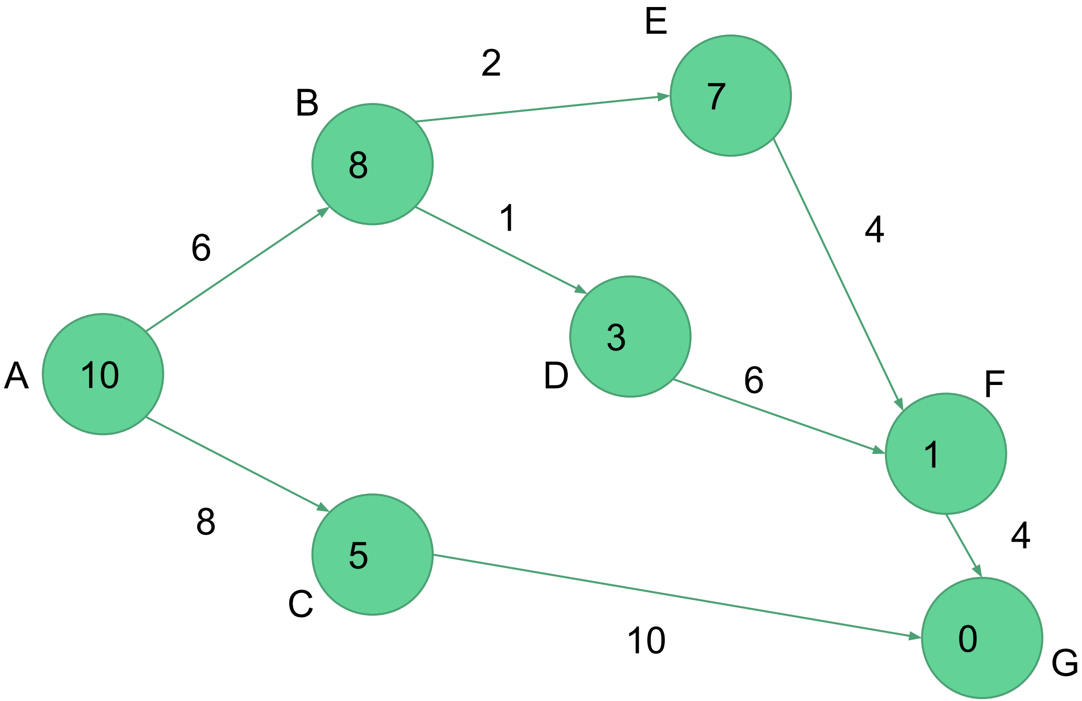
Heuristics

Tree Search becomes Graph Search

1. Add the initial state (root) to the <fringe>/**OPEN**
2. Choose a node (curr) to examine from the <fringe>/**OPEN**
   (if there is nothing in <fringe> - FAILURE)
3. **If curr is in CLOSED, go to step 2**
4. Is curr a goal state?
   If so, SOLUTION
   If not, continue
5. Expand curr by applying all possible actions
   (add the new resulting states to the <fringe>/**OPEN**)
6. Add curr to **CLOSED**
7. Go to step 2

A* Tree Search



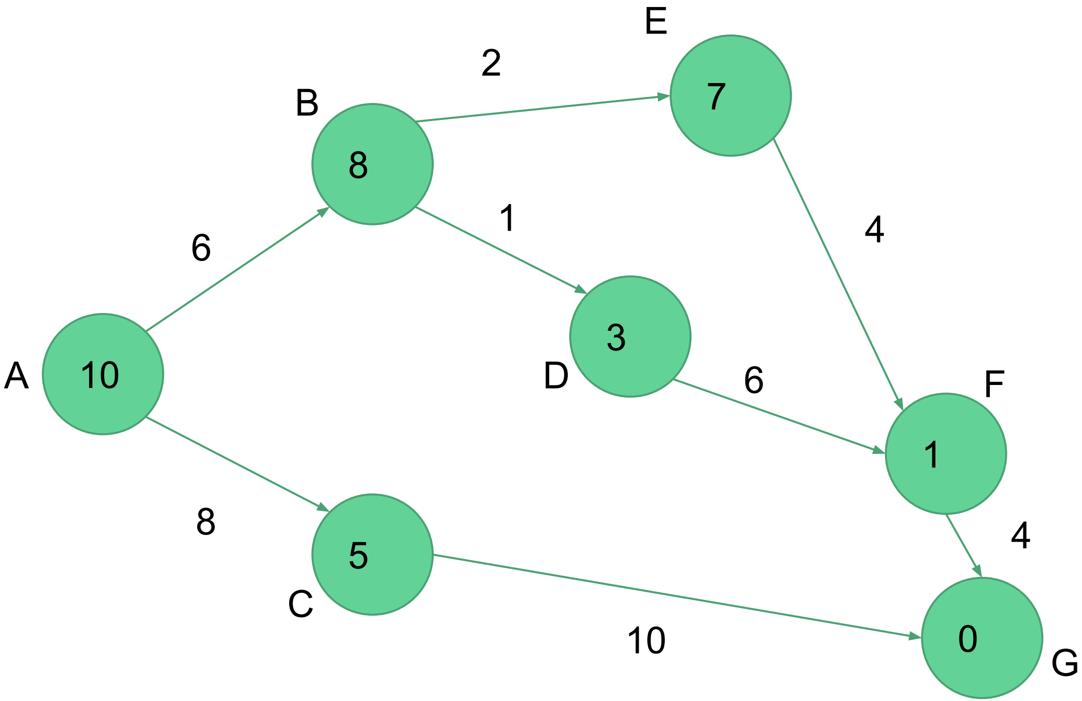
Use A* Tree Search to find the shortest path from A to G

Remember

- Evaluation function
- $f(n) = g(n) + h(n)$

Carefully keep track of your priority queue

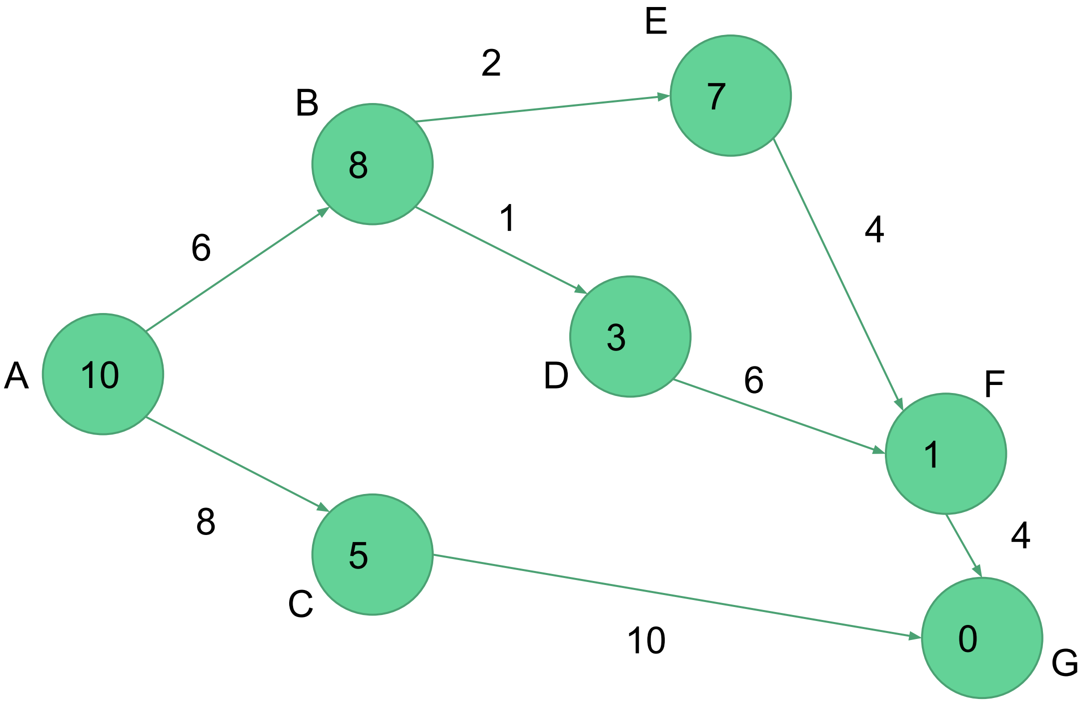
A* Tree Search



Shortest path?

A - B - D - F - G

A* Tree Search

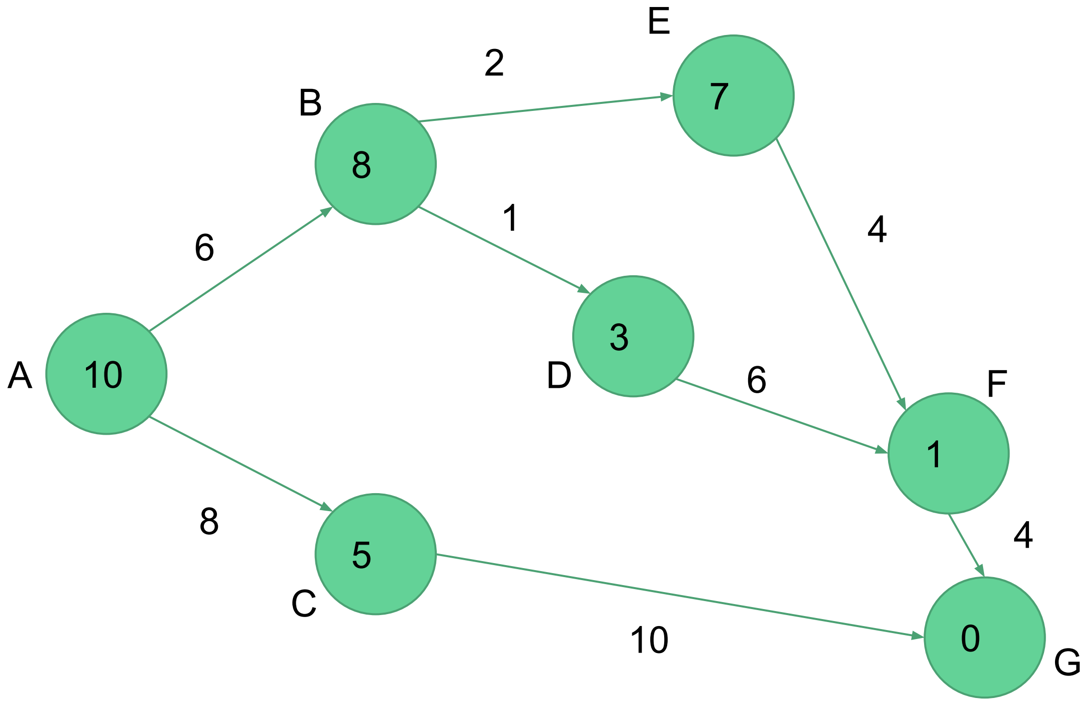


Shortest path?

~~A - B - D - F - G~~

A - B - E - F - G

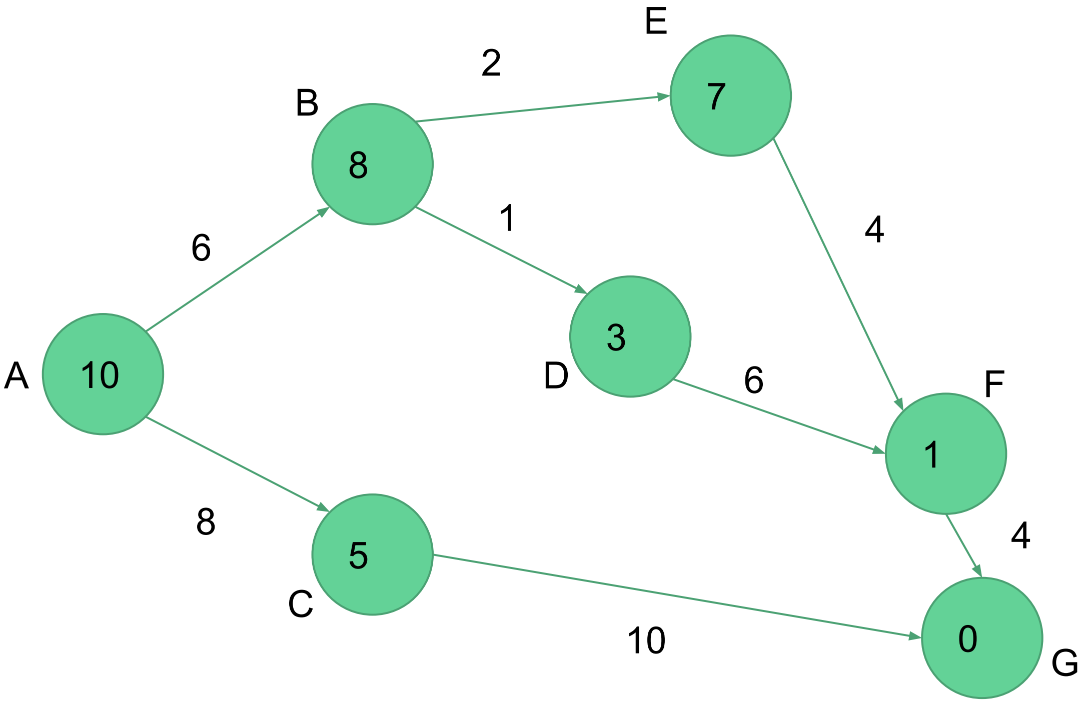
A* Tree Search



Priority Queue

Node	f(n)
A	10

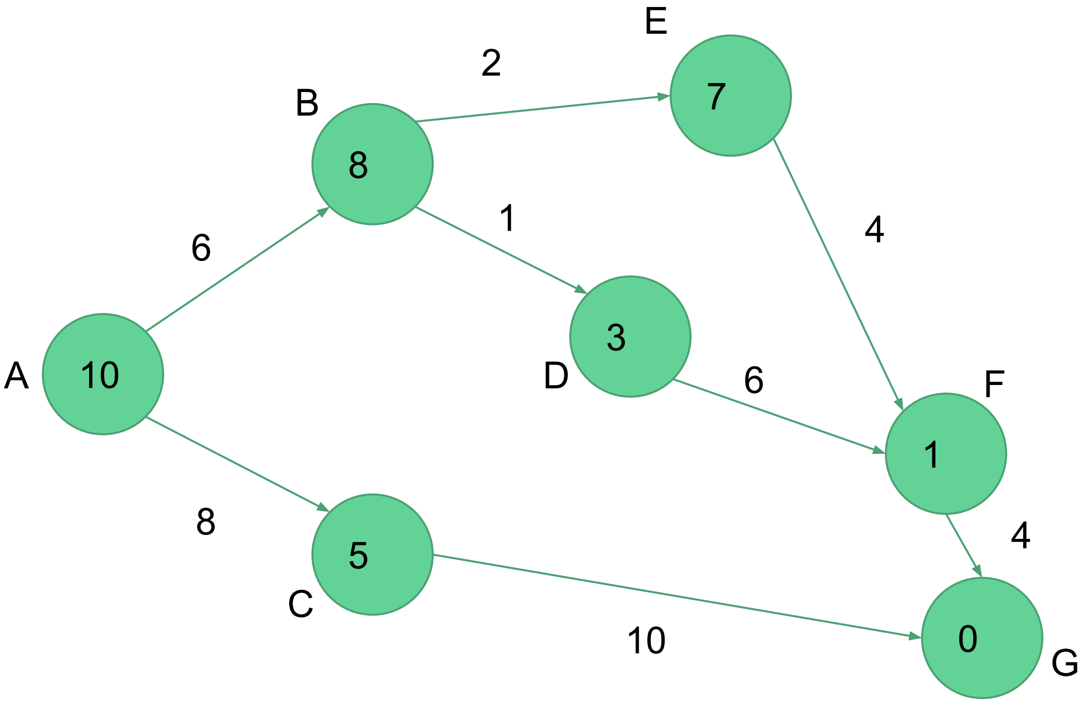
A* Tree Search



Priority Queue

Node	f(n)
C	13
B	14

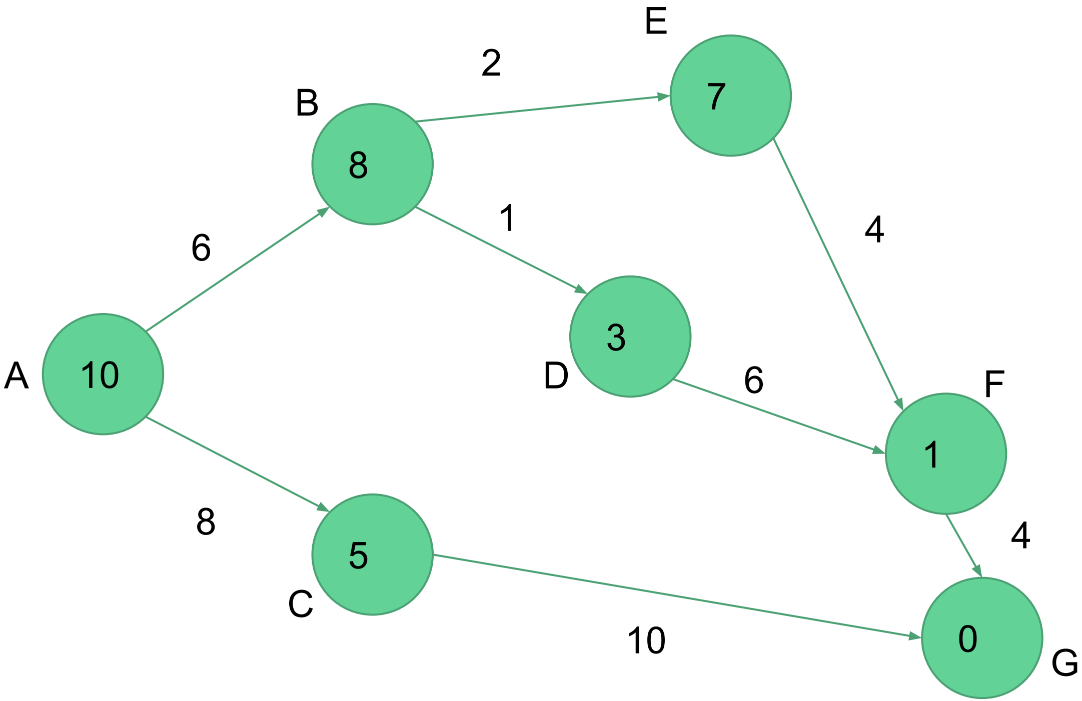
A* Tree Search



Priority Queue

Node	f(n)
B	14
G	18

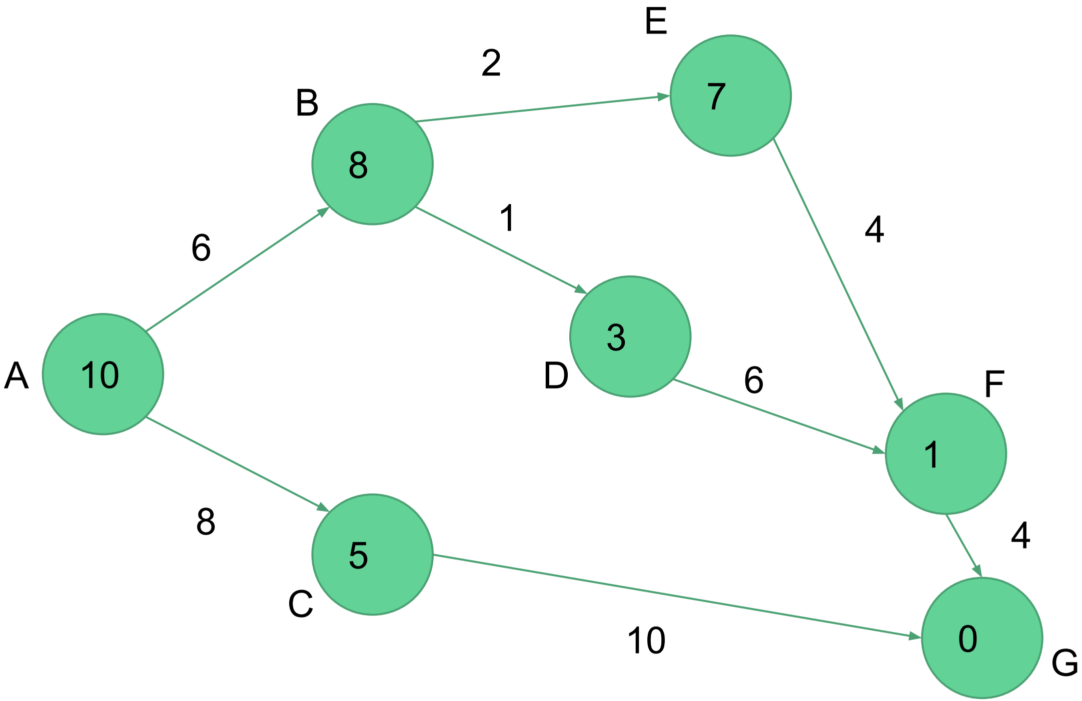
A* Tree Search



Priority Queue

Node	f(n)
D	10
E	15
G	18

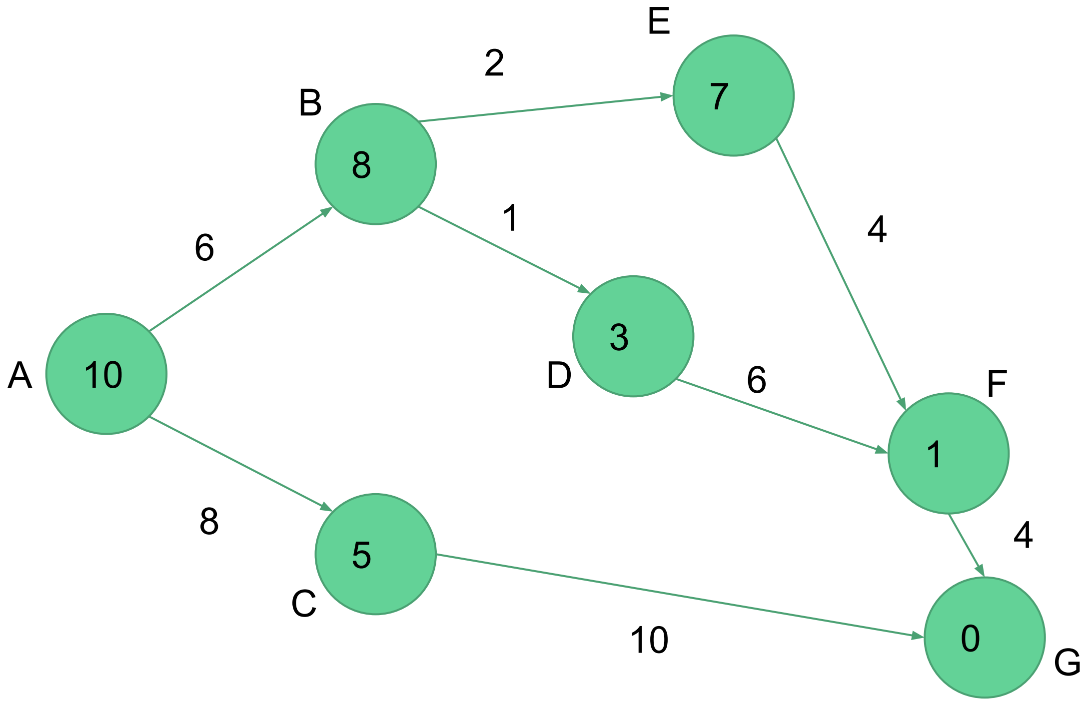
A* Tree Search



Priority Queue

Node	f(n)
F	14
E	15
G	18

A* Tree Search

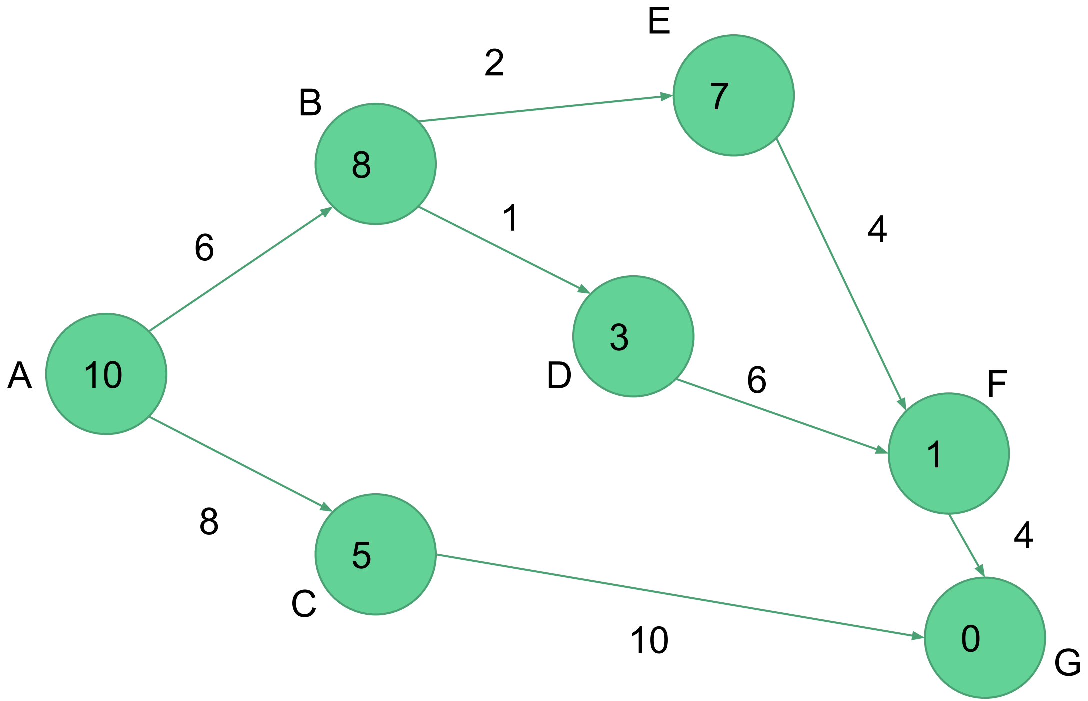


Priority Queue

Node	f(n)
E	15
G	17

A* Tree Search

tree search : abefg
graph search : abdfg

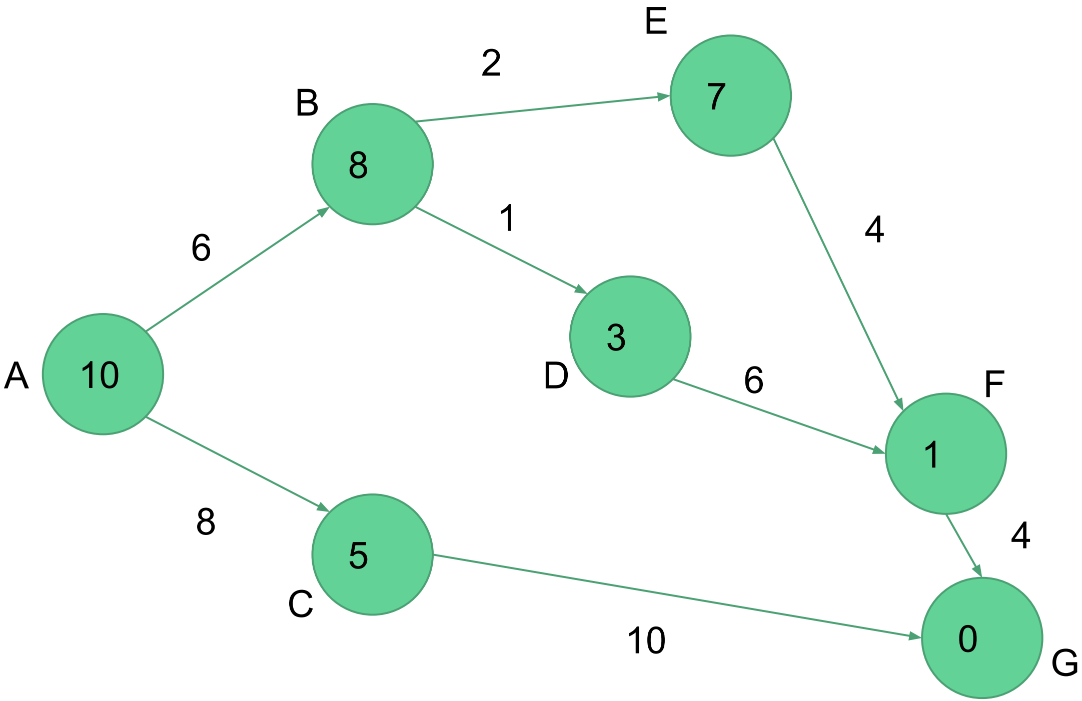


Priority Queue

Node	f(n)
F	13
G	17

But we have already expanded F

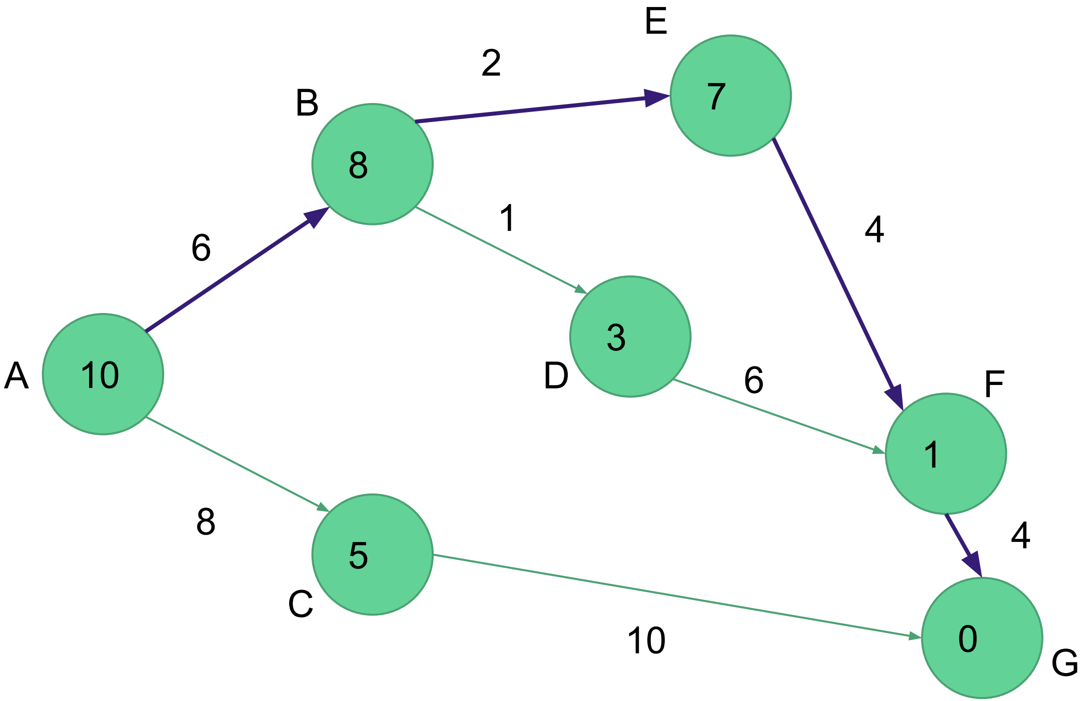
A* Tree Search



Priority Queue

Node	f(n)
G	16

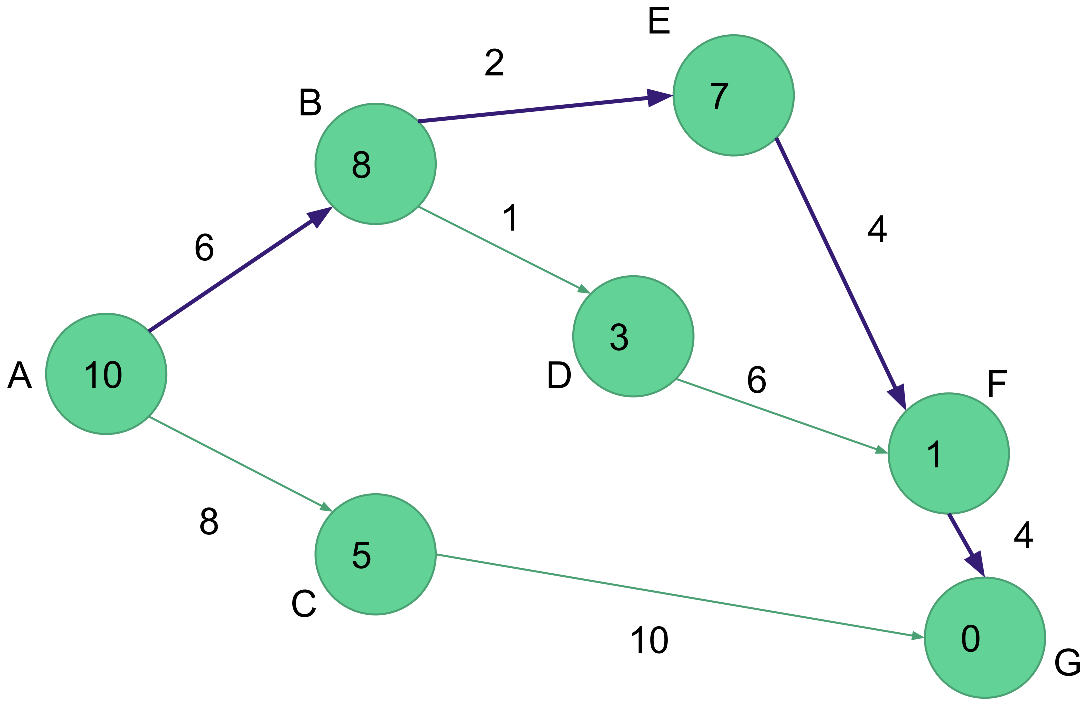
A* Tree Search



Priority Queue

Node f(n)

A* Tree Search



Is the heuristic h(n) admissible?

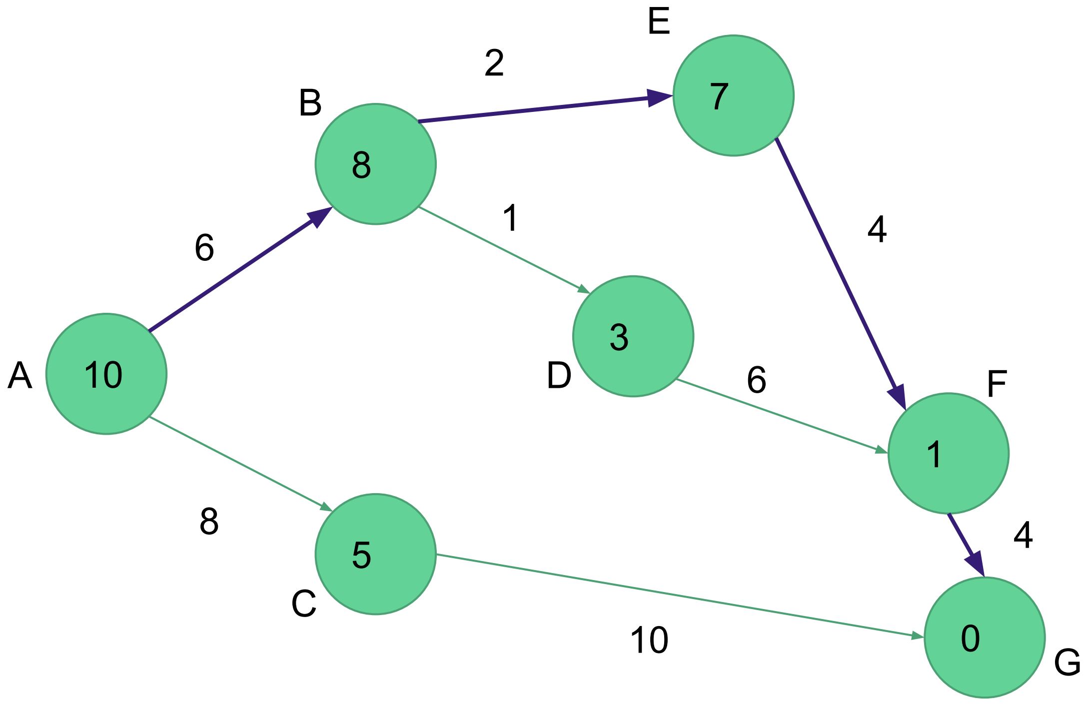
A heuristic *h(n)* is **admissible** if for every node *n*,

$h(n) \leq h^*(n)$, where $h^*(n)$ is the **true** cost to reach the goal state from *n*.

Never overestimates

aka always underestimates

A* Tree Search

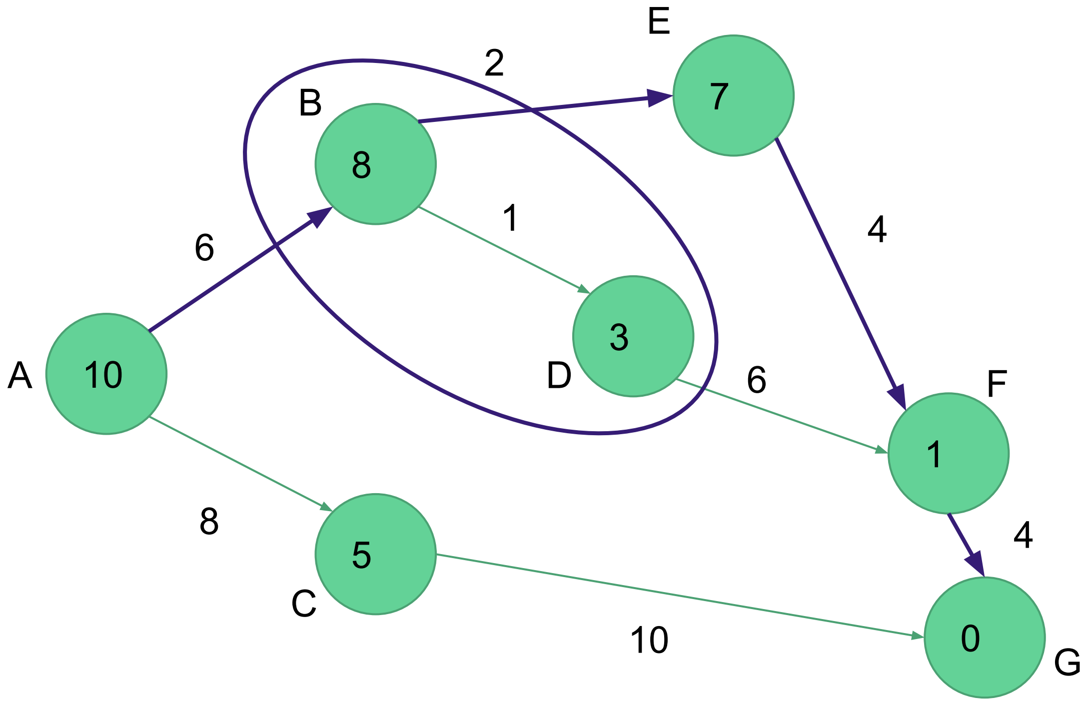


Is the heuristic h(n) consistent?

A heuristic *h(n)* is **consistent**, or monotone, if for every node *n* and each successor *p*,

> $h(n) \leq c(n,p) + h(p)$, where $c(n,p)$ is the cost to reach p from n.

A* Tree Search



Is the heuristic h(n) consistent?

A heuristic *h(n)* is **consistent**, or monotone, if for every node *n* and each successor *p*,

> $h(n) \leq c(n,p) + h(p)$, where $c(n,p)$ is the cost to reach *p* from *n*.

Admissible and Consistent Heuristics

A heuristic $h(n)$ is **admissible** if for every node n,

$h(n) \leq h^*(n)$, where $h^*(n)$ is the **true** cost to reach the goal state from n.

A heuristic $h(n)$ is **consistent**, monotone, if for every node n and each successor p,

$h(n) \leq c(n,p) + h(p)$, where $c(n,p)$ is the cost to reach p from n.

Most heuristics are admissible and consistent

Heuristics in A*

If h(n) is admissible, A* is guaranteed to produce an optimal solution

If h(n) is consistent, A* Tree Search and A* Graph Search both work

If h(n) is inconsistent, only A* Tree Search will produce an optimal solution

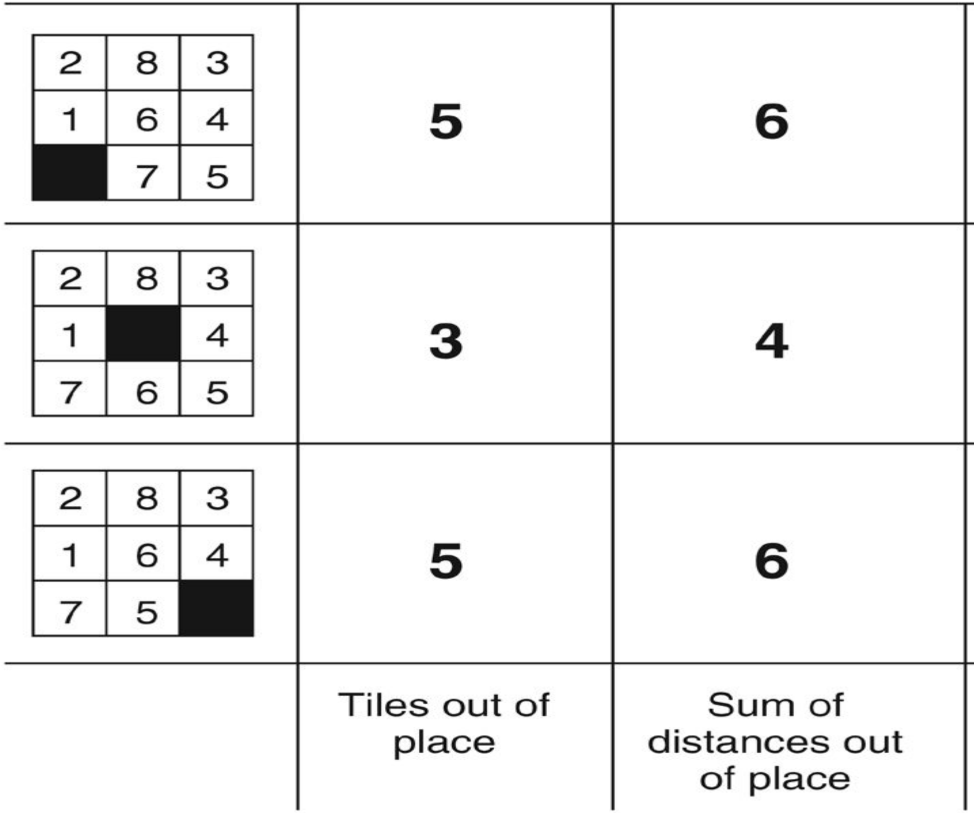
Heuristics for 8-puzzle

$h_1(n)$ = Tiles out of place

Hamming distance

$h_2(n)$ = Sum of distances out of place

Manhattan distance



Dominance

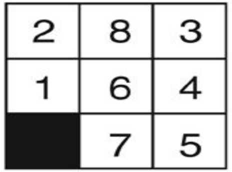
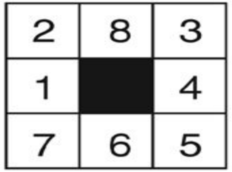
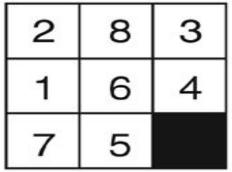
$h_1(n)$ = Tiles out of place

Hamming distance

$h_2(n)$ = Sum of distances out of place

Manhattan distance

Which heuristic is better?

	Tiles out of place	Sum of distances out of place
	**5**	**6**
	**3**	**4**
	**5**	**6**

Dominance

$h_2(n) \geq h_1(n)$

Both heuristics are admissible

h_2 **dominates** h_1

h_2 is better for search

2 8 3 1 6 4 7 5	**5**	**6**
2 8 3 1 4 7 6 5	**3**	**4**
2 8 3 1 6 4 7 5	**5**	**6**
	Tiles out of place	Sum of distances out of place

Let’s Relax

Problem relaxation

A problem with fewer restrictions on the actions is called a **relaxed problem**

The cost of an optimal solution to a relaxed problem is an admissible heuristic for the original problem

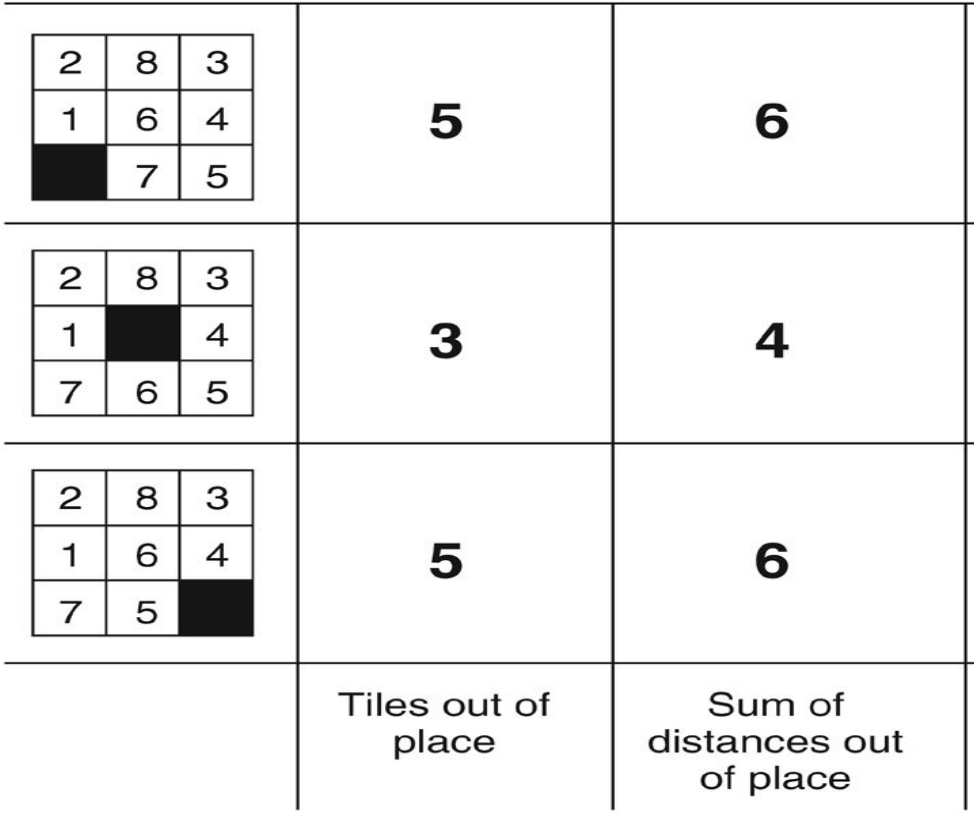
If the rules of the 8-puzzle are relaxed so that a tile can move **anywhere**, then $h_1(n)$ gives the shortest solution

If the rules are relaxed so that a tile can move to **any adjacent square**, then $h_2(n)$ gives the shortest solution

Relaxation in 8-puzzle

If the rules of the 8-puzzle are relaxed so that a tile can move **anywhere**, then $h_1(n)$ gives the shortest solution

If the rules are relaxed so that a tile can move to **any adjacent square**, then $h_2(n)$ gives the shortest solution



Relaxation in Towers of Hanoi

Only top disk can be moved

A larger disk cannot be placed on a smaller disk

Relaxation 1: Any disk can be moved to any position on any peg

Relaxation 2: Every disk (except largest) can move only to neighboring peg
(assumes goal peg is farthest one)

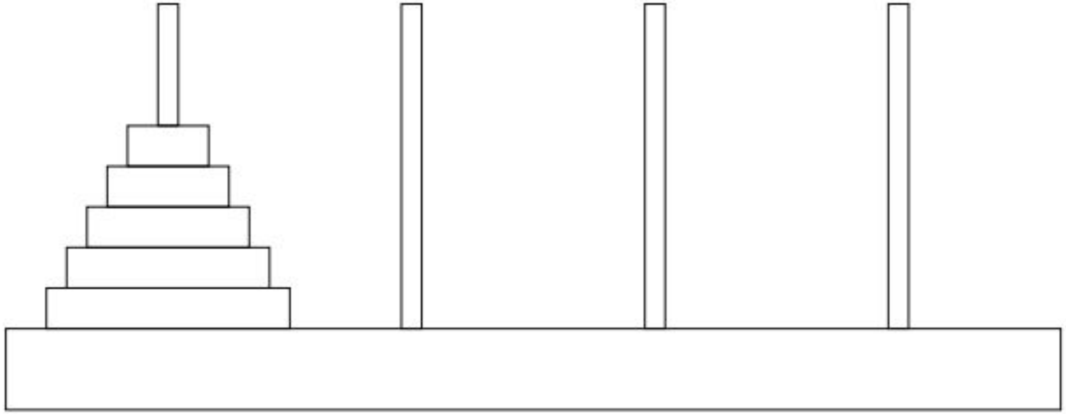


Figure 6: Five-disk four-peg Towers of Hanoi problem

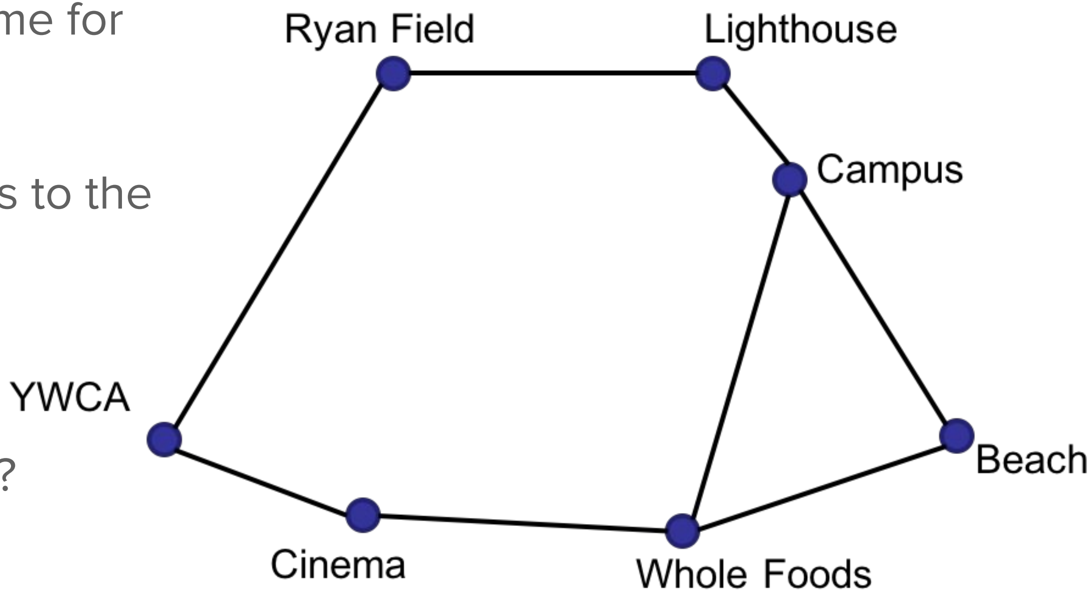
Heuristics in pathfinding

It's Friday night and a new movie came out. You want to get there in time for the show.

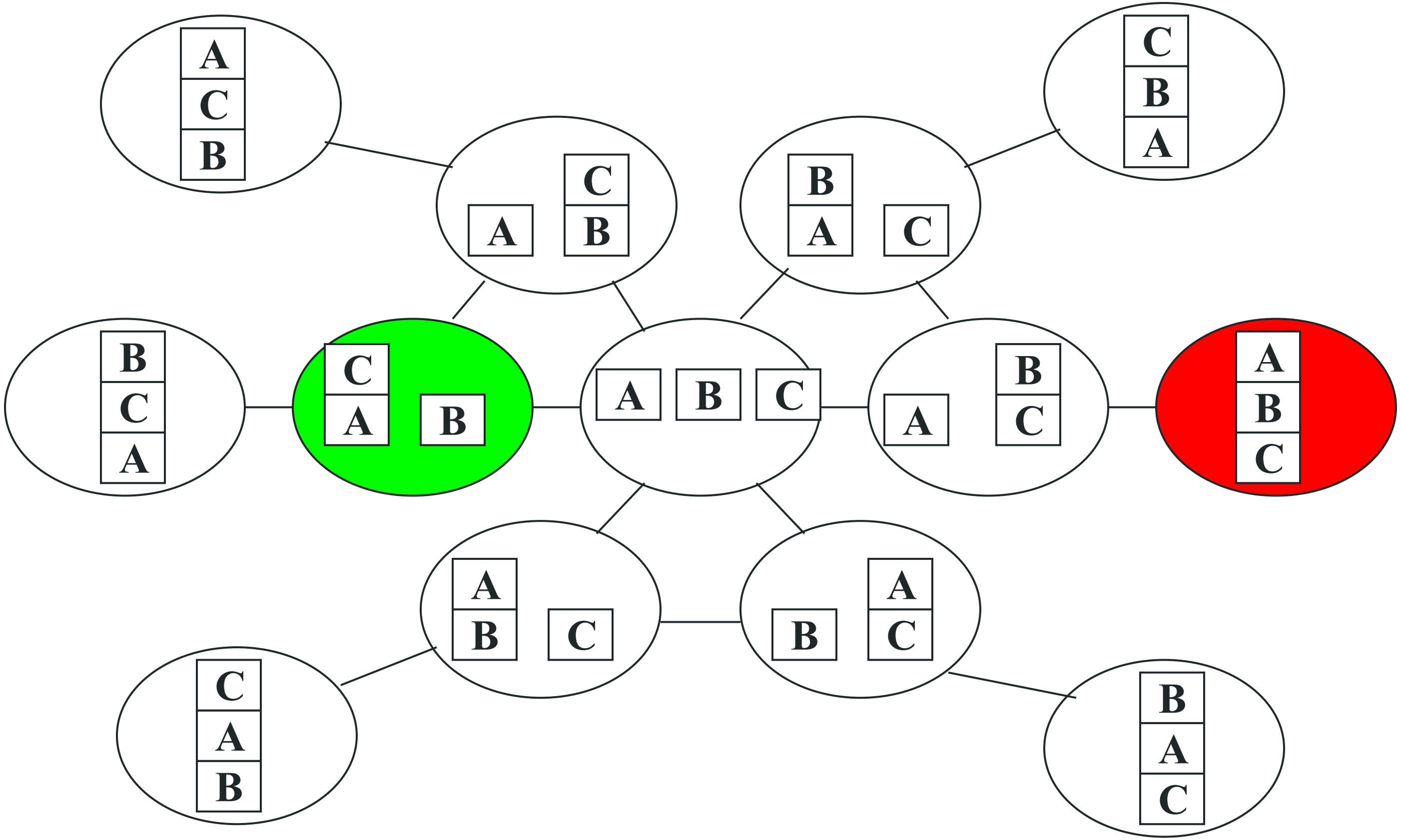
How to quickly get from campus to the cinema?

What heuristic do you use?

Is it admissible? Is it consistent?



Blocks World



Subproblems

Admissible heuristics can also be derived from solution to subproblems

Pattern database stores heuristics

	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

Initial State

	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

Goal

			3
			7
			11
12	13	14	15

Subproblem

Felner, A., Zahavi, U., Holte, R., Schaeffer, J., Sturtevant, N., & Zhang, Z. (2011). Inconsistent heuristics in theory and practice. *Artificial Intelligence*, *175*(9-10), 1570-1603.

What if 2 agents searching for opposing solutions?

Next week: Adversarial search

Chapter

Friday: Tree search code