We are going to discuss two algorithms for graphs, the greedy search and a search. So let's say we have a road map. In this case we have the map of Romania and we are trying to find our way to go from Arad to Bucharest. Since we don't know which way to go, we might have to try all of the different roads. Well, so the first algorithm we are going to look at is called greedy search.

And greedy search uses a heuristic. Greedy search in this case is going to use the straight line distance to Bucharest as the heuristic. And heuristic is going to help us get closer and closer to the goal. Heuristics are not meant to be perfect, so they are going to be an estimate and we want our heuristic to be a lower estimate. So the heuristic estimate should be less than the actual cost of going there.

So for example, if we look at this Romania graph, the estimate for ARAD is 366km. The estimate for CBU is 253km, which says, well, CBU is closer than Orat. If we look at oradea, oradea is 380 kilometers. And so auradero is farther away based on our estimate. So let's look at our greedy search.

We are going to start from Arad, we are going to want to go to Bucharest, and the greedy search is going to use the heuristic, and the actual cost is not considered in greedy search. So we are going to have a data structure, we are going to use a priority queue, and we want to keep picking the item that is most promising, that says, hey, I am the closest one to the destination. I am the closest one to book arrest. So we are going to have our greedy cost or greedy estimate as the first item in our priority queue. And the second item in our priority queue is going to be the actual cost it took to get to this location, to get to the city and the name of the city.

We are not actually going to use the actual cost except at the very end we'll say, hey, this is the search that we have and this is the price that we have found from greedy search. And we're also going to keep an unordered set to see if we have visited a city or not, so that we don't go around in loops. So we start our algorithm by pushing into the priority queue the heuristic of the start, in this case Arad, which in our case is going to be 366. So we are going to have the 366 as the heuristic. And then actual cost to get here has been zero.

And our starting city is Arat. So if we continue on we can look at. We only have one item in the priority queue, 366 with ARAD as the city. So our greedy search is starting from Arad, trying to get to Bucharest and we'll keep processing until we get to our destination. Or if the priority queue is empty, that means, well, we weren't able to find it.

The first thing we are going to do is we are going to take the things out of the priority queue. So the top item, which is the heuristic estimate, we have the actual cost that it has cost to get here and the current node that we are at. So we are visiting Arad. It costs zero to get here and we have a heuristic cost of 366. If we are at the targets, great, we would return the cost.

If not, we want to make sure whether we have visited this place before or not. If we haven't, then we can continue. We'll put it into our visited one and then we are going to look at all of the neighbors of Arat, all of the neighbors of the city and each one is going to have have the distance to get to that city. So for example we look at Timisiora. So Tim Siora has a heuristic cost of 329 and it would cost 118 to get there.

So we push it into the priority queue. The first item is the heuristic cost. That's how it's going to get sorted. So 329 and we are keeping the actual cost to get to Arad and then to Timisiora and then the city we are going to be at Timisiora, continuing on with Arad's other neighbors. The next one is CBU for heuristic cost of 253 and the next one after that is 374Zeret.

Now that we have processed all the neighbors, we go to our priority queue and take out the top element. The lowest element based on the heuristic cost is cpu. So going from ARA to CPU looks like the best possible way. And we haven't reached the target. We have not been to CBU before.

So we put CPU into our set. Then we look at CPUs neighbors. So one possibility is to go to fogarat. Heuristic cost of 178. That seems like a good estimate.

That's getting closer. Or the next one is Oradea. Well, CBU to Oradea, that's going to be moving away and has a much higher heuristic cost, 380. And the next one after that is RIM, Niki Vilcea, that has a cost of 193. And of course we could go back to Arad, but then that will have a heuristic cost of 366.

So that's not something that we would want.

So we have put all of these in the priority queue, and based on that we take our next item from the priority queue and the next item is Fagaras. So we have gone from Arat to CBU to Fagaras. So we are getting closer to Bucharest. And we haven't.

We haven't gotten to the target, we haven't been to Fagaras before. So we insert into Fagarads into our visited set. We look at our neighbors. Well, from Fagarest we can get to Bucharest, and the heuristic cost says zero. That's great.

So we put it into our queue, or we can go to Ptesty, or we can go to cpu.

And now we look at our priority queue again and now what we have is we are at Bucharest. So the cost to get here has been 417. The heuristic cost from here is zero because we are at at the target. So we return the actual cost. So we can say, hey, greedy search found a distance of 417.

Looking at the map, we have come from Arad, CBU, Fagaras, Bucharest. Was that the shortest path? It turns out it wasn't. It was a good path in the sense. We never had to backtrack, we never had to do searching.

We could just go straight towards Bucharest, but we did not find the shortest possible path. The shorter path would have been Arat, cbu, rimniq, ptsd, and then Bucharest, which takes us to our second algorithm A. So A is going to have a similar data structure, but it's going to keep track of both the heuristic cost and the actual cost, sorry, heuristic estimate. And the actual cost it took to get here and use both of those together as the sorting part for the priority queue. So once again we have a priority queue.

We are going to keep track of our score to the goal, how much road we have covered. We are also going to keep track of where we came from so that we can reconstruct the path later. And we have our set of visited nodes. So initially our. The distance we have covered is zero.

So that's our cost. And we push that one into Our into the priority queue. So we are starting our a star search. We have our current cost and current node. So we are starting from Arat with a current cost of 366.

That's the how much we have come here, how many kilometers zero plus the heuristic cost that makes it 366. So Arad is at 366. Next if we are at the target we would reconstruct the path. We would return. But we are not at the target yet.

So let's continue. We have not been to Arad so we put Arad into our visited one. And then we look at Arad's neighbors. So we look at first neighbor Timmy Siora. Timmy Siora has a distance of 118.

That's going to be the actual distance. And we don't have any distances established for timcrf. So we are going to go into going to our if loop, sorry if close saying hey, we found the short path to timisiora. That's 118 where we came from is our end. So we are now going to add Timisiora into our priority queue.

But when we add it we are going to use 118 the distance we have traveled plus the heuristic distance. So it's going to be 118 plus 329 both together. And that gets us 447. So and then we look at Arad's other neighbors. So next we add CBU with a 393.

And next we have xeric with 449. So given 447, 393 and 449 CPU is the best one. So that should be at the top of the priority queue. So we take that one out of the queue saying hey, we are looking at CPU with 393. We are not at the target yet.

We haven't visited CPU before. We put it into our visited and now we are going to look at CPUs possible neighbors. So we look at the first neighborhood is going to have a g score of 230 39. That's 140 plus 99. So 239 would be the actual distance to get there.

Since we hadn't been there that means that's a good way to get there. So that's our best distance to get to Fagaras where we came from is cbu. And so we are going to now add it to our priority queue. When we are adding it we are once again adding it. The distance we have come 239/ the heuristic distance 178.

So we are always adding the heuristic distance and the actual distance together into our priority queue. So now we add 417 Fagaras and continuing on with the other neighbors, we add Auredea with 671 and Rymniki Wilsea with 413. So given all of the ones we have in the priority queue, the next one we are sorry, we have to add one more neighbor or there's a possibility. Then we get to visit Rim Nikki Wilser. That has a cost of 413.

So we go through its neighbors and for each neighbor we add the potential connection so we can get to CBU or CRYOA as as part of it. So now if you look at our priority queue, we have 415 Pitesti, 417 Fagaras, 447 Timicero. Since we have 415 Pitesti, that's the one we are going to test next to see if there is a better good path from Pitesti. So we go from Pitesti and we look at Pitestes neighbors. Well, one of the Pitestes neighbors is Bucharest.

So we add Bucharest into our queue.

And but we, and but we don't stop our algorithm as soon as we find Bucharest. We have to wait for it to come out of the priority queue. So now we are looking at our priority queue and we are now visiting Bucharest with 4:15. So now we have found the target. And now that we have found the target, we can create the path how we came here using the came from map that we had started.

We started from the start part, we reverse our path and then we can print our path saying, hey, the path that we found is Arad, cbu, Rymniki, Wilsea, Pitesti and Bucharest. And this path is actually a little bit shorter at 415 rather than 417. And we return from our algorithm having completed it. So the important thing about a star is that the A algorithm is guaranteed to find you the shortest path as long as you have a heuristic that does not overestimate. That underestimates.

So let's think about the worst possible heuristic. The worst possible heuristic says you are, you are a distance of 0 km from Bucharest. In that case, what would happen? We would start from Arad, we would add our neighbors, and when we add our neighbors, the priority queues is going to get the numbers Actual cost plus heuristic cost. Well, actual cost is zero.

Sorry, heuristic cost is zero. Actual cost is going to be 75, 140 and 118. So that means we are going to go to Xerint as the first try and then after that we'll go to Oradeus. Oradea is going to have 146 CPU at 140. Timmy Sierra 118.

So then we'll choose CPU. So with a heuristic of zero, we are still going to get the best possible answer. We are still going to get the shortest possible path, but instead of going directly towards our destination, we would do C, similar to what the Dijkstra algorithm was doing, and cover all the possible paths, make sure we get to everything at distance 100, then 200, then 300 and so on. So the good thing about greedy search is if we have a good heuristic, greedy search is very directed, it's got a very low memory cost and it goes towards the target really fast.

But it's not guaranteed to find the best solution. A, A has a bigger memory cost because we are keeping this priority queue that can grow very large. But A is guaranteed to find you the shortest possible path. So that was two new algorithms, greedy search and a search for looking at graphs and in this case, map of Romania.