Type of search	Problem	Path found	Path length	Node expanded
DFS	Humans, Robots and Ferry	H on left:3 R on left:3 H on right:0 R on right:0 ferry is on the left. H on left:2 R on left:2 H on right:1 R on right:1 ferry is on the right. H on left:3 R on left:2 H on right:0 R on right:1 ferry is on the left. H on left:0 R on left:2 H on right:3 R on right:1 ferry is on the right. H on left:2 R on left:2 H on right:1 R on right:1 ferry is on the left. H on left:1 R on left:1 H on right:2 R on right:2 ferry is on the right.	9	10

		H on left:3 R on left:1 H on right:0 R on right:2 ferry is on the left. H on left:0 R on left:1 H on right:3 R on right:2 ferry is on the right. H on left:1 R on left:1 H on right:2 R on right:2 ferry is on the left. H on left:0 R on left:0 H on right:3 R on right:3 ferry is on the right.		
DFS	Farmer, Fox, Chicken, and Grain	left: ['Farmer', 'Fox', 'Chicken', 'Grain'] right: [] side: left left: ['Fox', 'Grain'] right: ['Farmer', 'Chicken'] side: right left: ['Fox', 'Grain', 'Farmer'] right: ['Chicken'] side: left	7	7

		left: ['Grain'] right: ['Chicken', 'Farmer', 'Fox'] side: right left: ['Grain', 'Farmer', 'Chicken'] right: ['Fox'] side: left left: ['Chicken'] right: ['Fox', 'Farmer', 'Grain'] side: right left: ['Chicken', 'Farmer'] right: ['Fox', 'Grain'] side: left left: [] right: ['Fox', 'Grain', 'Farmer', 'Chicken'] side: right		
DFS	4-Disk Towers of Hanoi	[[4, 3, 2, 1] ,[] ,[]] [[4, 3, 2] ,[1] ,[]] [[4, 3] ,[1] ,[2]] [[4, 3, 1] ,[] ,[2]] [[4, 3] ,[] ,[2, 1]] [[4] ,[3] ,[2, 1]] [[4, 1] ,[3] ,[2]] [[4] ,[3, 1] ,[2]] [[4, 2] ,[3, 1] ,[]] [[4, 2, 1] ,[3] ,[]] [[4, 2] ,[3] ,[1]] [[4] ,[3, 2] ,[1]] [[4, 1] ,[3, 2] ,[]] [[4] ,[3, 2, 1] ,[]] [[] ,[3, 2, 1] ,[4]] [[1] ,[3, 2] ,[4]] [[] ,[3, 2] ,[4, 1]] [[2] ,[3] ,[4, 1]] [[2, 1] ,[3] ,[4]]	40	40

		[[2] ,[3, 1] ,[4]] [[] ,[3, 1] ,[4, 2]] [[1] ,[3] ,[4, 2]] [[] ,[3] ,[4, 2, 1]] [[3] ,[] ,[4, 2, 1]] [[3, 1] ,[] ,[4, 2]] [[3] ,[1] ,[4, 2]] [[3, 2] ,[1] ,[4]] [[3, 2, 1] ,[] ,[4]] [[3, 2] ,[] ,[4, 1]] [[3] ,[2] ,[4, 1]] [[3, 1] ,[2] ,[4]] [[3] ,[2, 1] ,[4]] [[] ,[2, 1] ,[4, 3]] [[1] ,[2] ,[4, 3]] [[] ,[2] ,[4, 3, 1]] [[2] ,[] ,[4, 3, 1]] [[2, 1] ,[] ,[4, 3]] [[2] ,[1] ,[4, 3]] [[] ,[1] ,[4, 3, 2]] [[1] ,[] ,[4, 3, 2]] [[] ,[] ,[4, 3, 2, 1]]		
BFS	Humans, Robots and Ferry	H on left:3 R on left:3 H on right:0 R on right:0 ferry is on the left. H on left:2 R on left:2 H on right:1 R on right:1 ferry is on the right. H on left:3 R on left:2 H on right:0 R on right:1 ferry is on the left. H on left:0	7	10

		R on left:2 H on right:3 R on right:1 ferry is on the right. H on left:2 R on left:2 H on right:1 R on right:1 ferry is on the left. H on left:0 R on left:1 H on right:3 R on right:2 ferry is on the right. H on left:1 R on left:1 H on right:2 R on right:2 ferry is on the left. H on left:0 R on left:0 H on right:3 R on right:3 ferry is on the right.		
BFS	Farmer, Fox, Chicken, and Grain	left: ['Farmer', 'Fox', 'Chicken', 'Grain'] right: [] side: left left: ['Fox', 'Grain'] right: ['Farmer', 'Chicken']	7	9

		side: right left: ['Fox', 'Grain', 'Farmer'] right: ['Chicken'] side: left left: ['Grain'] right: ['Chicken', 'Farmer', 'Fox'] side: right left: ['Grain', 'Farmer', 'Chicken'] right: ['Fox'] side: left left: ['Chicken'] right: ['Fox', 'Farmer', 'Grain'] side: right left: ['Chicken', 'Farmer'] right: ['Fox', 'Grain'] side: left left: [] right: ['Fox', 'Grain', 'Farmer', 'Chicken'] side: right		
BFS	4-Disk Towers of Hanoi	[[4, 3, 2, 1] ,[] ,[]] [[4, 3, 2] ,[1] ,[]] [[4, 3] ,[1] ,[2]] [[4, 3] ,[] ,[2, 1]] [[4] ,[3] ,[2, 1]] [[4, 1] ,[3] ,[2]] [[4, 1] ,[3, 2] ,[]] [[4] ,[3, 2, 1] ,[]] [[] ,[3, 2, 1] ,[4]] [[] ,[3, 2] ,[4, 1]] [[2] ,[3] ,[4, 1]] [[2, 1] ,[3] ,[4]] [[2, 1] ,[] ,[4, 3]] [[2] ,[1] ,[4, 3]] [[] ,[1] ,[4, 3, 2]]	15	70

		[[] ,[] ,[4, 3, 2, 1]]		

Why the maximum length of the OPEN list is more for one algorithm than the other for the tower of Hanoi problem:

BFS has 7/4 times the maximum length of the OPEN list compared to DFS. This is because DFS works based on continuing on exploring on one “branch” until it hits a dead end while BFS works by exploring all the branches in similar “pace”(it just explores one level down for every node in the previous level). Hene it requires a larger memory than DFS. Quoting from the lecture slides, BFS’s “OPEN list will require storage O(b^d), where d is the depth of the closest goal node and b is the tree’s branching factor”.

Why the solution PATH length is different for one algorithm from that of the other for the tower of Hanoi problem:

BFS’s length of Path is 15 compared to DFS’s length of Path of 40. This is because BFS will have an easier time backtracking the optimal, shortest path when it discovers it. DFS on the other hand can arrive to the goal node using a less optimal path, and may miss the shortest path at point in the search