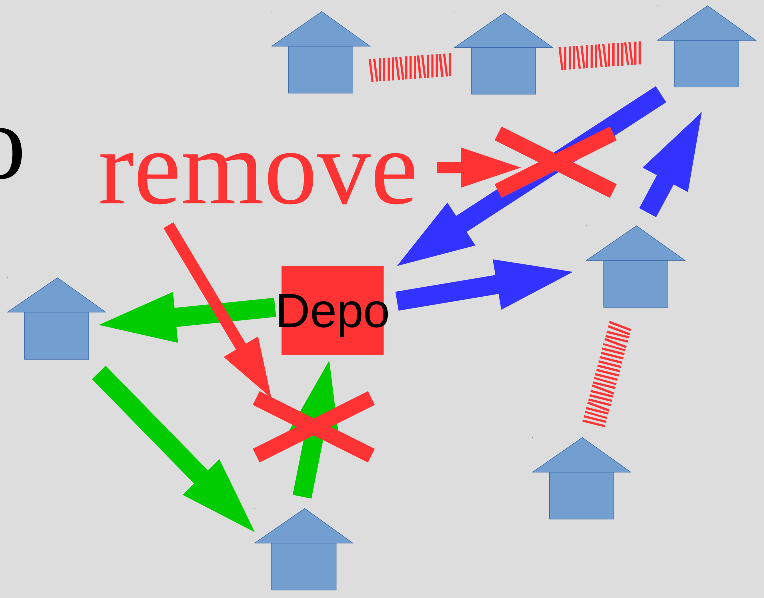
The problem is not the turning directions but rather the job distribution



Heuristics Exercise 2

Relax two rules:

1. Trucks don't need to go back to depot at end
2. Trucks can teleport to any place they have already been



This lets you build a minimum spanning tree from your current graph and let this be the total estimated cost(after removing return edge)