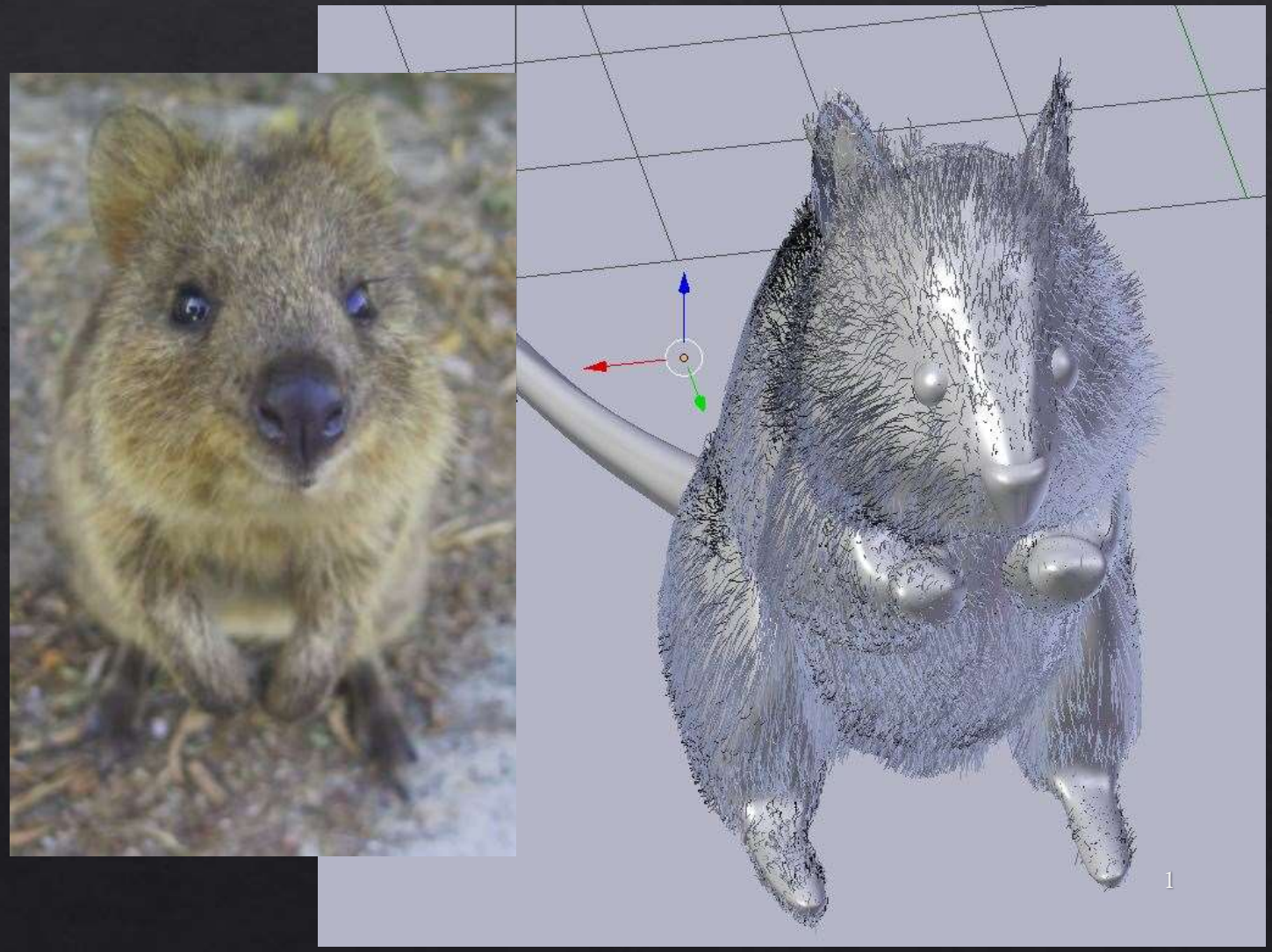
COMP3308/3608

Weeks 2-3 Practice Questions

…and quokkas! :)



By Jess McBroom

Before You Begin

Hello! :)

These slides contain practice questions with solutions to help you revise the Intro to AI content from weeks 2-3. They also contain quokkas!

Please note that these are informal questions (I've been pretty comprehensive, but the questions don't cover everything from the lectures/tutorials so it's important to still look through those).

The questions are divided into 3 parts:

1. Search Algorithm Practice
2. General Knowledge
3. Think!

If you have any questions about anything in these slides, feel free to ask them on Piazza!

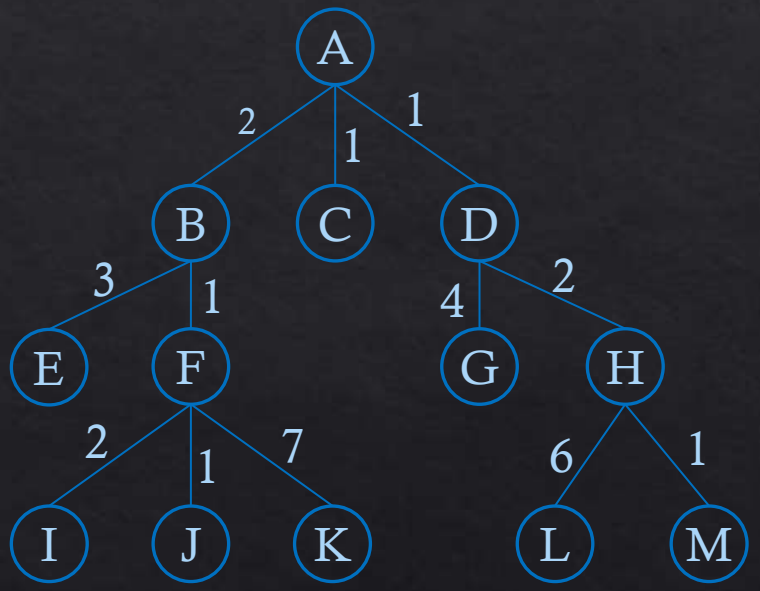
Enjoy!

Part 1 – Search Algorithm Practice

Question 1

For the following tree, write down the order in which nodes are expanded using:

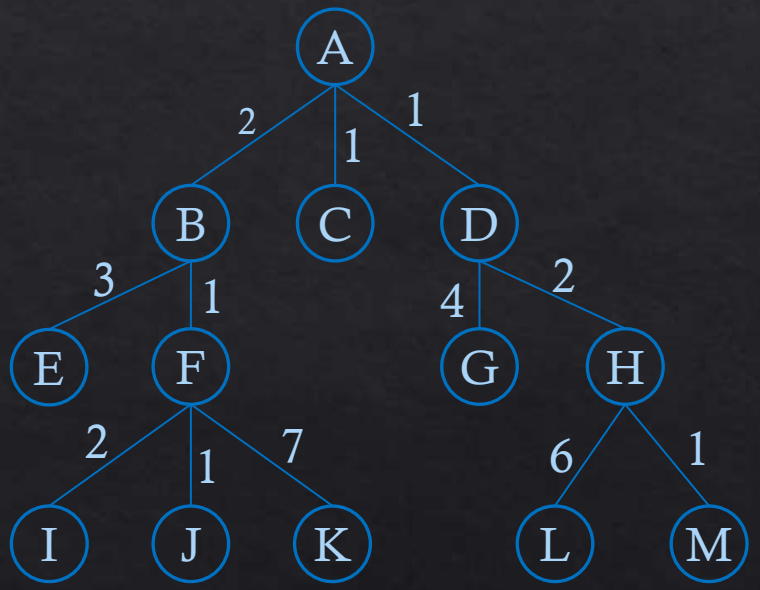
BFS



Question 1

For the following tree, write down the order in which nodes are expanded using:

BFS



Short Answer:

A B C D E F G H I J K L M

Worked Solution:

We expand “layer” by layer, from left to right, until we reach a goal node. There are no goal nodes here, so we expand everything:

Layer 1: A

Layer 2: B C D

Layer 3: E F G H

Layer 4: I J K L M

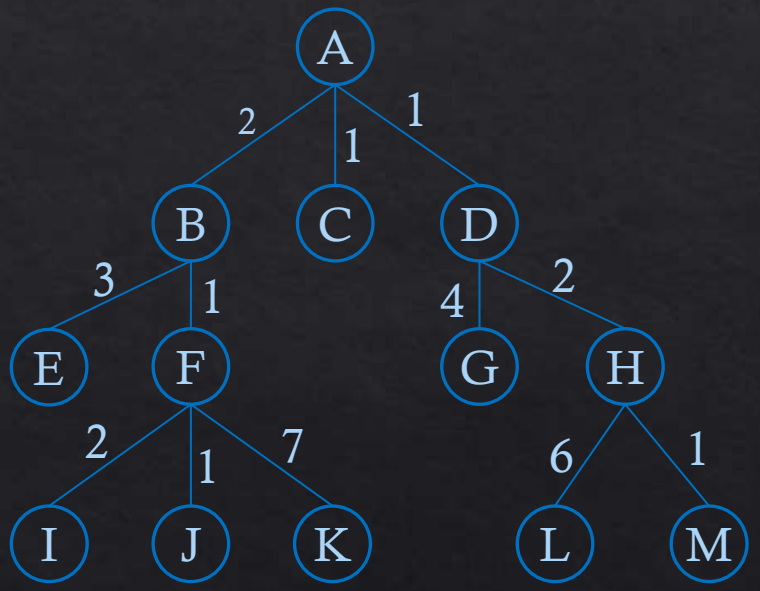
So, all together we have:

A B C D E F G H I J K L M

Question 2

For the following tree, write down the order in which nodes are expanded using:

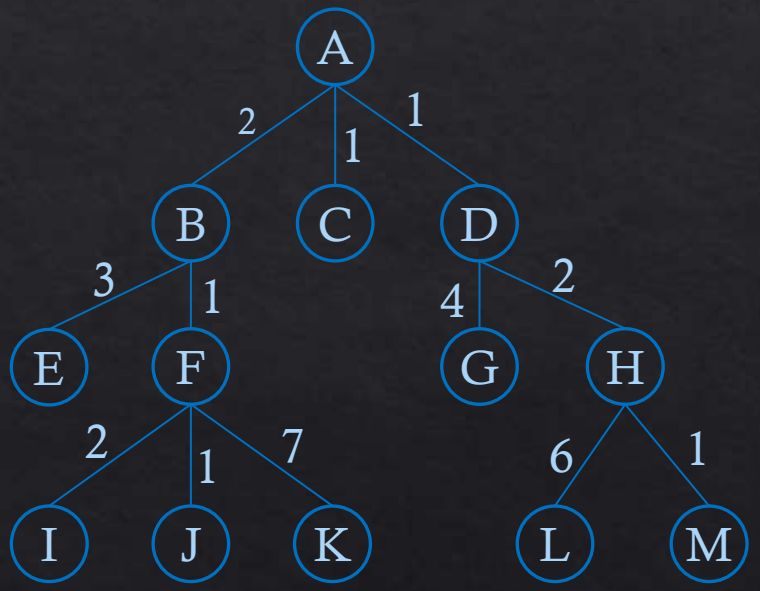
DFS



Question 2

For the following tree, write down the order in which nodes are expanded using:

DFS



Short Answer:

A B E F I J K C D G H L M

Worked Solution:

We start at the root, and keep expanding the leftmost child until we can't expand anymore. Then we backtrack! There are no goal nodes here, so we expand everything. So, we have:

A → B → E, backtrack to B

F → I, backtrack to F

J, backtrack to F

K, backtrack to A

C, backtrack to A

D → G, backtrack to D

H → L, backtrack to H

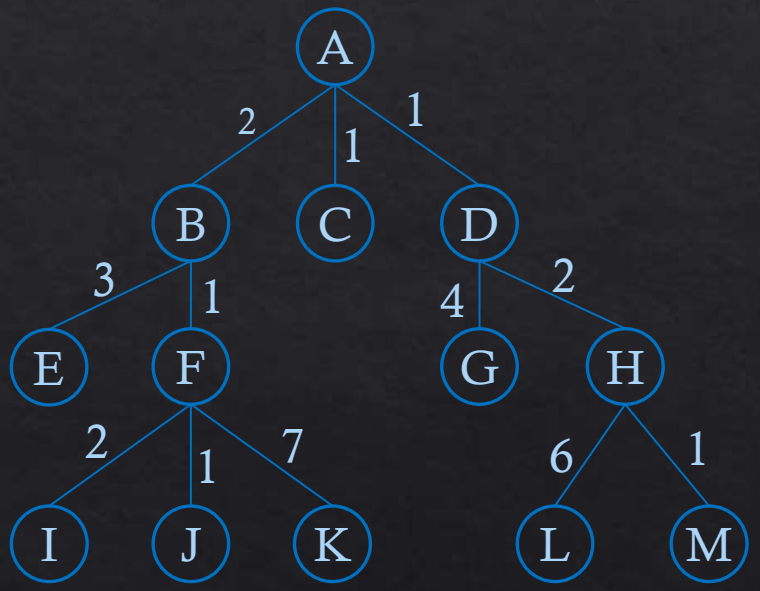
M

So, all together we have: A B E F I J K C D G H L M

Question 3

For the following tree, write down the order in which nodes are expanded using:

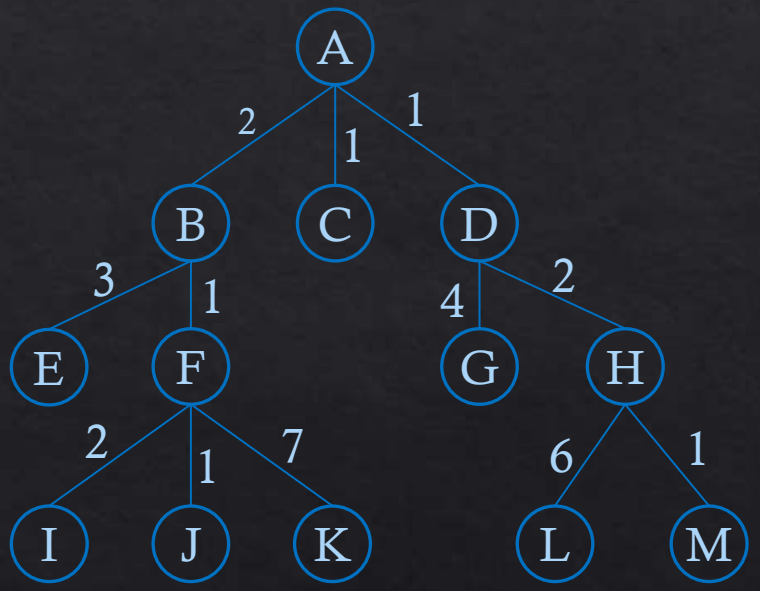
IDS



Question 3

For the following tree, write down the order in which nodes are expanded using:

IDS



Short Answer:

A A B C D A B E F C D G H A B E F I J K C D G H L M

Worked Solution:

This is just like DFS, but done multiple times! We first limit the depth to 0 (first layer). Then, if we don't find a goal node, we change the depth limit to 1 (first 2 layers) and try again. If we again don't find the goal, we keep increasing the limit by 1. Here, there is no goal node, so we have to consider every possible depth:

DFS up to depth 0: A

DFS up to depth 1: A B C D

DFS up to depth 2: A B E F C D G H

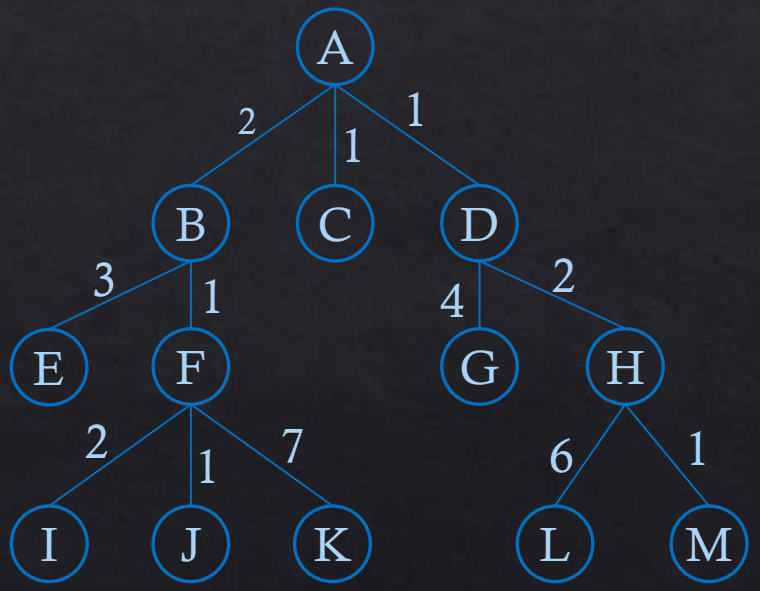
DFS up to depth 3: A B E F I J K C D G H L M

So, all together we have: A A B C D A B E F C D G H A B E F I J K C D G H L M

Question 4

For the following tree, write down the order in which nodes are expanded using:

UCS (expand the leftmost node in the case of ties)



Question 4

For the following tree, write down the order in which nodes are expanded using:

UCS (expand the leftmost node in the case of ties)

Tip: when doing this on paper, you can add up the total cost to each node from the root first (orange numbers). Then, the solution will be the nodes in order of increasing cost

Short Answer: A C D B F H J M E I G L K

Worked Solution:

We start with the root, then expand the node with the lowest cost on each step, taking the leftmost node if there are ties: (blue = expanded nodes, white= fringe)

1: | (A,0)

2: A | (C, 1) (D, 1) (B,2)

3: A C | (D,1) (B,2)

4: A C D | (B,2) (H,3) (G,5)

5: A C D B | (F,3) (H,3) (E,5) (G,5)

6: A C D B F | (H,3) (J,4) (E,5) (I,5) (G,5) (K,10)

7: A C D B F H| (J,4) (M,4) (E,5) (I,5) (G,5) (L,9) (K,10)

8: A C D B F H J| (M,4) (E,5) (I,5) (G,5) (L,9) (K,10)

9: A C D B F H J M| (E,5) (I,5) (G,5) (L,9) (K,10)

10: A C D B F H J M E | (I,5) (G,5) (L,9) (K,10)

11: A C D B F H J M E I | (G,5) (L,9) (K,10)

12: A C D B F H J M E I G | (L,9) (K,10)

13: A C D B F H J M E I G L| (K,10)

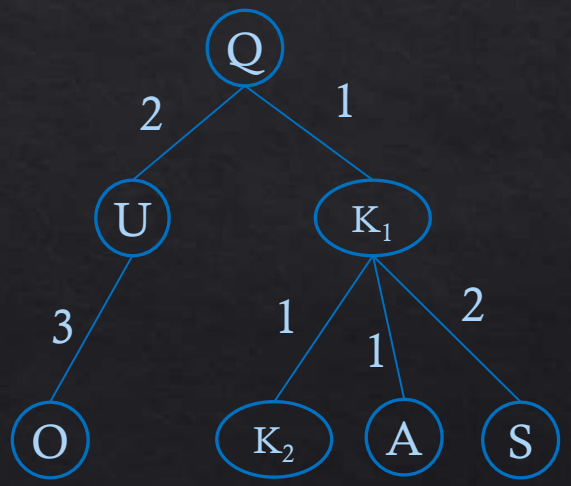
14: A C D B F H J M E I G L K

Question 5

For the following tree, write down the order in which nodes are expanded using:

BFS, DFS, IDS and UCS

(expand the leftmost node in the case of ties)

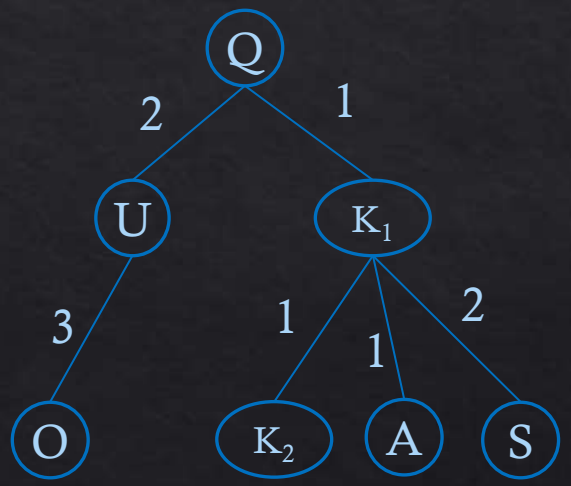


Question 5

For the following tree, write down the order in which nodes are expanded using:

BFS, DFS, IDS and UCS

(expand the leftmost node in the case of ties)



Short Answers:

BFS:

Q U K_1 O K_2 A S

DFS:

Q U O K_1 K_2 A S

IDS:

Q Q U K_1 Q U O K_1 K_2 A S

UCS:

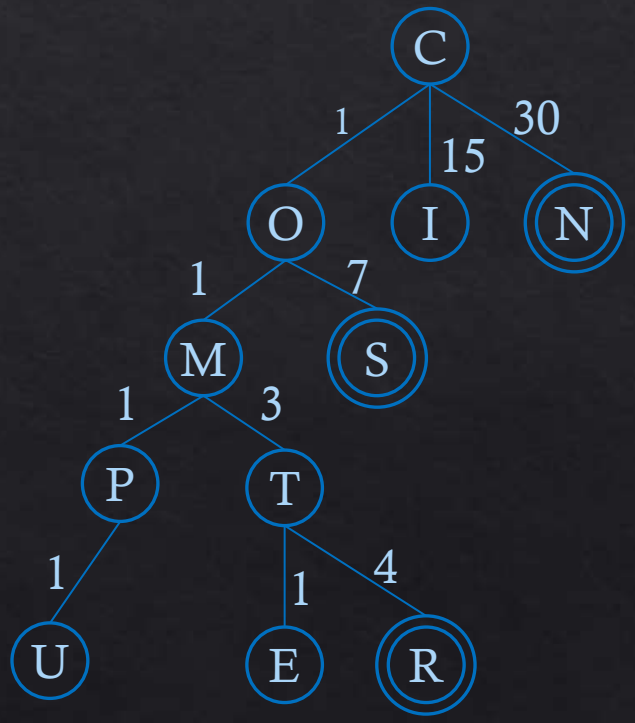
Q K_1 U K_2 A S O

Question 6

For the following tree (there are goal nodes!), write down the order in which nodes are expanded using:

BFS, DFS, IDS and UCS

(expand the leftmost node in the case of ties)

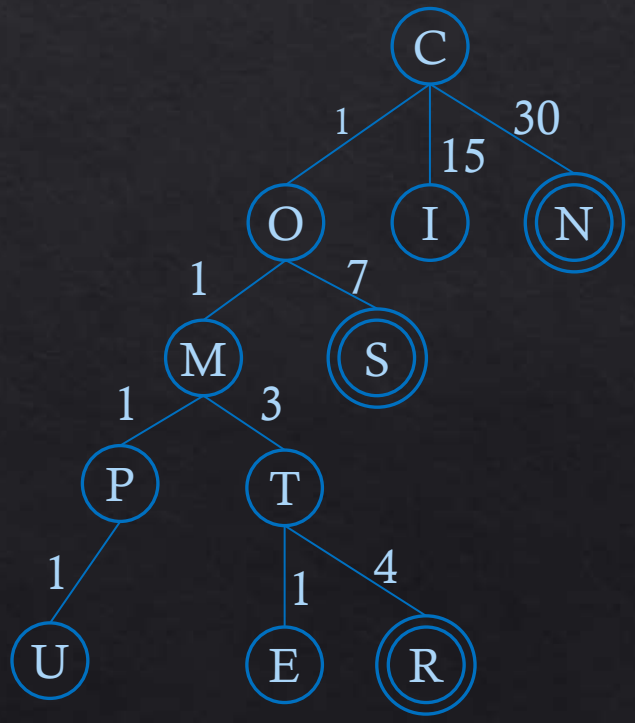


Question 6

For the following tree (there are goal nodes!), write down the order in which nodes are expanded using:

BFS, DFS, IDS and UCS

(expand the leftmost node in the case of ties)



Short Answers:

BFS:

C O I N

DFS:

C O M P U T E R

IDS:

C C O I N

UCS:

C O M P U T E S

Great Work!

Quokka Quote:

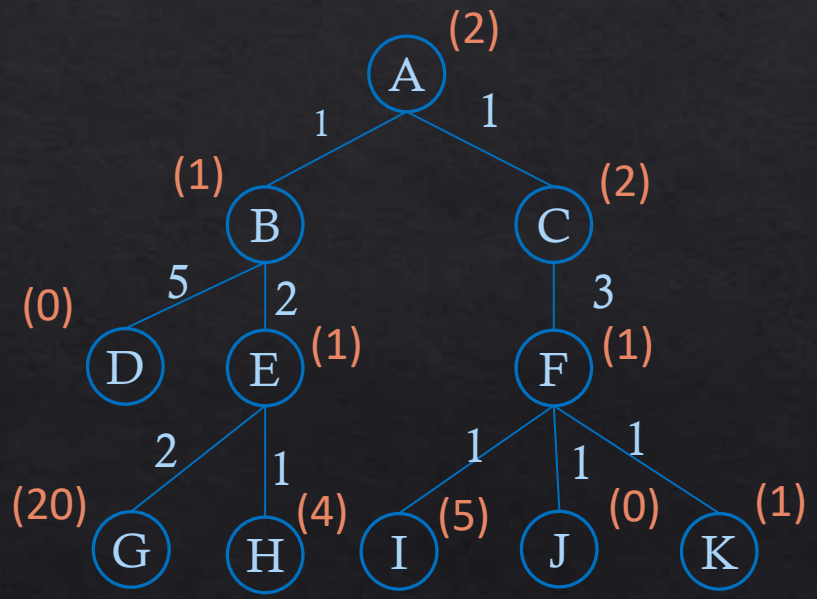
With quokkas, there’s no need to kanga-rue your AI practice! :D

Question 7

For the following tree, write down the order in which nodes are expanded using:

Greedy Search

(expand the leftmost node in the case of ties)

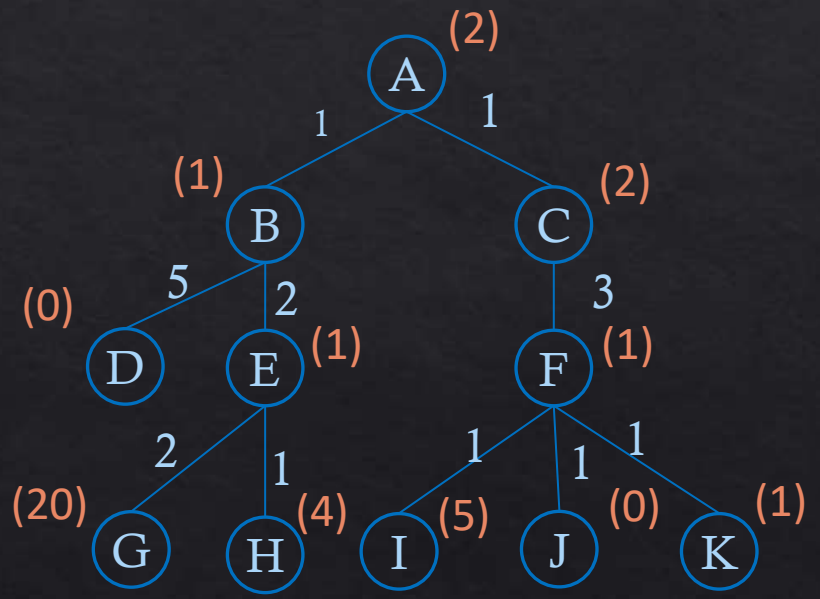


Question 7

For the following tree, write down the order in which nodes are expanded using:

Greedy Search

(expand the leftmost node in the case of ties)



Short Answer: A B D E C F J K H I G

Worked Solution:

We do the same as with UCS, except we order nodes by their heuristic values, and not their costs from the root. (blue = expanded nodes, white= fringe)

1: | (A,2)

2: A | (B,1) (C, 2)

3: A B | (D,0) (E,1) (C,2)

4: A B D | (E,1) (C,2)

5: A B D E | (C,2) (H,4) (G,20)

6: A B D E C | (F,1) (H,4) (G,20)

7: A B D E C F | (J,0) (K,1) (H,4) (I,5) (G,20)

8: A B D E C F J | (K,1) (H,4) (I,5) (G,20)

9: A B D E C F J K | (H,4) (I,5) (G,20)

10: A B D E C F J K H | (I,5) (G,20)

11: A B D E C F J K H I | (G,20)

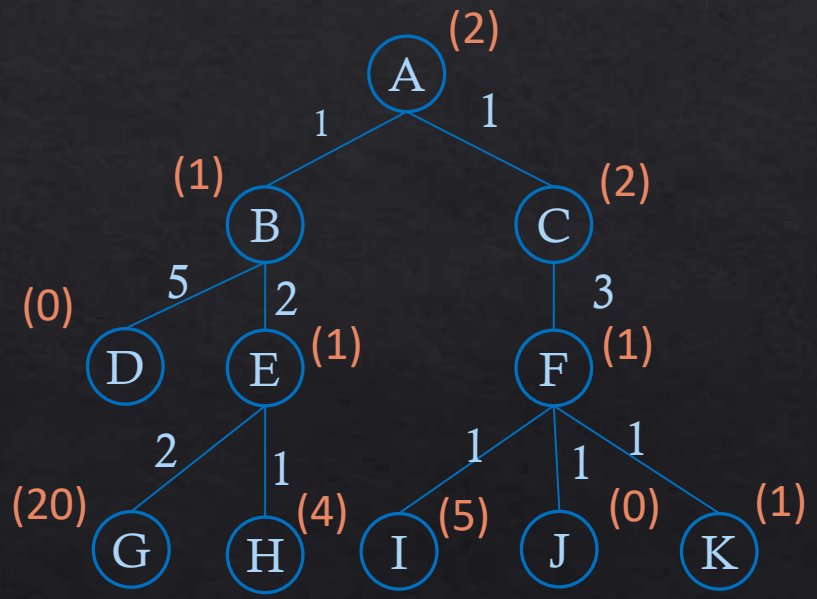
12: A B D E C F J K H I G

Question 8

For the following tree, write down the order in which nodes are expanded using:

A* Search

(expand the leftmost node in the case of ties)

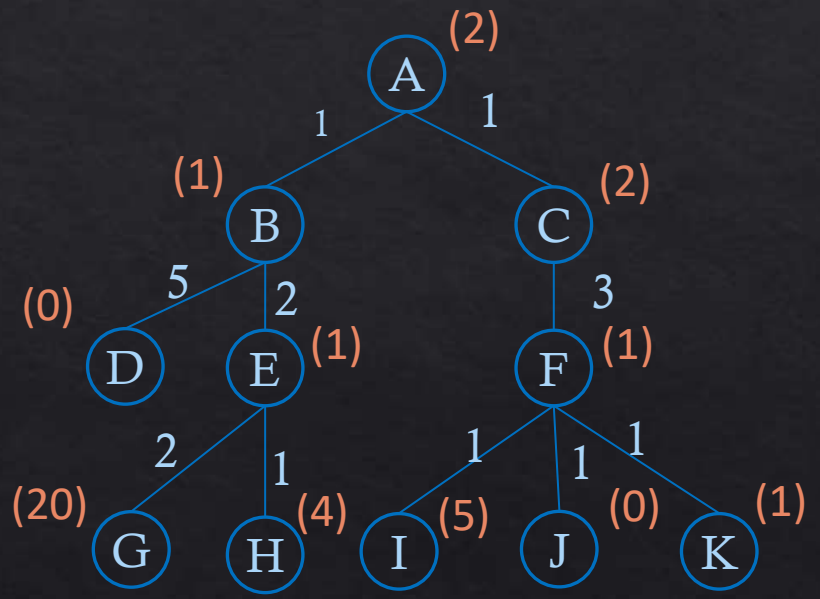


Question 8

For the following tree, write down the order in which nodes are expanded using:

A* Search

(expand the leftmost node in the case of ties)



Short Answer: A B C E F J D K H I G

Worked Solution:

We do the same as with UCS and Greedy search, except we order nodes by their f values (which is the cost from the root plus the heuristic value). (blue = expanded nodes, white= fringe)

1: | (A,2)

2: A | (B,2) (C, 3)

3: A B | (C,3) (E,4) (D,6)

4: A B C | (E,4) (F,5) (D,6)

5: A B C E | (F,5) (D,6) (H,8) (G,25)

6: A B C E F | (J,5) (D,6) (K,6) (H,8) (I,10) (G,25)

7: A B C E F J | (D,6) (K,6) (H,8) (I,10) (G,25)

8: A B C E F J D | (K,6) (H,8) (I,10) (G,25)

9: A B C E F J D K | (H,8) (I,10) (G,25)

10: A B C E F J D K H | (I,10) (G,25)

11: A B C E F J D K H I| (G,25)

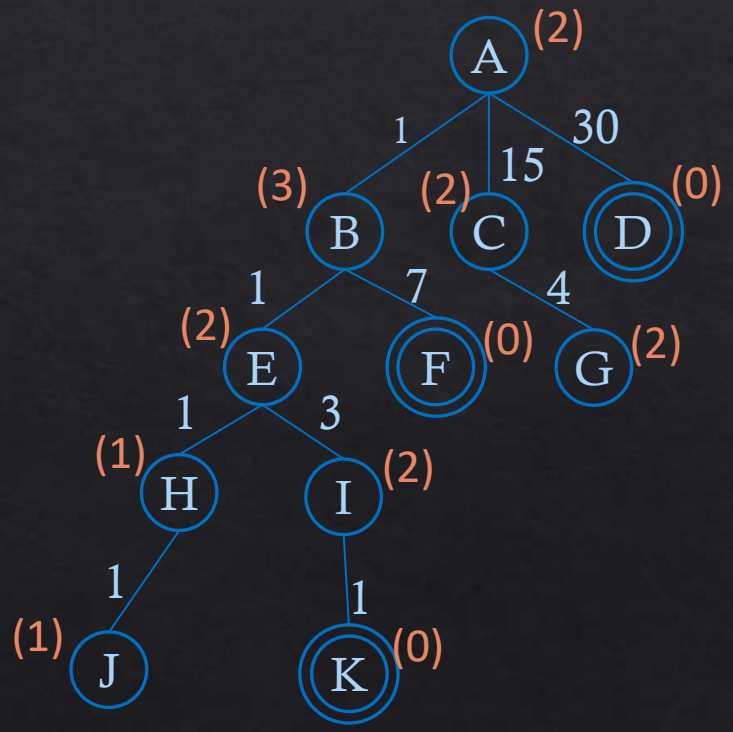
12: A B C E F J D K H I G

Question 9

For the following tree (there are goal nodes!), write down the order in which nodes are expanded using:

Greedy Search and A*

(expand the leftmost node in the case of ties)

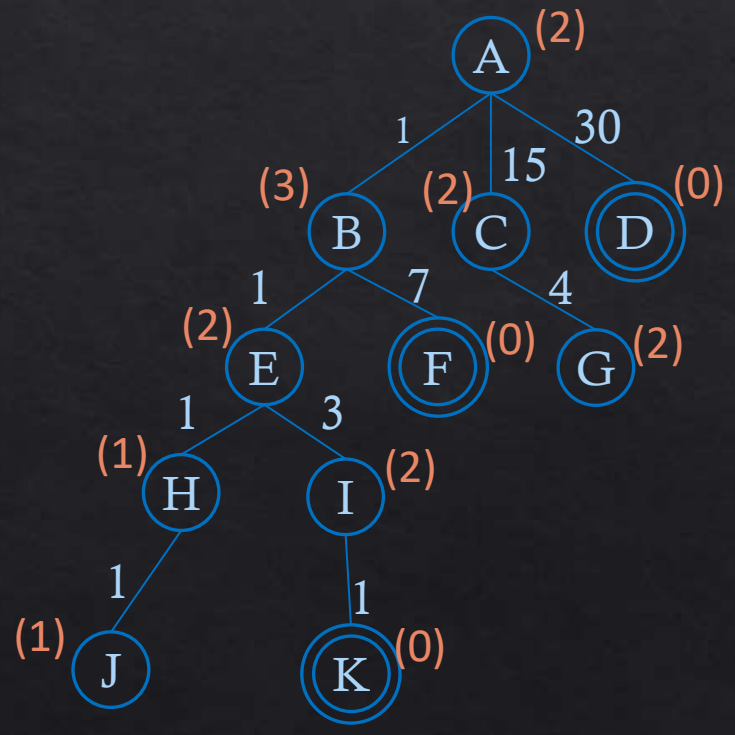


Question 9

For the following tree (there are goal nodes!), write down the order in which nodes are expanded using:

Greedy Search and A*

(expand the leftmost node in the case of ties)



Short Answers:

Greedy:

A D

A*:

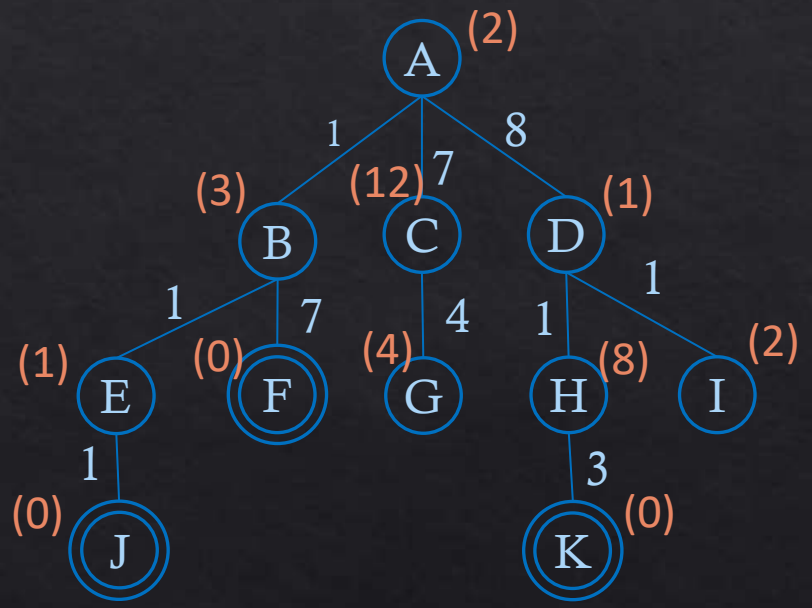
A B E H J I K

Question 10

For the following tree (there are goal nodes!), write down the order in which nodes are expanded using:

Greedy Search and A*

(expand the leftmost node in the case of ties)

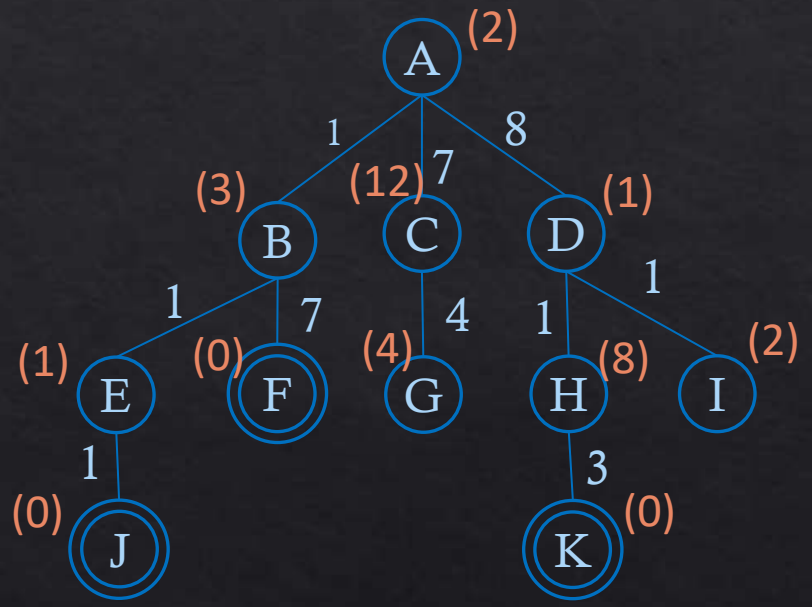


Question 10

For the following tree (there are goal nodes!), write down the order in which nodes are expanded using:

Greedy Search and A*

(expand the leftmost node in the case of ties)



Short Answers:

Greedy:

A D I B F

A*:

A B E J

Didn’t those questions just make you feel like….

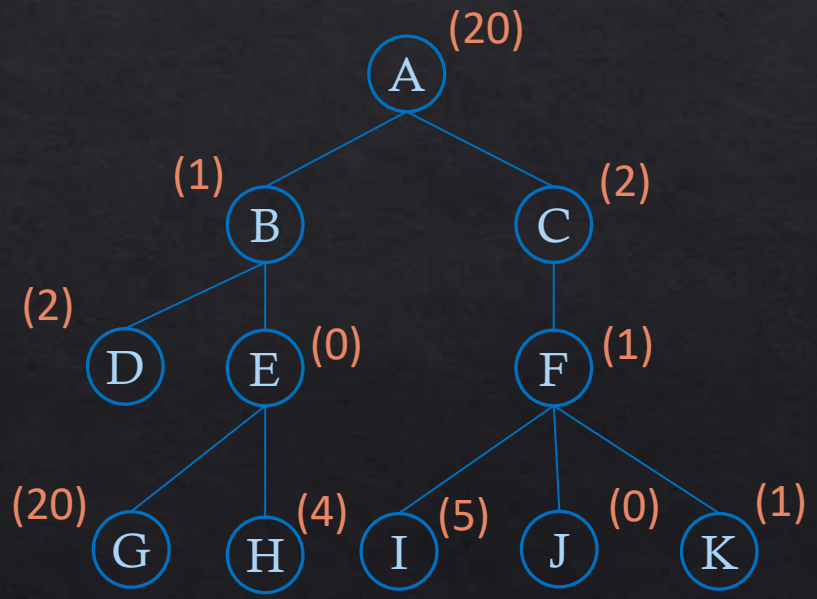
A STAR!?

Question 11

For the following tree, write down the order in which nodes are visited using:

Hill Climbing (to find a minimum)

(expand the leftmost node in the case of ties)

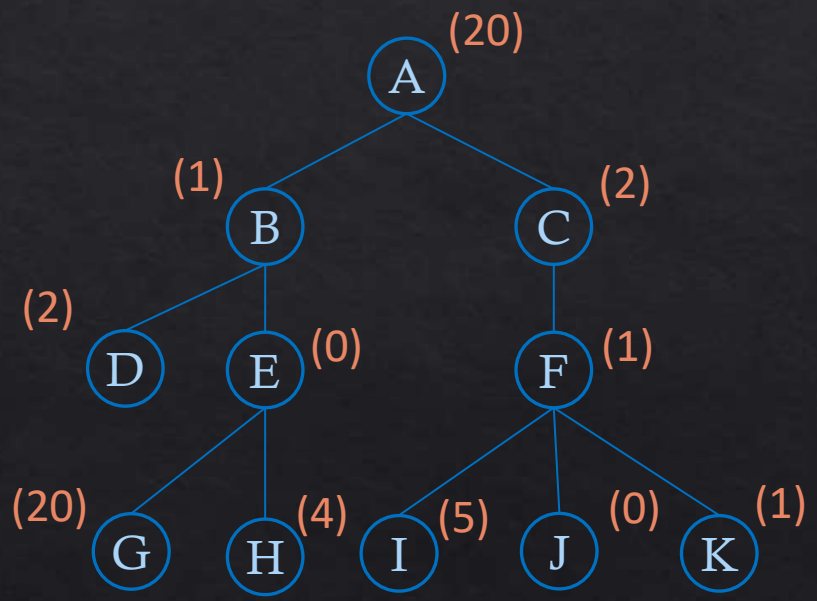


Question 11

For the following tree, write down the order in which nodes are visited using:

Hill Climbing (to find a minimum)

(expand the leftmost node in the case of ties)



Short Answer: A B E

Worked Solution:

We start at A, which has a value of 2. The children of A are B and C, and these have values 1 and 2 respectively. 1 is the best (smallest) choice, and it is less than A, so we select B.

At B, we have a choice of D (2) and E(0). E is best, and it is smaller than A, so we choose E next.

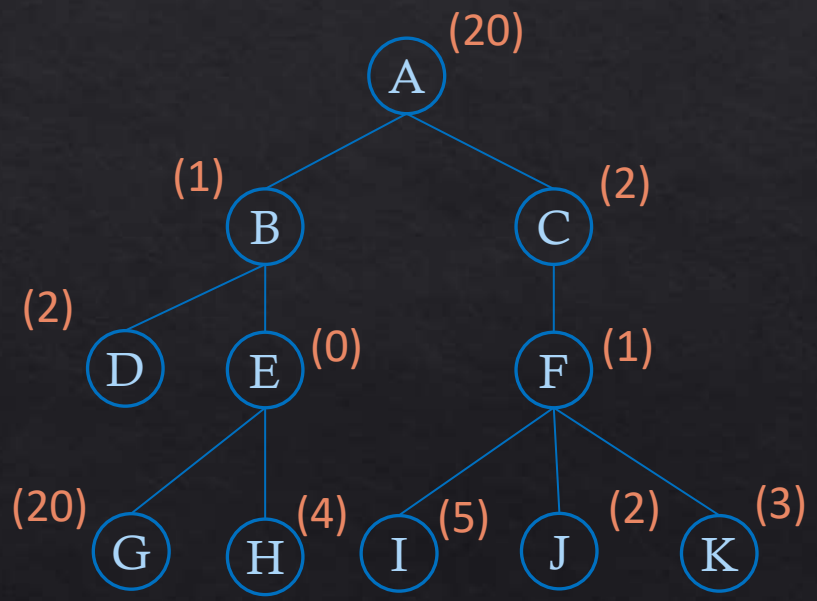
The children of E are G and H. Both of these have values larger than E, so we do not select either of them and instead stop our search.

Question 12

For the following tree, circle the children selected at each level using:

Beam Search (to find a minimum, with k=2)

(expand the leftmost node in the case of ties)

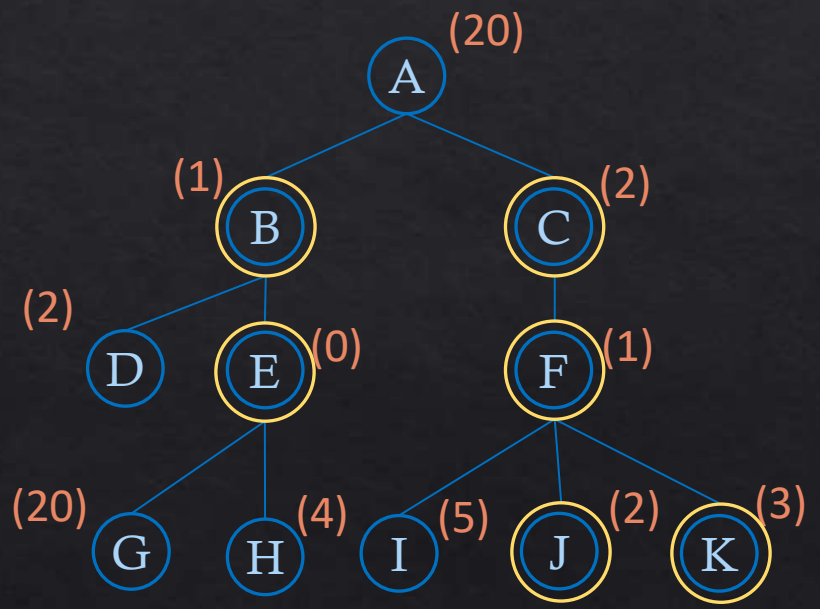


Question 12

For the following tree, circle the children selected at each level using:

Beam Search (to find a minimum, with k=2)

(expand the leftmost node in the case of ties)



Short answer: BC | EF | JK

Explanation:

We start at A, and select the best two children (B and C).

Combined, B and C have 3 children: D, E and F. The best two of these are E and F, so we choose these.

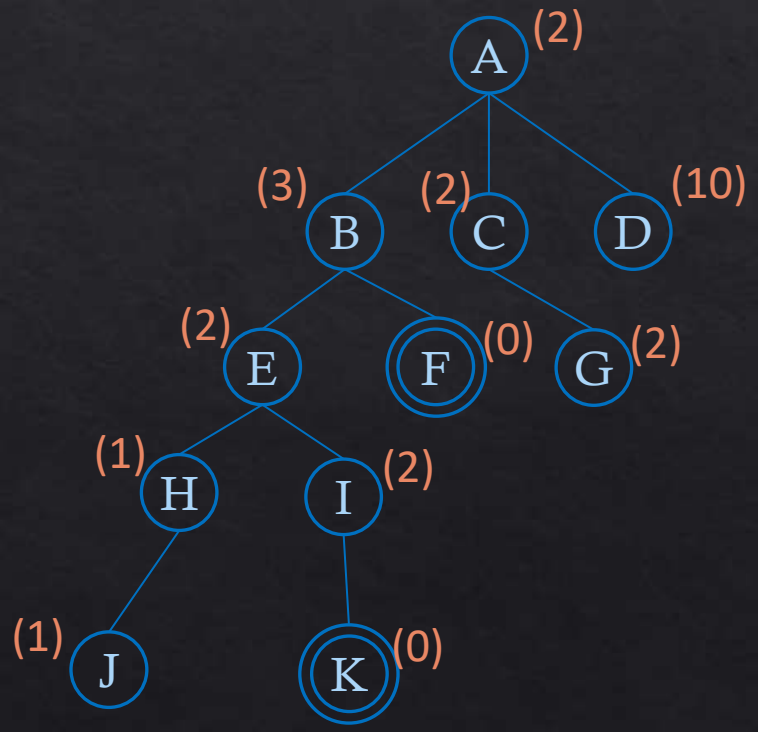
Combined, E and F have 5 children: G, H, I, J, K. The best two of these are J and K, so we select these. (Note: J and K are worse than their parents, but we still select them. This is different from hill climbing, where we don't choose a child unless it's better)

Question 13

For the following tree (there are goal nodes!), write down the order in which nodes are visited using:

Hill Climbing and Beam Search

(expand the leftmost node in the case of ties)

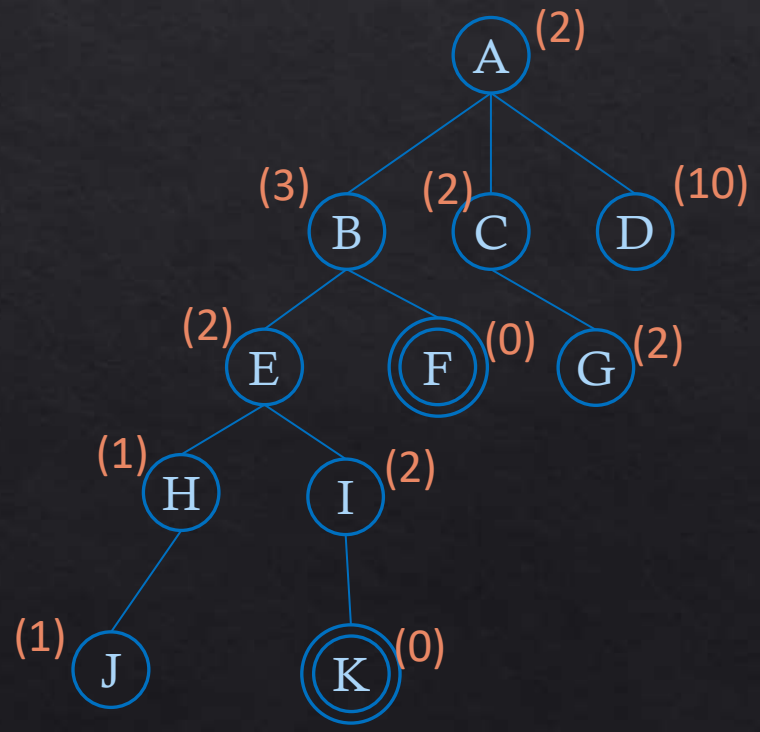


Question 13

For the following tree (there are goal nodes!), write down the order in which nodes are visited using:

Hill Climbing and Beam Search

(expand the leftmost node in the case of ties)



Short Answers:

Hill Climbing:

A

Beam Search:

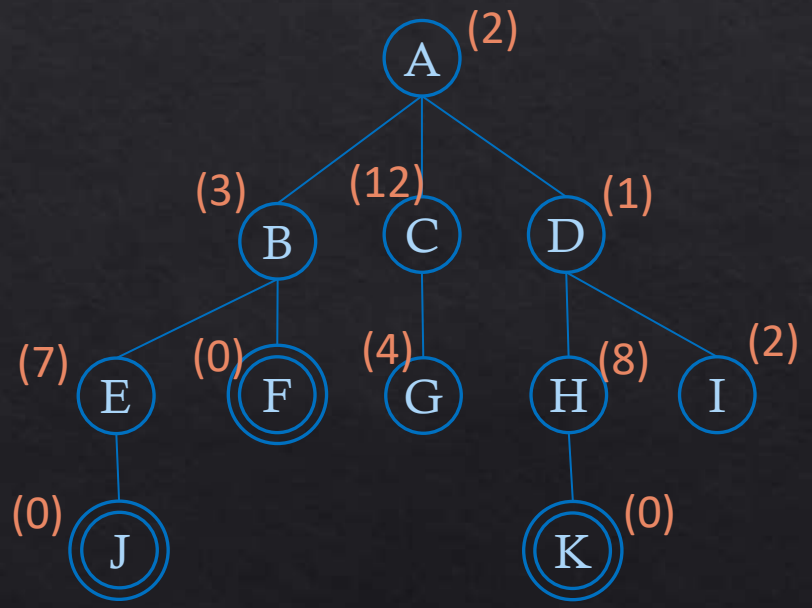
A B C E F

Question 14

For the following tree (there are goal nodes!), write down the order in which nodes are visited using:

Hill Climbing and Beam Search

(expand the leftmost node in the case of ties)

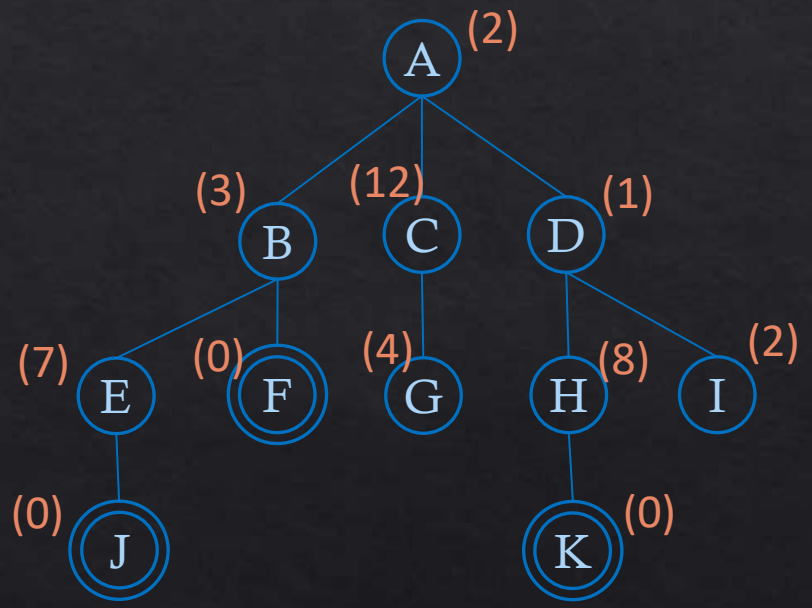


Question 14

For the following tree (there are goal nodes!), write down the order in which nodes are visited using:

Hill Climbing and Beam Search

(expand the leftmost node in the case of ties)



Short Answers:

Hill Climbing:

A D

Beam Search:

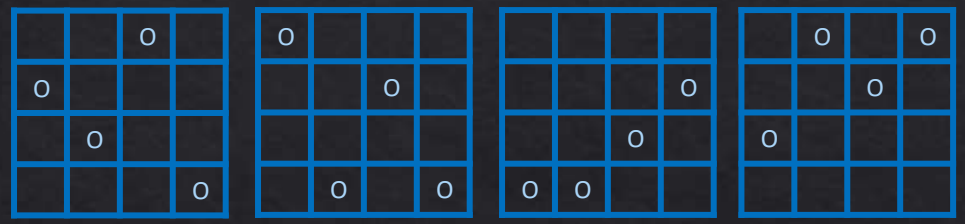
A B D F

Excellent tree searching!

Question 15

We are trying to solve the 4 queens puzzle, where we need to place 4 queens on a 4 by 4 board so that none are attacking. Apply the following genetic algorithm (for one iteration) and write down the next population.

Initial population:

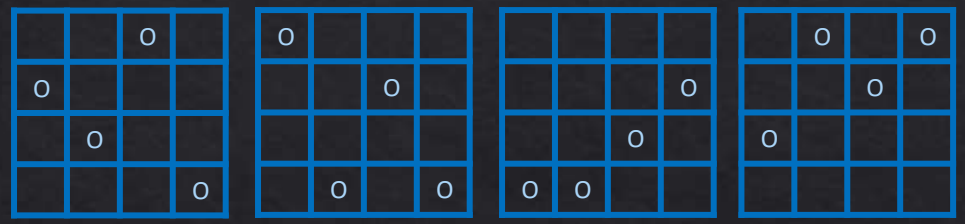


Fitness function: the number of pairs of non-attacking queens
Selection: select for crossover a) the fittest and second most fit individual and b) the fittest and third most fit individual
Crossover: cross the first two columns of each individual with the last two columns of the other
Mutation: Move the queen in the third column down by one

Question 15

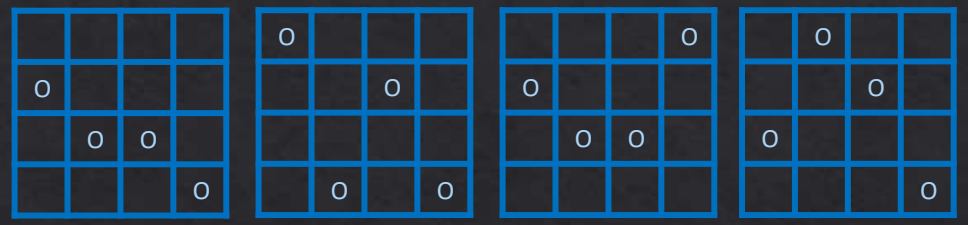
We are trying to solve the 4 queens puzzle, where we need to place 4 queens on a 4 by 4 board so that none are attacking. Apply the following genetic algorithm (for one iteration) and write down the next population.

Initial population:

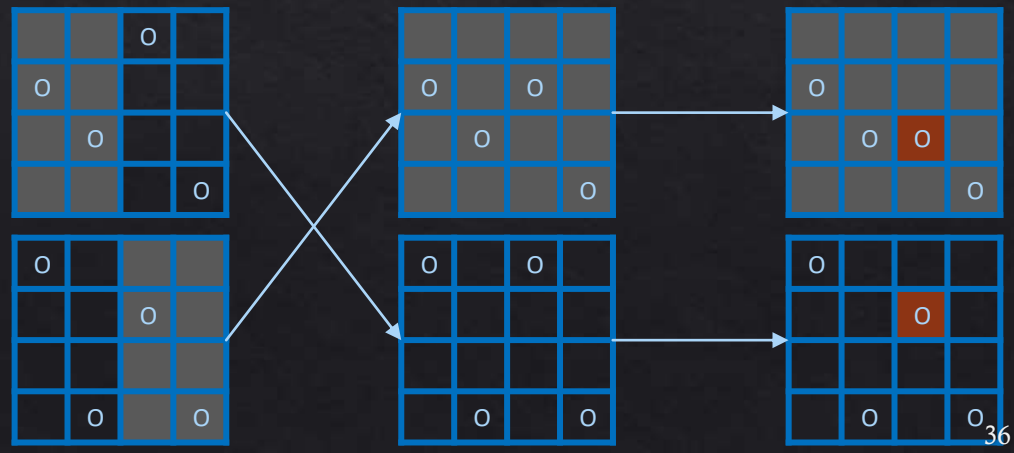


Fitness function: the number of pairs of non-attacking queens
Selection: select for crossover a) the fittest and second most fit individual and b) the fittest and third most fit individual
Crossover: cross the first two columns of each individual with the last two columns of the other
Mutation: Move the queen in the third column down by one

Short Answer:



Explanation:
The fitness levels of the original individuals are: 5, 4, 2 and 3. We cross the fittest individual (1) with the second most fit (2) and perform the mutation step:



We do the same for the fittest individual (1) and the third most fit (4) and end up with the new population above

Question 16

For the following graph, write down the nodes visited when running the Simulated Annealing algorithm for 8 iterations starting from node S and searching for a minimum. Set T = 30 initially and decrease it by 20% after each iteration (ie. T = T*0.8). You should accept a worse successor with probability

$$p = e^{\frac{v(n)-v(m)}{T}}$$

where v(n) is the value of the parent node and v(m) is the value of the successor.

Actually, hang on a sec!!! This algorithm involves randomness! We're all going to get different answers!! We're going to have to be a bit cunning here and use the same random numbers to stop this from happening:

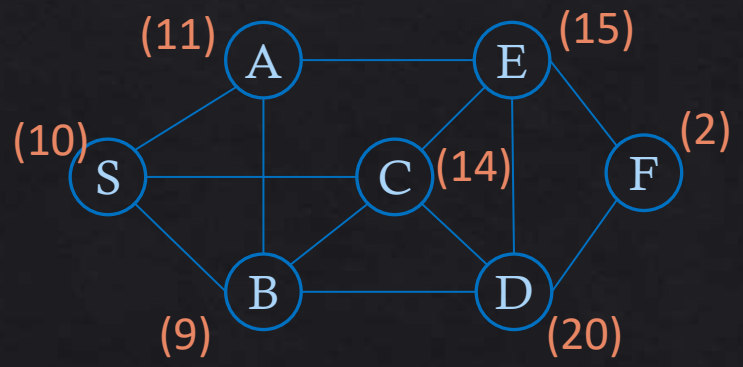
0.78 0.32 0.26 0.67 0.49 0.72 0.95 0.02 0.51
0.28 0.81 0.39 0.32

To choose a random successor of a node

1. List all the successors of that node (Order them alphabetically) eg. A B C D
2. Take the next random number from the list eg. 0.22
3. Multiply that number by the number of successors eg. 0.22 *4 = 0.88
4. round the result down to the nearest integer eg. 0
5. choose the successor corresponding to that number eg. A

To decide whether to accept a "bad" successor

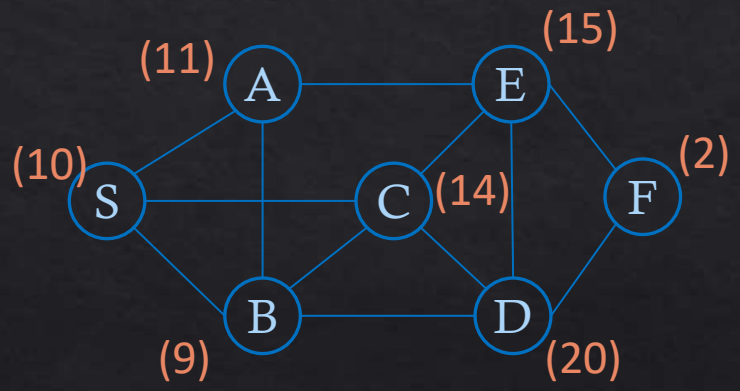
1. calculate p using the formula to the left
2. take the next random number from the list
3. if that random number is less than p, accept the successor



Question 16

For the following graph, write down the order in which nodes are visited using:

Simulated Annealing



Short Answer:

S C D C C B D F F

Worked Solution:

We start at S with T=30. We choose a random successor of S following the steps on the previous slide: select the next random number from the list (0.78), multiply by the number of successors (0.78 * 3 = 2.34) and round down (2). We got a 2, so we choose successor number 2. This is C because, when we order the successors alphabetically (A B C), A is in position 0, B is in position 1 and C is in position 2.

C is worse than S, so we don't automatically accept it. We first calculate p = e ^ ((10-14)/30) = 0.875 . Now, we take the next random number (0.32) and, since it is less than 0.875, we accept C and move to it.

Now we need to reduce the temperature (T = 30*0.8 = 24) and repeat the process. On the next iteration, we move to D, then we move to C, then we decide to stay on C instead of moving to E, then we move to B, then D, then F, then we decide to stay on F instead of moving to D

Almost a-kneeling quokka!

Part 2 – General Knowledge Practice

Question 1

What is an admissible heuristic?

Question 1

What is an admissible heuristic?

It is a heuristic that never overestimates the cost to the goal node.

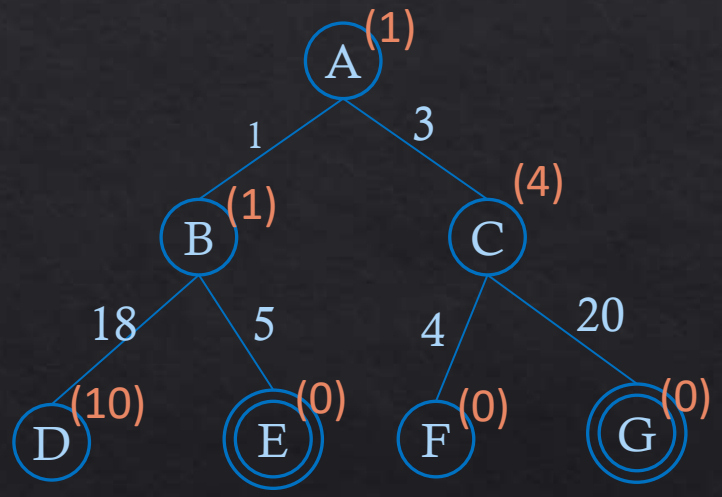
That is, for each node, it either:

1. underestimates the cost to the goal node, or
2. gives the exact cost to the goal node.

Let’s practise this! :)

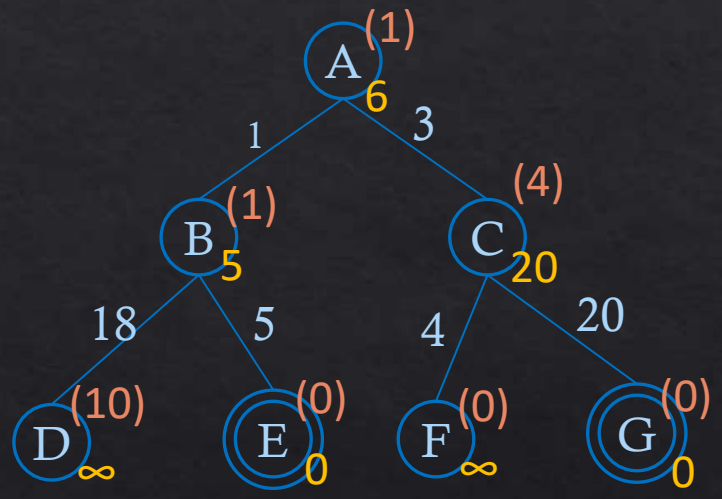
Question 2

For the following tree, is the heuristic (shown in orange) admissible?



Question 2

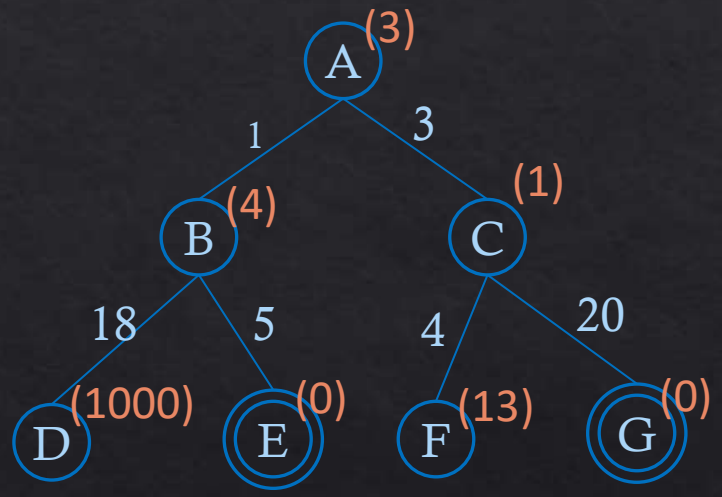
For the following tree, is the heuristic (shown in orange) admissible?



Yes. The heuristic values are never larger than the actual costs, shown in yellow, so the heuristic is admissible

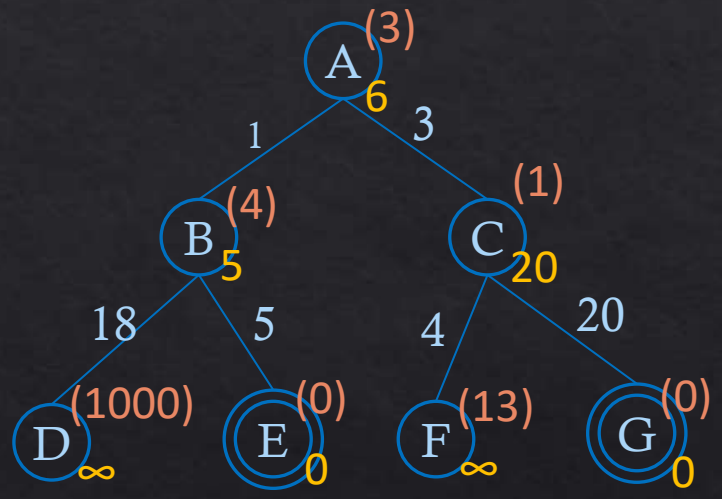
Question 3

For the following tree, is the heuristic (shown in orange) admissible?



Question 3

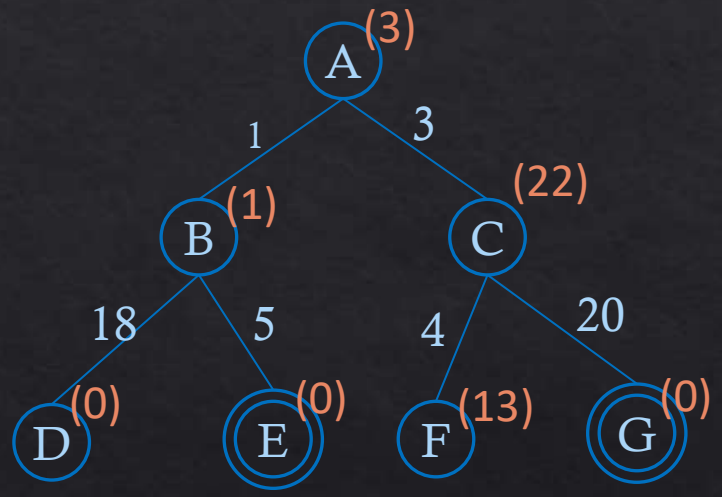
For the following tree, is the heuristic (shown in orange) admissible?



Yes. The heuristic values are never larger than the actual costs, shown in yellow, so the heuristic is admissible. Note that the actual cost is ∞ on paths that do not lead to a goal node.

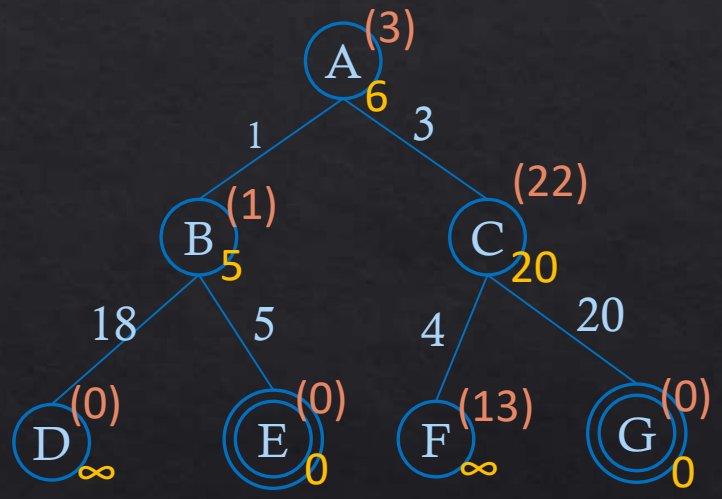
Question 4

For the following tree, is the heuristic (shown in orange) admissible?



Question 4

For the following tree, is the heuristic (shown in orange) admissible?



No. The heuristic overestimates to cost to a goal node from node C.

Keep it up!

Quokka Quote:

I HOP you're enjoying these questions!

Question 5

What does it mean to say a search strategy is complete?

Question 5

What does it mean to say a search strategy is complete?

It means it will always find a goal node if one exists!

Question 6

Assume we are searching a tree with a finite branching factor, infinite depth and no restrictions on step costs. On top of this, we could have any heuristic.

Under these conditions, which of the following search strategies would be complete ?

1. BFS
2. DFS
3. UCS
4. IDS
5. A*
6. Greedy Search

Question 6

Assume we are searching a tree with a finite branching factor, infinite depth and no restrictions on step costs. On top of this, we could have any heuristic.

Under these conditions, which of the following search strategies would be complete ?

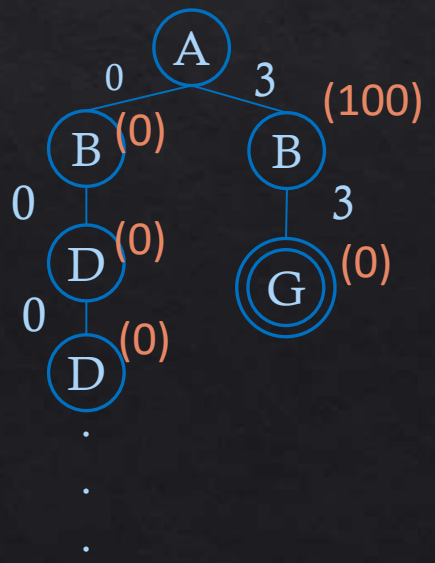
1. BFS
2. DFS
3. UCS
4. IDS
5. A*
6. Greedy Search

Short Answer: BFS, IDS

Explanation:

BFS and IDS are always complete if the branching factor is finite

The rest could get stuck exploring infinitely deep branches with no goal nodes, as in this example:



Question 7

What does it mean to say a search strategy is optimal?

Question 7

What does it mean to say a search strategy is optimal?

It means it will always find the best goal node – that is, the goal node that costs the least to get to from the root.

Question 8

Assuming the branching factor is finite, under what condition(s) is BFS optimal?

Question 8

Assuming the branching factor is finite, under what condition(s) is BFS optimal?

If all the step costs are the same, BFS if optimal.

(In general, if the cost to reach each node is a non-decreasing function of it's depth, then BFS if optimal.

That just means all the nodes with the same depth have the same cost, and deeper nodes never cost less than shallower ones)

Question 9

What is the difference between informed and uninformed search strategies?

Of the following strategies, which are informed, and which are uninformed?

1. BFS
2. Greedy Search
3. IDS
4. A*
5. DFS
6. UCS

Question 9

What is the difference between informed and uninformed search strategies?

Of the following strategies, which are informed, and which are uninformed?

1. BFS
2. Greedy Search
3. IDS
4. A*
5. DFS
6. UCS

Informed search strategies use a heuristic (ie. they have some way of estimating the cost to a goal node), but uniformed search strategies do not!

Informed:

A*, Greedy Search

Uninformed:

BFS, IDS, DFS, UCS

Thumbs up!

Speaking of the last question, it’s time to inform you of your fantastic progress!

Keep it up! :D

Question 10

What does it mean for a heuristic to be consistent?

Question 10

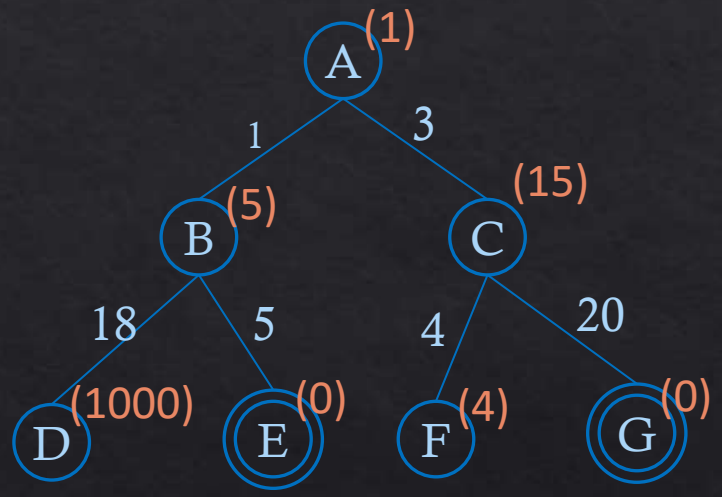
What does it mean for a heuristic to be consistent?

It means it satisfies the triangle inequality. This is true if and only if the f-values along every path are non-decreasing.

Let's practise this! :)

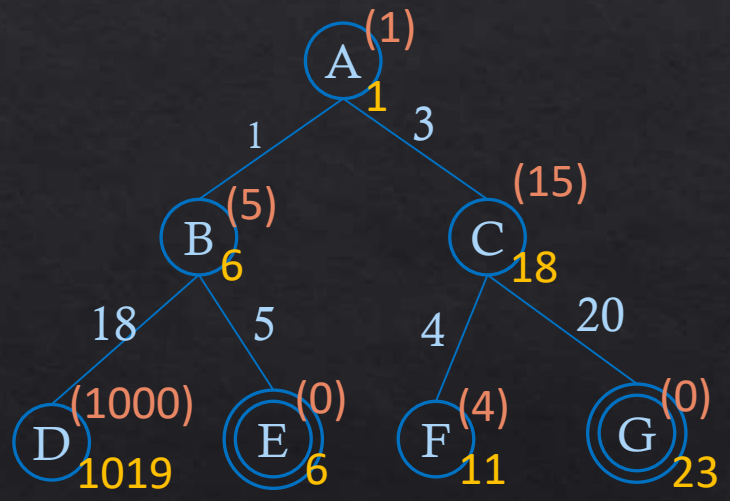
Question 11

For the following tree, is the heuristic (shown in orange) consistent?



Question 11

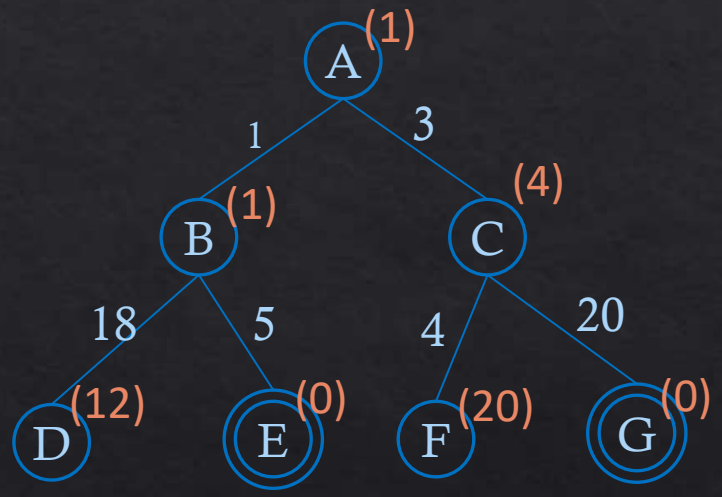
For the following tree, is the heuristic (shown in orange) consistent?



No. The f-values decrease along the path from C to F: at C the f-value (shown in yellow) is 18, and this decreases to 11 at F.

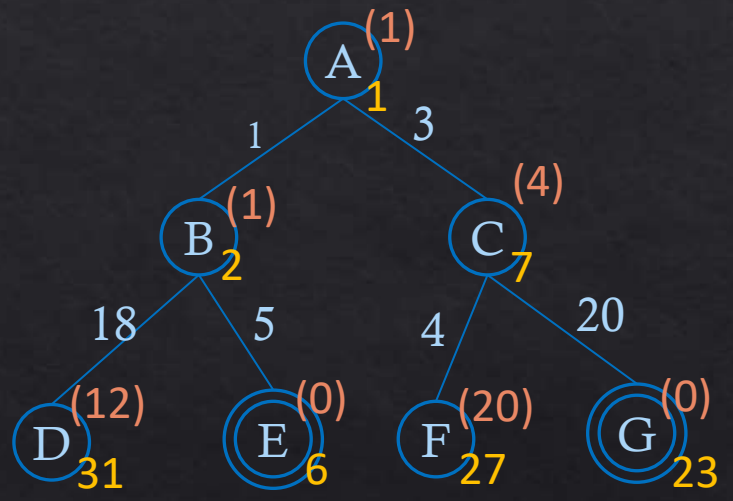
Question 12

For the following tree, is the heuristic (shown in orange) consistent?



Question 12

For the following tree, is the heuristic (shown in orange) consistent?



Yes. The f-values are all non-decreasing.

Question 13

What does it mean for an admissible heuristic to dominate another admissible heuristic?

Question 13

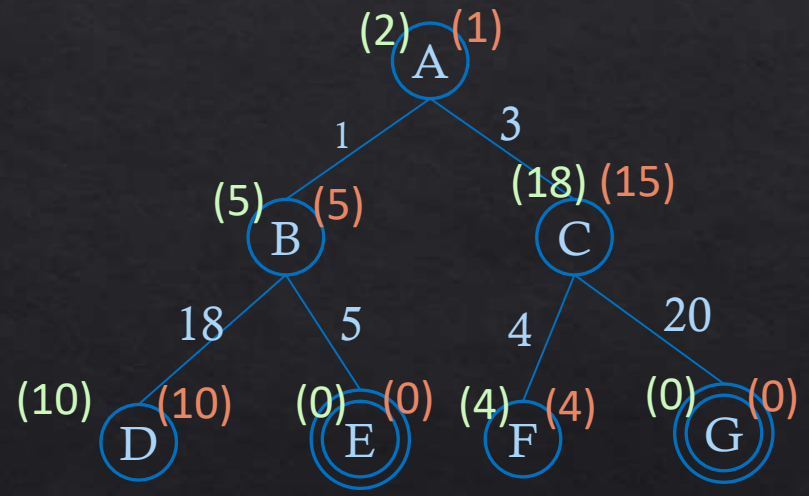
What does it mean for an admissible heuristic to dominate another admissible heuristic?

One admissible heuristic dominates another if, for all nodes, it gives an estimate that is equal to or greater than the other heuristic's estimate. That is, it is dominant if it gives a better estimate of the true cost to a goal node.

Let's practise! :D

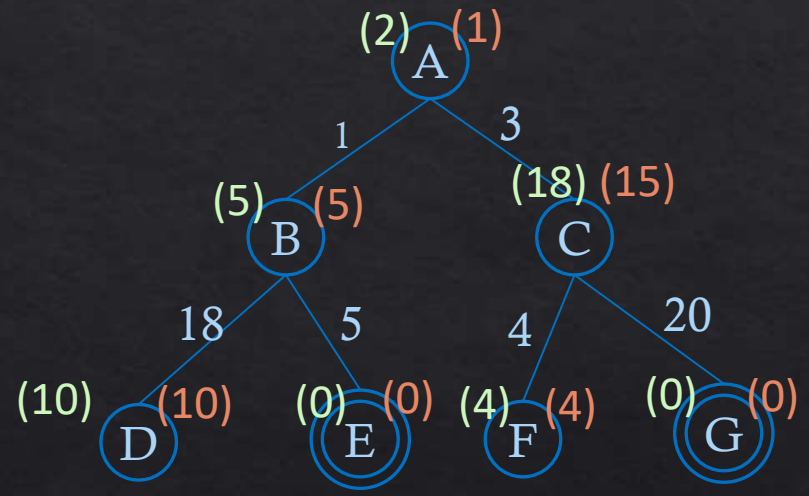
Question 14

For the following tree, which of the two admissible heuristics, if any, are dominant? (One heuristic is shown in orange and the other in green)



Question 14

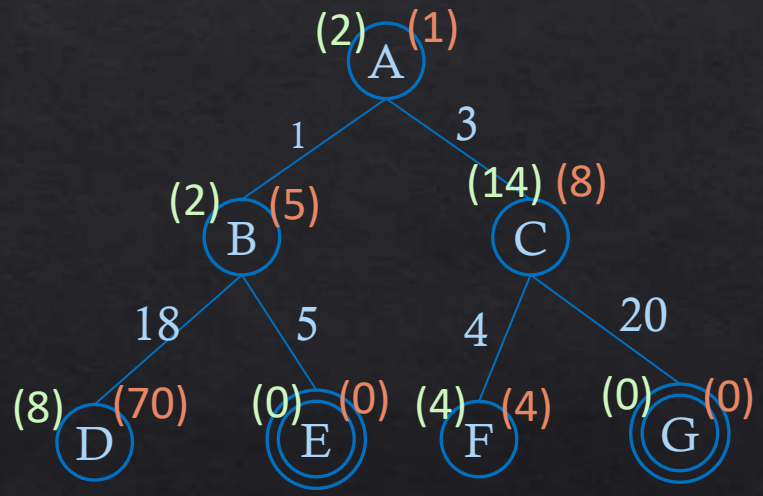
For the following tree, which of the two admissible heuristics, if any, are dominant? (One heuristic is shown in orange and the other in green)



Green. For all nodes, the green heuristic values are greater than or equal to the orange ones.

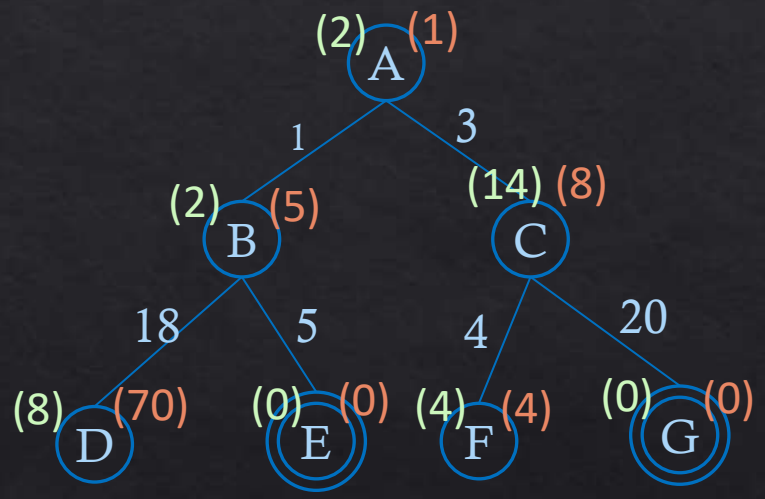
Question 15

For the following tree, which of the two admissible heuristics, if any, are dominant? (One heuristic is shown in orange and the other in green)



Question 15

For the following tree, which of the two admissible heuristics, if any, are dominant? (One heuristic is shown in orange and the other in green)



Neither. Sometimes the green heuristic values are larger and sometimes the orange ones are larger.

In this situation, we could define a new heuristic as the max of both of these, and this new heuristic would dominate both.

It’s great to have the best search method!

Question 16

When is BFS a special case of A*?

Question 16

When is BFS a special case of A*?

When f(n) = depth(n)

See lecture 3a for more details. :)

Question 17

What is a relaxed problem?

Question 17

What is a relaxed problem?

It's a problem with fewer restrictions on what you can do.

For example, a relaxed version of the 8-puzzle would be a puzzle where you could pick up tiles and place them anywhere in a turn, instead of only being able to slide them along one place.

See the week 3 lecture slides for more details.

Question 18

How do relaxed problems relate to admissible heuristics?

Question 18

How do relaxed problems relate to admissible heuristics?

If you calculate the cost to solve a relaxed version of a problem, then this gives you an admissible heuristic for the original problem.

General Knowledge Suggestion

Do you have a general understanding of all of the following? If not, try revising the topics you're not too sure about!

Search Algorithms:

- BFS
- DFS
- IDS
- UCS
- A*
- Greedy Search
- Hill Climbing
- Beam Search
- Simulated Annealing
- Genetic algorithms

Comparing Algorithms:

- Optimality
- Completeness
- Time Complexity
- Space Complexity

Heuristics:

- Admissibility
- Consistency
- Dominance
- Problem Relaxations

Part 3 – Test Your Understanding/ Think

Question 1

Will IDS always find the same goal node as BFS if we are searching a tree? Why or why not?

(Assume we are applying them in the same way as in the tutorials)

Question 1

Will IDS always find the same goal node as BFS if we are searching a tree? Why or why not?

(Assume we are applying them in the same way as in the tutorials)

Yes, it will.

Clearly if the root is a goal node, then BFS and IDS will both stop here.

What if there is a goal node at depth 1? BFS will expand nodes at depth 1 from left to right, and so will IDS. So, if there are any goal nodes at depth 1, they will both choose the leftmost one

What if there aren't any goal nodes at depth 1, but there are at depth 2? BFS will again expand the nodes at this depth from left to right, and so too will IDS (remember we are only considering the ordering in which they expand the nodes at depth 2 and not other depths). They will both choose the leftmost goal node.

Similarly, if there isn't a goal node at depth 2, but there is at depth 3, again they will both choose the leftmost goal node at depth 3.

And so on…

BFS and IDS will both choose a goal node at the minimum depth and, if there are multiple goal nodes at the same depth, they will both choose the leftmost one

So, yes IDS will find the same solution as BFS!

Question 2

In the week 3 tutorial, question 6 was a genetic algorithm question. Can you think of a simple reason why you'd expect the average fitness of the population to increase on each iteration when applying that algorithm?

Question 2

In the week 3 tutorial, question 6 was a genetic algorithm question. Can you think of a simple reason why you'd expect the average fitness of the population to increase on each iteration when applying that algorithm?

On each iteration, the worst individual is forgotten – we only use the best 3 individuals to generate the next population. So, you'd expect the average fitness to usually be better in the next population (though, due to mutations it may not always be better)!

Are these questions STRETCHING your imagination?

Question 3

In lectures, we talked about the relationship between consistency and f values. We said:

- If a heuristic is consistent, then the f values are non-decreasing
- If the f values are non-decreasing, the heuristic is consistent

Can you explain intuitively why this makes sense?

That is, how could you explain this without giving a formal proof? Try to use an example!

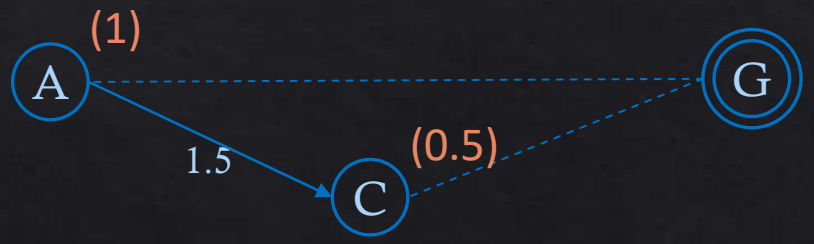
Question 3

In lectures, we talked about the relationship between consistency and f values. We said:

- If a heuristic is consistent, then the f values are non-decreasing
- If the f values are non-decreasing, the heuristic is consistent

Can you explain intuitively why this makes sense?

That is, how could you explain this without giving a formal proof? Try to use an example!



Firstly, we know that a heuristic is consistent if it satisfies the triangle inequality. For example, in this graph, the heuristic is consistent because

$$h(A) \leq 1.5 + h(C).$$

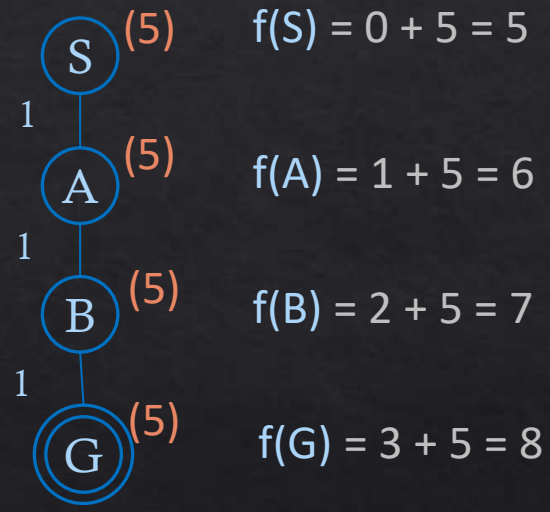
Let's think about what this means. 1.5 + h(C) is like the estimated distance from A to G if you go through C, whereas h(A) is the estimated direct distance. So, that inequality is basically just saying that our estimate for the route through C shouldn't be shorter than the direct distance (detours should take longer!).

Now then, if the estimated distance from A to G increases if you go through C, then we'd expect the estimated total distance from the start to G via A to increase if we have to go through C as well.

But wait, the estimated total distance is just the f-value! We're basically saying f(A) should be ≤ f(C). That is, we'd expect the f values to be non-decreasing.

Question 4

What's wrong with the following argument?
Mwahahahhahaha!!! The lecture slides are wrong!!!!
They say consistency implies admissibility, but that's not true, and here's a counter example to prove it:



f(S) = 0 + 5 = 5

f(A) = 1 + 5 = 6

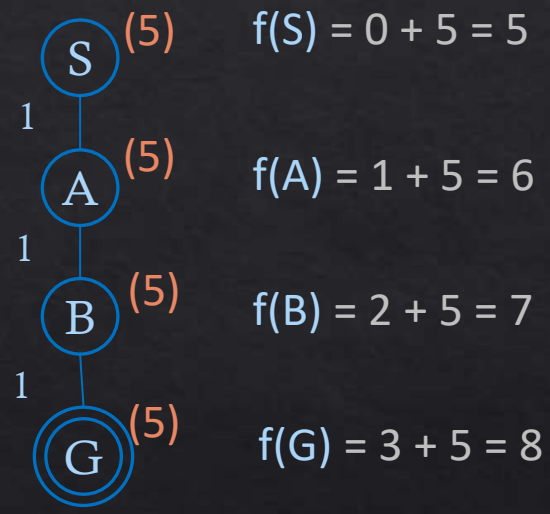
f(B) = 2 + 5 = 7

f(G) = 3 + 5 = 8

Here, the heuristic is consistent because the f-values are non-decreasing, but it's not admissible because the heuristic overestimates the cost to the goal nodes.
NOTE: the slides aren't actually wrong! But why? What's wrong with this argument?

Question 4

What's wrong with the following argument?
Mwahahahhahaha!!! The lecture slides are wrong!!!!
They say consistency implies admissibility, but that's not true, and here's a counter example to prove it:



f(S) = 0 + 5 = 5

f(A) = 1 + 5 = 6

f(B) = 2 + 5 = 7

f(G) = 3 + 5 = 8

Here, the heuristic is consistent because the f-values are non-decreasing, but it's not admissible because the heuristic overestimates the cost to the goal nodes.
NOTE: the slides aren't actually wrong! But why? What's wrong with this argument?

Remember that our heuristic, h, must estimate the cost at a goal node to be 0!!

We cannot say h in this example is consistent, because it isn't even well defined (h(G) should be 0, not 5!)!

But what if we change h(G) from 5 to 0? Then f(G) = 3 and f(G) < f(B). Ie. f is no longer non-decreasing, meaning it is not consistent.

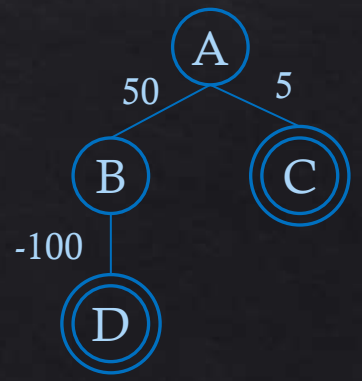
So, rest easy – this counter example isn't actually a counter example, and there's nothing wrong with the lecture slides! :D

Question 5

UCS is complete if the step costs are strictly greater than some small positive value, epsilon, but what's wrong with having other step costs?

Take the example shown here. Even though there's a path with a negative cost, UCS will still find a solution (C). And, if node C wasn't there, it would still get to node D. UCS may no longer be optimal, but surely it will still find a solution, won't it?

Can you think of an example where UCS wouldn't be complete if steps costs less than epsilon were allowed?

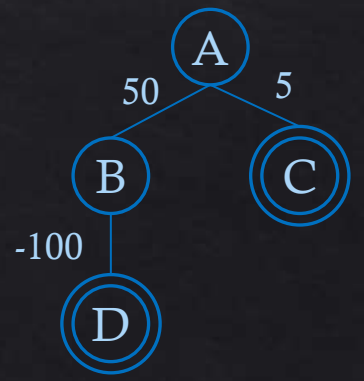


Question 5

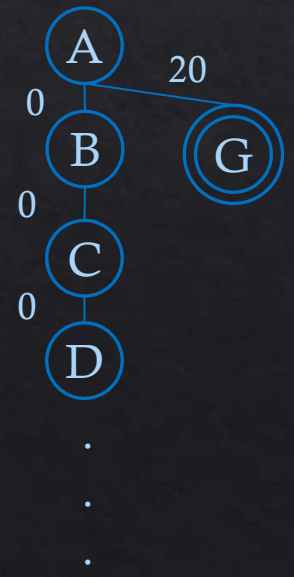
UCS is complete if the step costs are strictly greater than some small positive value, epsilon, but what's wrong with having other step costs?

Take the example shown here. Even though there's a path with a negative cost, UCS will still find a solution (C). And, if node C wasn't there, it would still get to node D. UCS may no longer be optimal, but surely it will still find a solution, won't it?

Can you think of an example where UCS wouldn't be complete if steps costs less than epsilon were allowed?



Imagine a tree like this, where you have a branch that goes on forever. On top of this, imagine that all the nodes in that branch cost 0 to get to. Since UCS always expands the cheapest node, it will get stuck expanding nodes on that branch forever! And it will never reach the goal node… :,(



Question 6

The berry sort problem is a problem where we need to sort n berries for a quokka. There is one berry of each size between 1 and n (ie. one of size 1, one of size 2, one of size 3 etc.) and we need to sort them in ascending order. The berries start in a random order in a line, and each turn we can swap two berries that are right next to each other.

Can you come up with an admissible heuristic for this problem by working out a relaxed version of the problem?

Question 6

The berry sort problem is a problem where we need to sort n berries for a quokka. There is one berry of each size between 1 and n (ie. one of size 1, one of size 2, one of size 3 etc.) and we need to sort them in ascending order. The berries start in a random order in a line, and each turn we can swap two berries that are right next to each other.

Can you come up with an admissible heuristic for this problem by working out a relaxed version of the problem?

A relaxed version of this problem could be the same problem, but where we can swap any two berries and not just berries right next to each other. The admissible heuristic would be the number of swaps needed to solve this relaxed problem.

BERRY good work,

BERRY good work, beLEAVE me!

Question 7

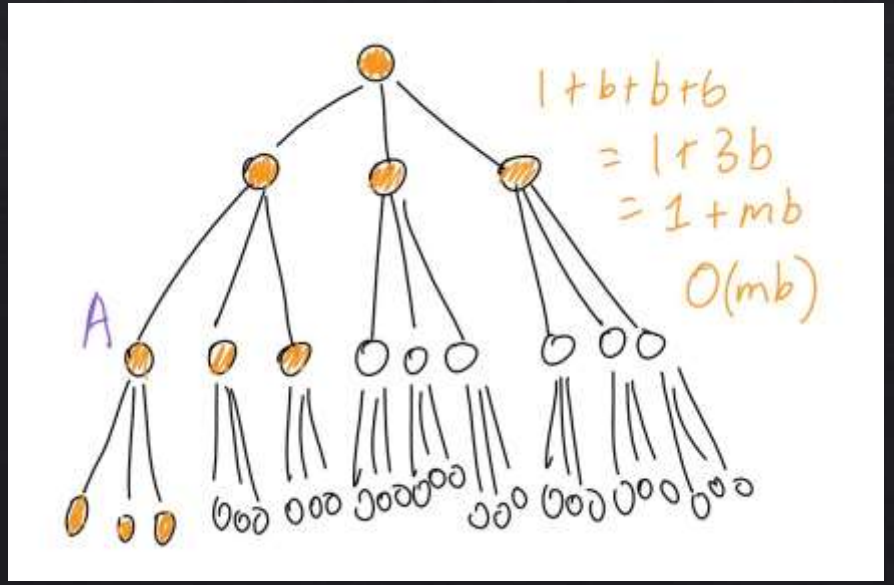
What is the space complexity of DFS in terms of b, the branching factor, m, the maximum depth and d, the depth of the least-depth solution. Explain why.

Question 7

What is the space complexity of DFS in terms of b, the branching factor, m, the maximum depth and d, the depth of the least-depth solution. Explain why.

For DFS, we don't need to keep a node in memory once we've explored all of its descendants.

So, what's the maximum number of nodes we can have in memory? The number of nodes highlighted in orange in the diagram are the maximum number in the case where m=3, b=3. We see for a general tree, in the worst case we only need to store 1 + mb nodes in memory, so the space complexity is O(bm)



Question 8

What is the time complexity of DFS in terms of b, the branching factor, m, the maximum depth and d, the depth of the least-depth solution. Explain why.

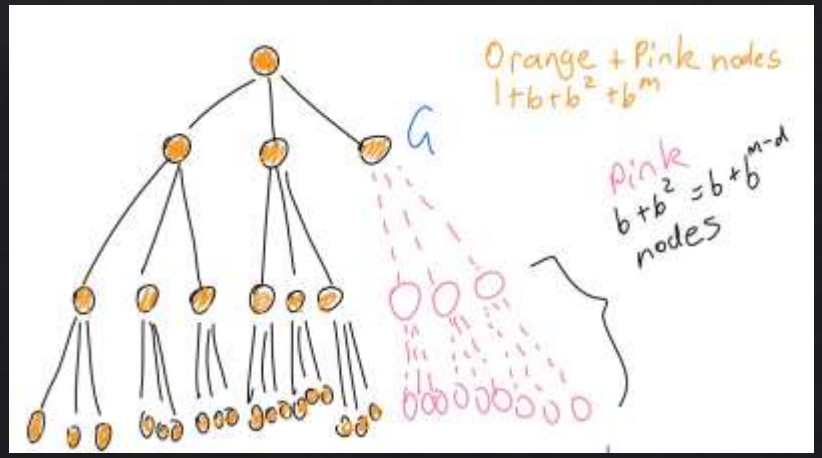
Question 8

What is the time complexity of DFS in terms of b, the branching factor, m, the maximum depth and d, the depth of the least-depth solution. Explain why.

The diagram shows the number of nodes that would need to be expanded (orange) if m=3, d=1, b=3. The pink nodes are just there for the calculation and aren't actually nodes.

We can see the total number of nodes (orange + pink) is $1 + b + b^2 + b^3$, and the total number of pink nodes is $b + b^2$. So, the number of orange nodes is $1 + b + b^2 + b^3 - b - b^2 = 1 + b^3$.

In general, in the worst case, we would need to expand/ add to the fringe $(1 + b + b^2 + b^3 + \dots + b^m) - (b + b^2 + \dots + b^{m-d})$ nodes, which is $1 + b^{m-d+1} + b^{m-d+2} + \dots + b^m$ nodes. So, the time complexity is $O(b^m)$.



All Done!
Well done, and best of luck for the rest of the semester!!!

(PS. I hope you now love quokkas as much as I do…)