Connect K Coursework

Academic responsible:	Dr Francesco Belardinelli
Department:	Department of Computing
Module:	Computer Vision
Academic year:	2021/2022
Student:	Mr. Maksym Tymchenko
CID:	01063479
Personal tutor:	Dr. Marek Rei
Date:	2/12/2021

Department of Computing
South Kensington Campus
Imperial College London
London SW7 2AZ
U.K.

1 Time complexity without α-β Pruning

Table 1: Execution time without pruning for k = 3.

columns(m), rows(n)	Execution time (seconds)
$m = 3, n = 3$	0.166
$m = 3, n = 4$	1.3519
$m = 4, n = 3$	12.241
$m = 4, n = 4$	311.178

Table 2: Execution time without pruning for k = 4.

columns(m), rows(n)	Execution time (seconds)
$m = 3, n = 3$	0.2827
$m = 3, n = 4$	5.4178
$m = 4, n = 3$	50.872
$m = 4, n = 4$	over 3600

Table 1 shows the execution time of a connect-k game from an empty board state to the end of the game of size (m, n) with $k = 3$. Table 2 shows the execution time of a connect-k game from an empty board state to the end of the game of size (m, n) with $k = 4$. The expected time complexity should be $O(b^d)$ where b is the branching factor (number of legal moves) and d is the maximum depth of the tree. The branching factor of connect-k game starts at the number of columns ($b = m = num\_cols$) for an empty board. Since, the first states of the game take the longest time to perform a mini-max search, the dominating branching factor of the whole game is expected to be the number of columns m. Therefore, the expected time complexity would be $O(m^d)$ where m is the number of columns and d is the maximum depth of the tree.

This means that if the number of columns m increases, the time complexity should increase exponentially. This is confirmed by the data where if we increase m from 3 to 4, the execution time increases by a factor of (73 = 12.241/0.166) for n=3 and a factor of (230 = 311.178/1.35) for n=4.

The difference for different ns is because n influences the maximum depth of the tree. This is because to reach a terminal state for n = 4 the tree will need to be searched deeper than for n = 3 since there will be an extra row at the top of the board.

The difference for different ks is because k influences the maximum depth of the tree. This is because to reach a terminal state for k = 4 the tree will need to be searched deeper than for k = 3 since to find 4 Xs in a row will take a deeper search than to find 3 Xs in a row.

The expected space complexity is $O(bd)$ where b is the branching factor and d is the maximum depth (same as a depth first search).

The number of visited states is expected to be proportional to the maximum depth of the tree. In fact, in the alternative implementation of the search with a defined maximum depth, the time complexity increased by a factor proportional to the number of columns if the maximum depth d was increased by 1.

2 Time complexity with α-β Pruning

Table 3: Execution time with pruning for k = 3.

columns(m), rows(n)	Execution time (seconds)
$m = 3, n = 3$	0.054
$m = 3, n = 4$	0.164
$m = 4, n = 3$	0.292
$m = 4, n = 4$	1.841

Table 4: Execution time with pruning for k = 4.

columns(m), rows(n)	Execution time (seconds)
$m = 3, n = 3$	0.035
$m = 3, n = 4$	0.138
$m = 4, n = 3$	0.645
$m = 4, n = 4$	10.045

Table 3 shows the execution time of a connect-k game from an empty board state to the end of the game of size (m, n) with $k = 3$ using pruning. Table 4 shows the execution time of a connect-k game from an empty board state to the end of the game of size (m, n) with $k = 4$ using pruning.

Compared to the execution time without pruning the time complexity is expected to improve. This is because the effective branching factor is expected to decrease from b to b/2, making the complexity equal to $O((m/2)^d)$ where m is the number of columns. An exponential improvement in performance is indeed seen compared to the search time without pruning.

This means that if the number of columns m increases, the time complexity should increase exponentially. This is confirmed by the data where if we increase m from 3 to 4, the execution time increases by a factor of (5.4 = 0.292/0.054) for n=3 and a factor of (11.22 = 1.841/0.164) for n=4.

The difference for different ns is because n influences the maximum depth of the tree. This is because to reach a terminal state for n = 4 the tree will need to be searched deeper than for n = 3 since there will be an extra row at the top of the board.

The difference for different ks is because k influences the maximum depth of the tree. This is because to reach a terminal state for k = 4 the tree will need to be searched deeper than for k = 3 since to find 4 Xs in a row will take a deeper search than to find 3 Xs in a row.