Chapter 3

1. a)

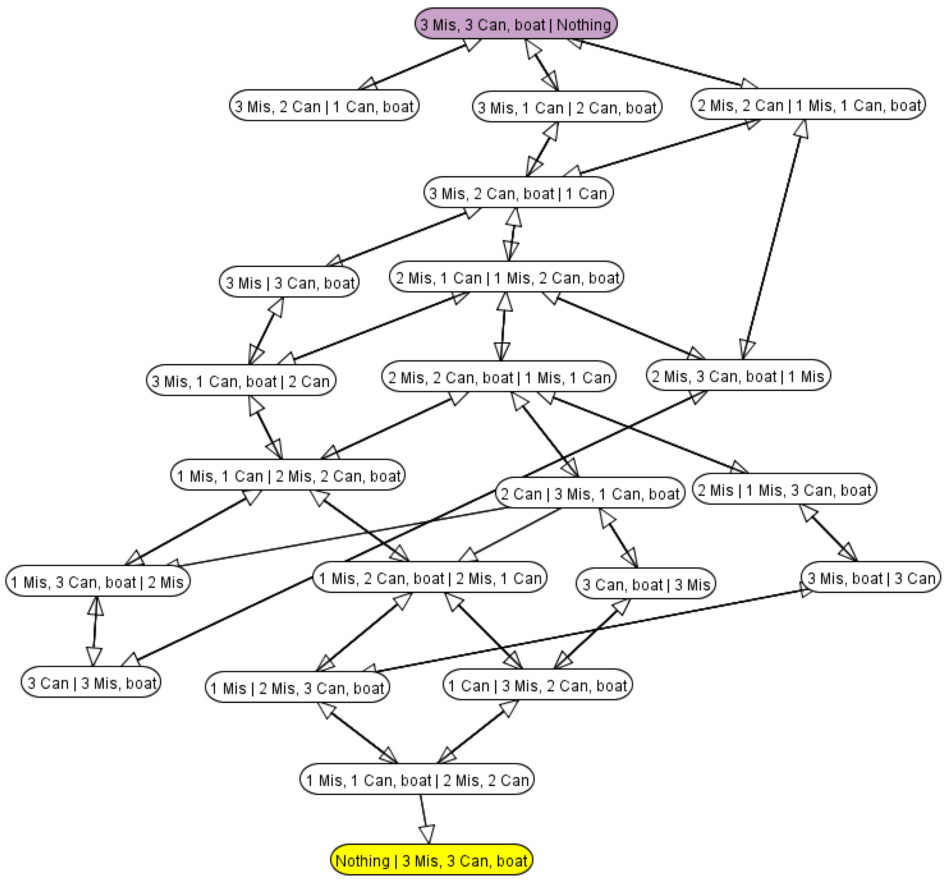
Iteration	Frontier at the end of this iteration
0	[a]
1	[a, b], [a, e]
2	[a, e], [a, b, c], [a, b, d]
3	[a, e], [a, b, c, d], [a, b, d, c]
4	[a, e], [a, b, c, d, b], [a, b, d, c, b]
5	[a, e], [a, b, c, d, b, c], [a, b, d, c, b, d]
...	...
n	[a, e], [a, b, c, d, b, ...], [a, b, d, c, b, ...]

We would never reach the goal state, because we have a loop in the graph and we will continue to expand the longest paths, which will never terminate.

1. b)

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

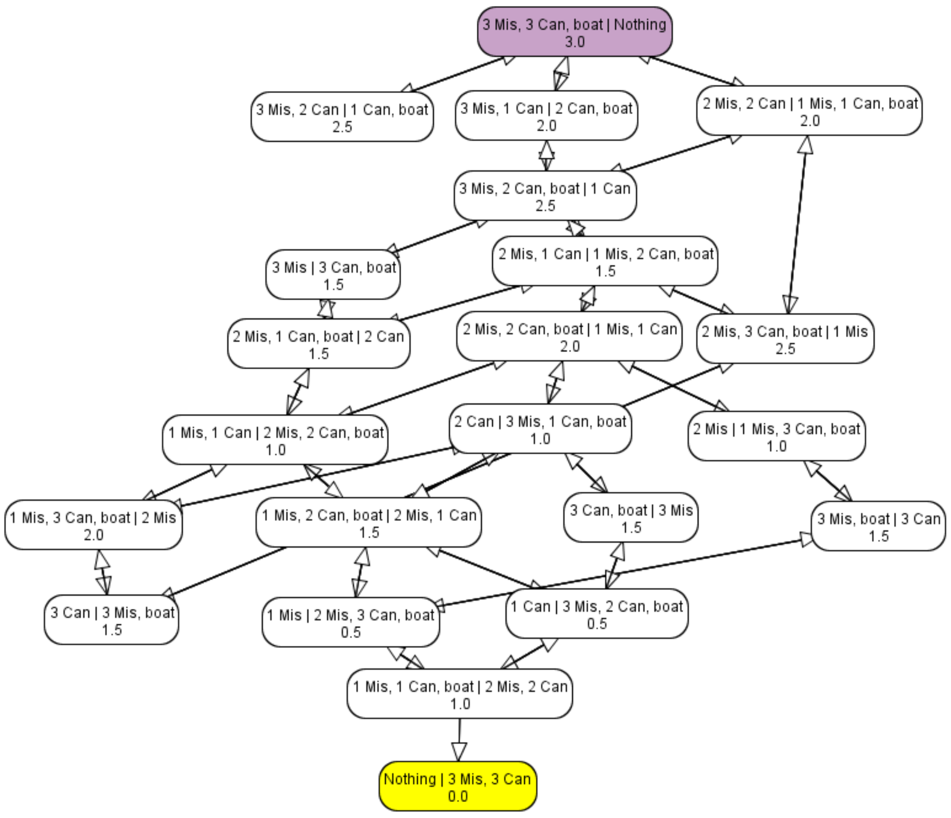
2. ii)


3 Mis, 3 Can, boat | Nothing
3 Mis, 2 Can | 1 Can, boat
3 Mis, 1 Can | 2 Can, boat
2 Mis, 2 Can | 1 Mis, 1 Can, boat
3 Mis, 2 Can, boat | 1 Can
3 Mis | 3 Can, boat
2 Mis, 1 Can | 1 Mis, 2 Can, boat
3 Mis, 1 Can, boat | 2 Can
2 Mis, 2 Can, boat | 1 Mis, 1 Can
2 Mis, 3 Can, boat | 1 Mis
1 Mis, 1 Can | 2 Mis, 2 Can, boat
2 Can | 3 Mis, 1 Can, boat
2 Mis | 1 Mis, 3 Can, boat
1 Mis, 3 Can, boat | 2 Mis
1 Mis, 2 Can, boat | 2 Mis, 1 Can
3 Can, boat | 3 Mis
3 Mis, boat | 3 Can
3 Can | 3 Mis, boat
1 Mis | 2 Mis, 3 Can, boat
1 Can | 3 Mis, 2 Can, boat
1 Mis, 1 Can, boat | 2 Mis, 2 Can
Nothing | 3 Mis, 3 Can, boat

2. iii) We expanded 17 nodes to reach the goal with a path cost of 15 (not the solution with the least crossings).

2. iv) The breadth first search expanded 443 nodes, reaching the goal with a path cost of 9 (an optimal path)

2. v)

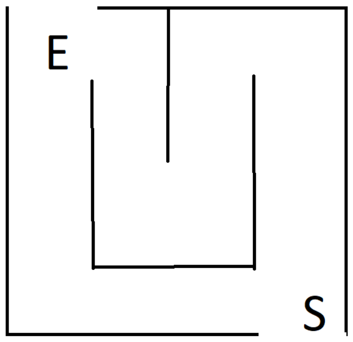

3 Mis, 3 Can, boat | Nothing
3.0
3 Mis, 2 Can | 1 Can, boat
2.5
3 Mis, 1 Can | 2 Can, boat
2.0
2 Mis, 2 Can | 1 Mis, 1 Can, boat
2.0
3 Mis, 2 Can, boat | 1 Can
2.5
3 Mis | 3 Can, boat
1.5
2 Mis, 1 Can | 1 Mis, 2 Can, boat
1.5
2 Mis, 1 Can, boat | 2 Can
1.5
2 Mis, 2 Can, boat | 1 Mis, 1 Can
2.0
2 Mis, 3 Can, boat | 1 Mis
2.5
1 Mis, 1 Can | 2 Mis, 2 Can, boat
1.0
2 Can | 3 Mis, 1 Can, boat
1.0
2 Mis | 1 Mis, 3 Can, boat
1.0
1 Mis, 3 Can, boat | 2 Mis
2.0
1 Mis, 2 Can, boat | 2 Mis, 1 Can
1.5
3 Can, boat | 3 Mis
1.5
3 Mis, boat | 3 Can
1.5
3 Can | 3 Mis, boat
1.5
1 Mis | 2 Mis, 3 Can, boat
0.5
1 Can | 3 Mis, 2 Can, boat
0.5
1 Mis, 1 Can, boat | 2 Mis, 2 Can
1.0
Nothing | 3 Mis, 3 Can
0.0

2. vi) Using A* search, we end up expanding 256 nodes, reaching the goal with a path cost of 9

3. i)

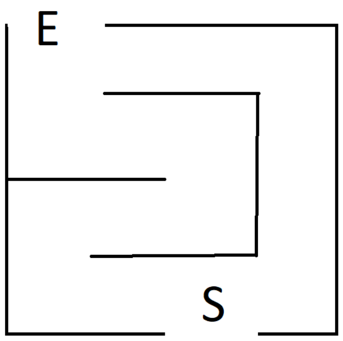
Search Algorithm	Abstract Data Type	Optimal?
Depth First	Stack	No
Breadth First	Queue	Yes
Best First	Priority Queue	No
A*	Priority Queue	Yes

ii) Depth first



This will fail if we expand the wrong path first, in this case the path that starts by going up. If we expand this path first, we will arrive at the goal state, but by taking a path that is longer than needed.

Best First



This will fail because the function wants to go up and to the left, but since it doesn't take the current path cost into account it won't back track once it starts on the longer path

Chapter 4

1.a) The gradient of l(p) is 6/(1-p) - 14/p

1.b)

step	p	Step size
0	0.5	0.02
1	0.82	0.01
2	0.6574	0.005
3	0.6763	0.0025
4	0.6817	0.00125
5	0.6838	0.000625

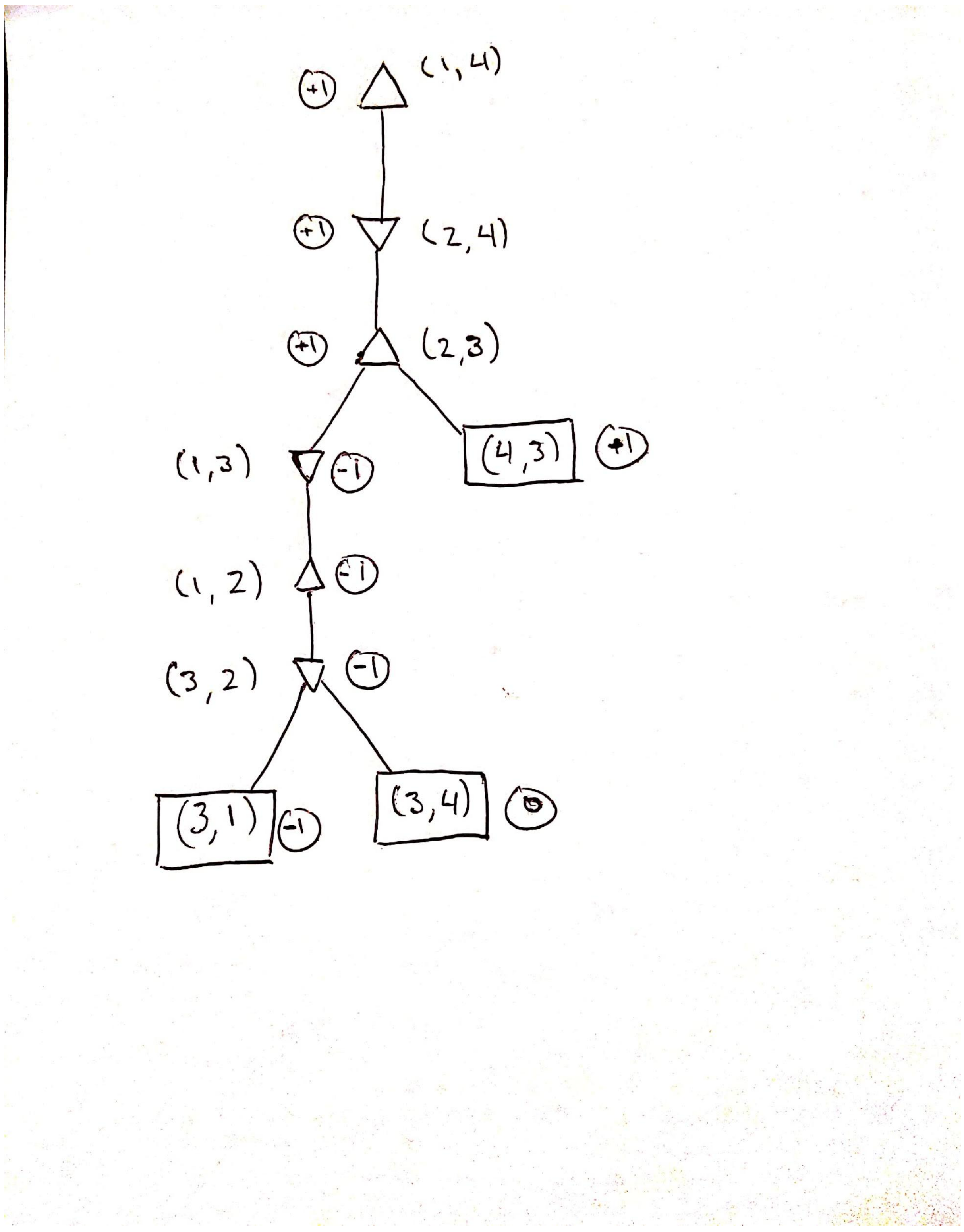
2. a) the second order gradient of l(p) is $14/p^2 - 6/(1-p)^2$

2.b)

step	p
0	0.5
1	0.7
2	0.7
3	0.7
4	0.7
5	0.7

Chapter 5

1/2.



3.a) With repeated moves, the optimal strategy would remain the same. There is still no way to stop player A from winning by simply pushing forward, unless we allow players to skip their turns.
3.b) The minimax algorithm would not terminate because it would continue to iterate down the path where player A does not just push forward and win immediately. To stop this, we

would have to put a certain depth limit to it, so that after reaching a certain depth it would mark those states as terminal states and give them a general value score.