Chapter 3. Search problems.

1) a)

With loop detection:

Iteration	Frontier at the end of this iteration
0	[a]
1	[a,b], [a,e]
2	[a,b,d], [a,b,c], [a,e]
3	[a,b,d,c], [a,b,c], [a,e]
4	[a,b,c], [a,e]
5	[a,b,c,d], [a,e]
6	[a,e]
7	[a,e,f]

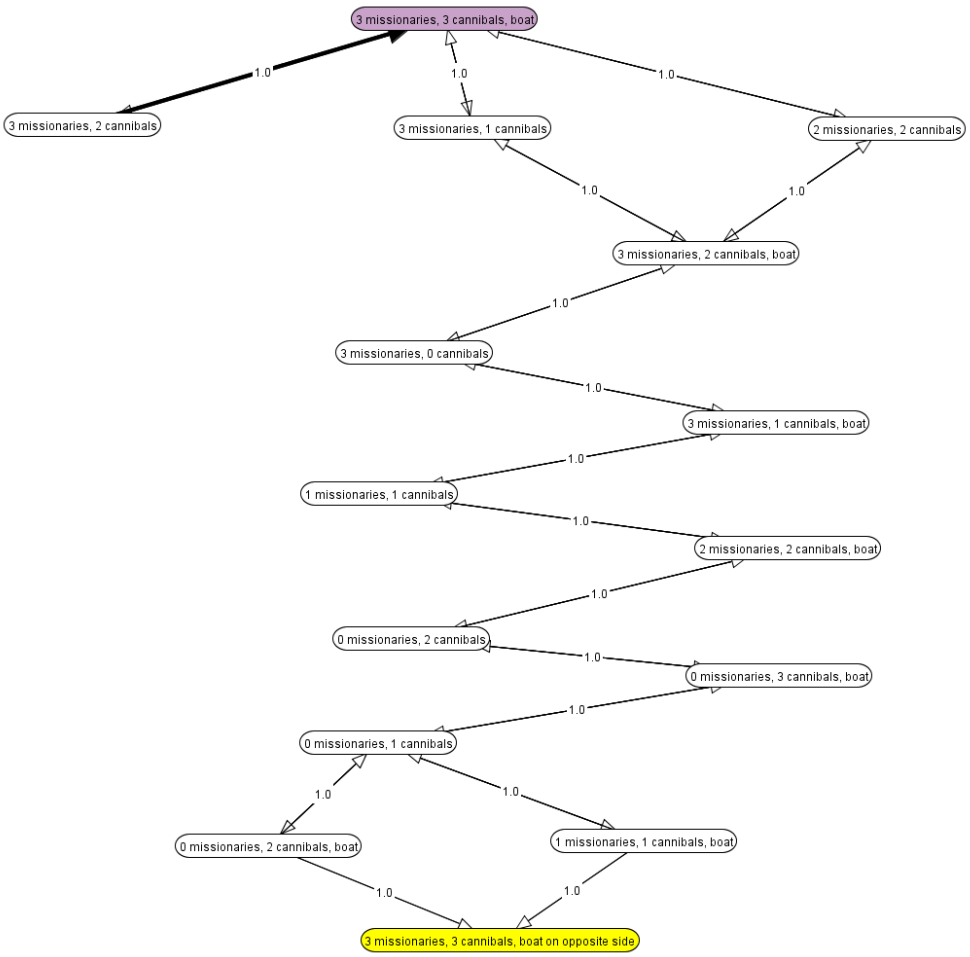
Without loop detection:

Iteration	Frontier at the end of this iteration
0	[a]
1	[a,b], [a,e]
2	[a,b,d], [a,b,c], [a,e]
3	[a,b,d,b], [a,b,d,c], [a,b,c], [a,e]
4	[a,b,d,b,d], [a,b,d,b,c], [a,b,d,c], [a,b,c], [a,e]
5	[a,b,d,b,d,b], [a,b,d,b,d,c], [a,b,d,b,c], [a,b,d,c], [a,b,c], [a,e]
n	…. Infinite loop

b)

Iteration	Frontier at the end of this iteration
0	[a]
1	[a,b], [a,e]
2	[a,b,d], [a,b,c], [a,e,f]

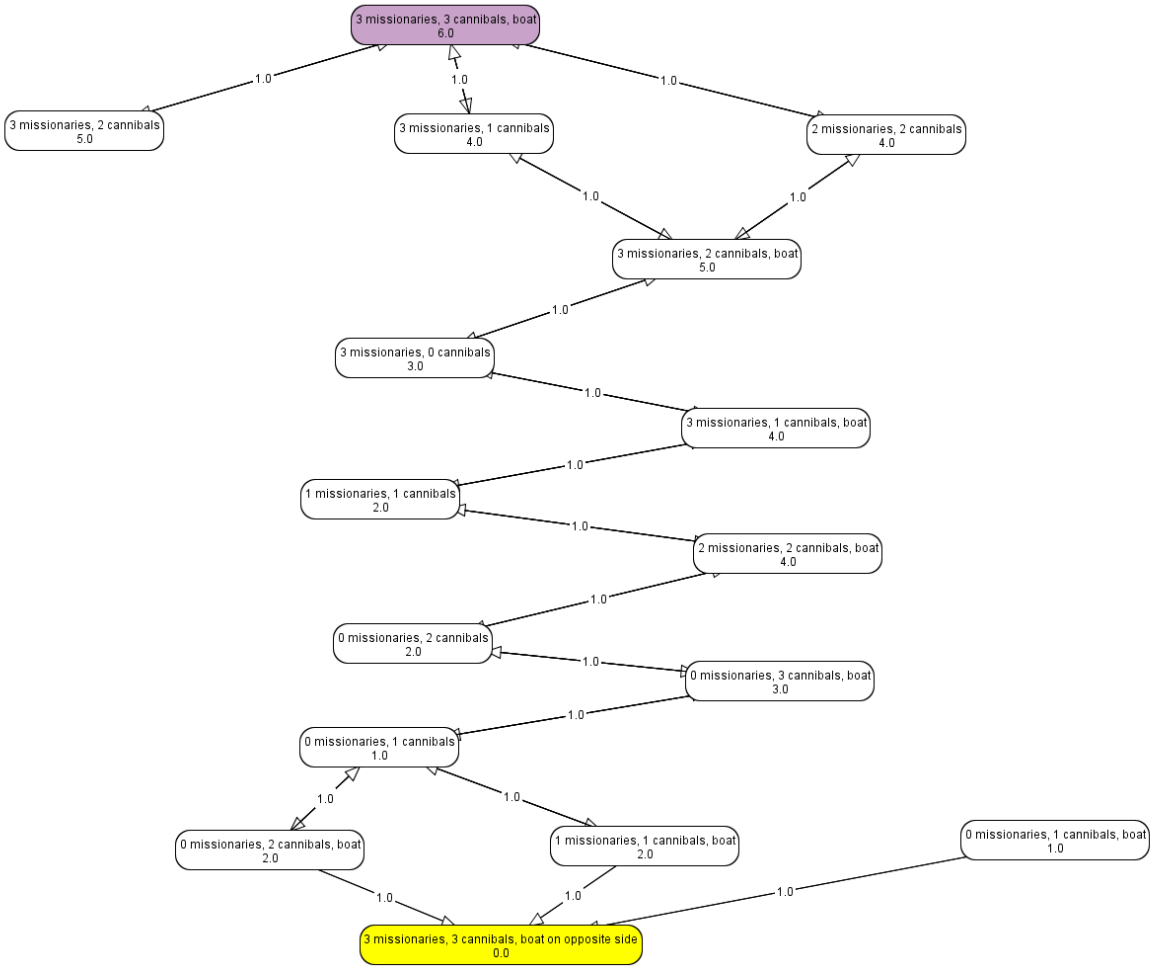
2) ii)



iii) Using Depth-First Search with loop detection, 13 nodes were expanded.

iv) Using Breadth-First Search with loop detection, 27 nodes were expanded.

v)



vi) Using A* Search with loop detection, 24 nodes were expanded.

3) **i)** The following chart assumes that we are indicating whether or not the search algorithm is guaranteed to give an optimal path in the given maze:

Search Algorithm	Abstract data type	Optimal
Depth First	Stack	Yes, but only with the assumption that loop detection is in place, otherwise no
Breadth First	Queue	Yes
Best First	Priority Queue	Yes, but only with the assumption that loop detection is in place, otherwise no
A*	Priority Queue	Yes

ii) Assuming that some implementation of loop detection is in place, all search algorithms will provide an optimal path because there only exists one path from the start to exit.

If loop detection were not in place, then both Depth First and Best First would have the following path:

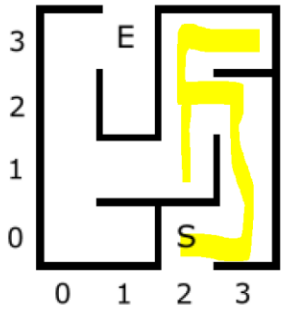


Figure 2

Using Depth First and Best First search algorithms without loop detection would cause an infinite loop and therefore a path out would never be found.

Chapter 4. Local Search in Continuous Spaces

1) a) $l`(p) = -\frac{14}{p} + \frac{6}{1-p} = -\frac{-20p+14}{p(1-p)}$

b)

Step	p	Step size
0	½ = 0.5000	0.02
1	0.8200	0.01
2	0.6574	0.005
3	0.6763	0.011
4	0.7001	0.011
5	0.6999 ≈ 0.700	

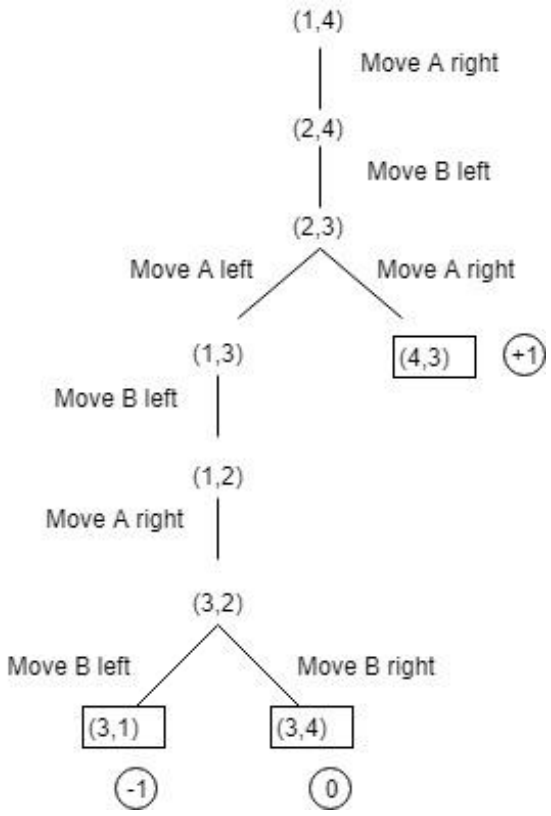
2) a) $l``(p) = -\frac{-20p^2+28p-14}{p^2(1-p)^2}$

b)

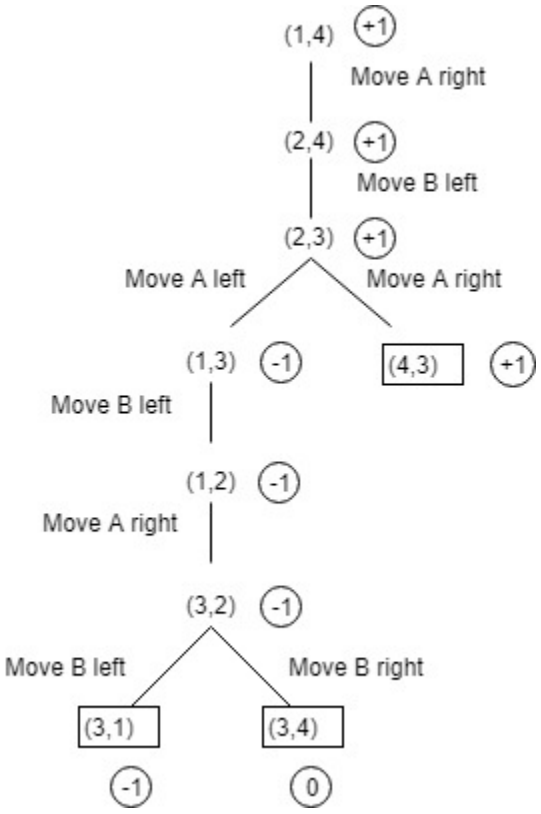
Step	p
0	½ = 0.50000
1	0.20625
2	0.17874
3	0.15924
4	0.14438
5	0.13254

Chapter 5. Adversarial Search

1. **a) b)**



2.



3. **a)** The optimal strategy for player A would be to follow the minimax values that maximizes their utility, which in this case would be to follow the values +1.

The optimal strategy for player B is the opposite of player A; for player B, the optimal strategy would be follow the minimax values that maximize his utility, which would be +1 in the point of view of player B. However, in the tree above, we look at maximizing the outcome in the point of view of player A, therefore player B's optimal strategy would be to follow the minimax values -1.

b) If we allowed repeated states, then the standard minimax algorithm would not always terminate with a strategy for each player because of how the minimax algorithm is implemented. As mentioned in the course textbook, the "minimax algorithm performs a complete depth-first exploration of the game tree". Therefore, similar to depth first search, if we allowed repeated states, then there would be infinite loops. To solve this issue, we could implement a loop detection mechanism by keeping track of visited states using an array for example and prevent reoccurring states.