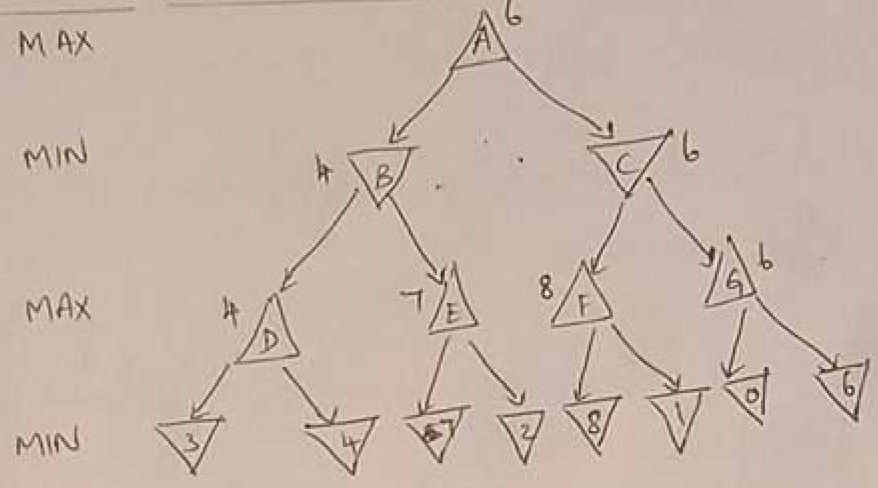
CS 540: INTRODUCTION TO ARTIFICIAL INTELLIGENCE

HOMEWORK ASSIGNMENT #5

QUESTION 1: GAME TREE SEARCH

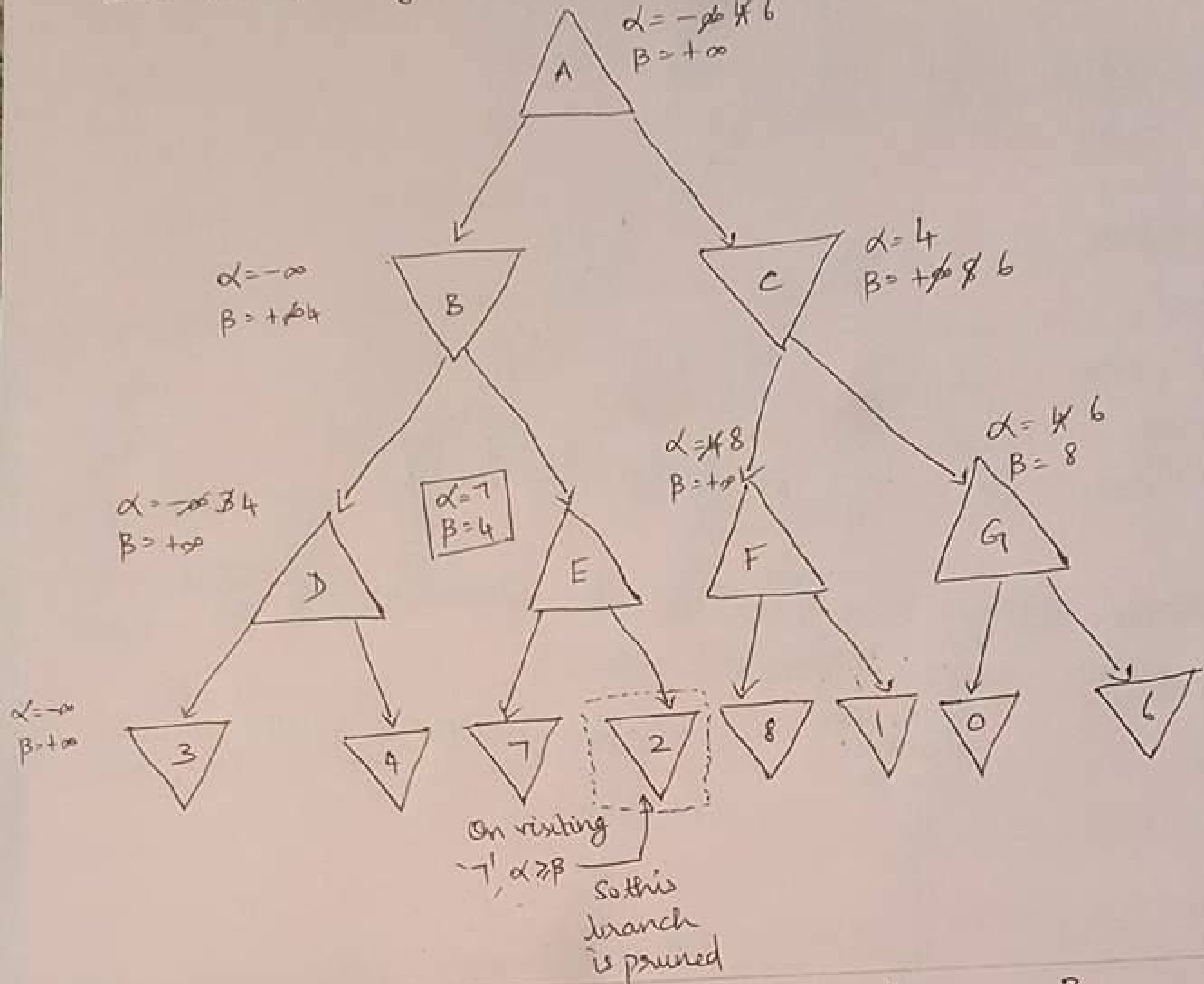
1)



Through optimal play of the Minimax algorithm, the game theoretic values of nodes A... G are

NODE	GAME THEORETIC VALUE
A	6
B	4
C	6
D	4
E	7
F	8
G	6

2) Using alpha-beta pruning to compute the minimax value at each node, we get



NODE	GAME THEORETIC VALUE	α	β
A	6	+6	$+\infty$
B	4	$-\infty$	+4
C	6	+4	+6
D	4	+4	$+\infty$
E	7	+7	+4
F	8	+8	$+\infty$
G	6	+6	+8

3) From the game tree in answer 2), we can see that On visiting node ▽7, the value of α at node △A is 7 and the value of β at node △E is 4. At this point, $\alpha \geq \beta$ and hence all the other subtrees are pruned. i.e. in this case the node ▽2 is pruned.

---

4) In the case of Minimax algorithm, we need to explore all the states in the state space to reach the optimal game play, which is very expensive in terms of memory usage. On the other hand, alpha-beta pruning helps to ignore (or) completely prune subtrees whenever they are found to have no impact on the optimal game play.

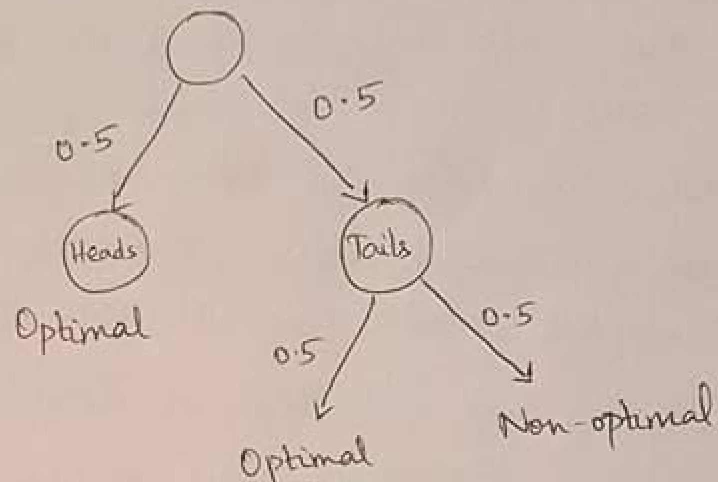
In the game tree described in answer (2), after visiting node △D, node ▽B knows that it can reach a minimum value of 4. on visiting node △E, the max player would choose the value of 7 on visiting the node ▽7. At this point node △B knows that the value of 4 obtained from △D is better than the current value of E i.e. 7. So, ▽B loses interest in △E and hence the node ▽2 is pruned. We still follow the optimal game play without having to visit each and every node of the game tree.

5) In the specified non-deterministic environment, a player can choose an optimal action in two different ways

1) when a HEAD turns up in the first flip of a fair coin

(or)

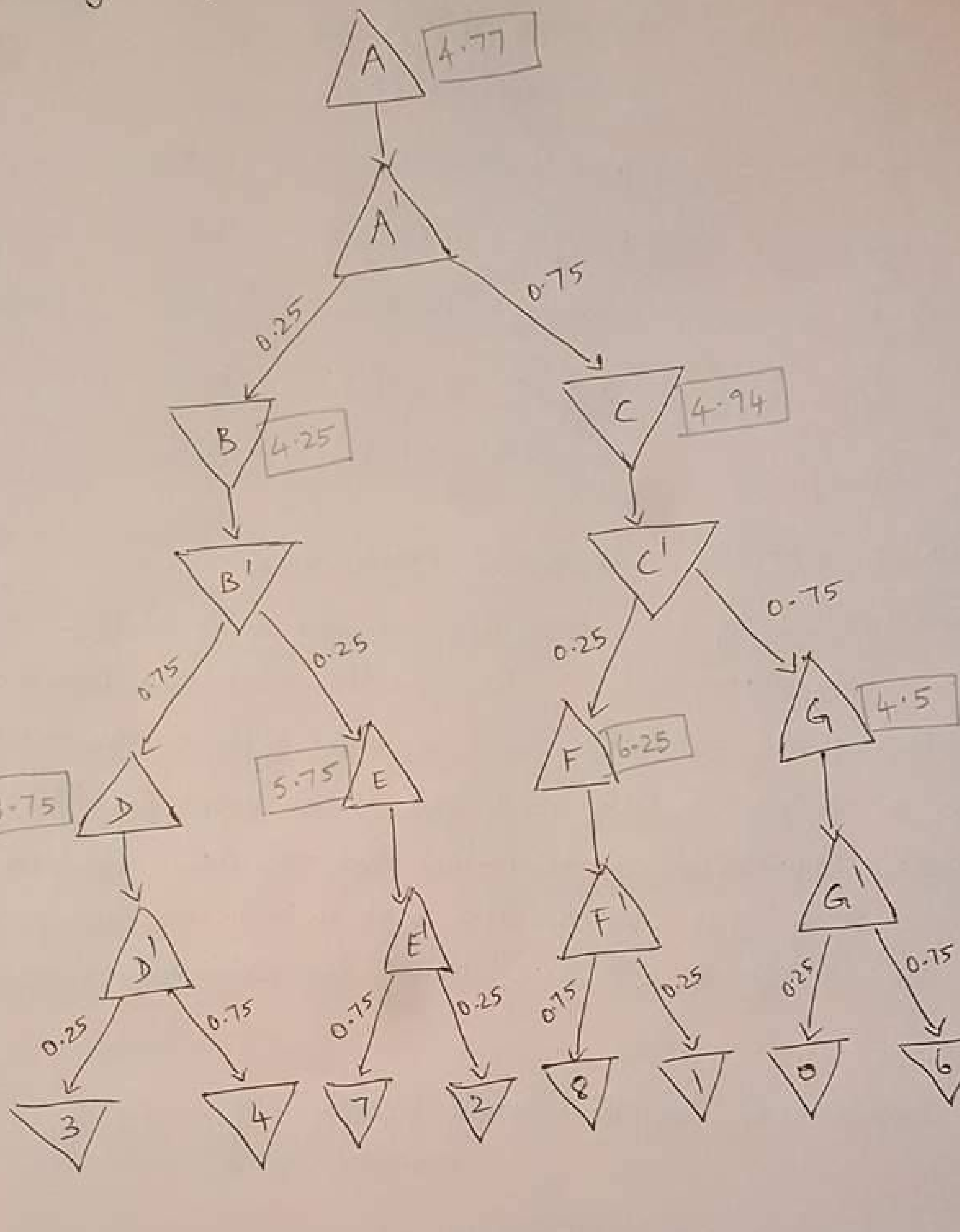
2) when a TAIL turns up in the first flip of the fair coin and the random choice in the second flip turns to be optimal.



Total probability of a player choosing an optimal path at each node $= 0.5 + (0.5 \times 0.5)$

$= \underline{0.75}$

b) Redrawing the game tree with chance nodes as specified, we get



7)

Node	Expected game theoretic value at the node	
A	0.25(4.25) + 0.75(4.94)	= 4.77
B	0.75(3.75) + 0.25(5.75)	= 4.25
C	0.25(6.25) + 0.75(4.5)	= 4.94
D	0.25(3) + 0.75(4)	= 3.75
E	0.75(7) + 0.25(2)	= 5.75
F	0.75(8) + 0.25(1)	= 6.25
G	0.25(0) + 0.75(6)	= 4.5

QUESTION 2: NATURAL LANGUAGE PROCESSING

1) One simple way for a computer program to break the corpus into "computer words" is to split the sentences based on whitespaces as delimiter. So, I read the files in the corpus into a string in Ruby and split them based on '\s' regular expression as delimiter and calculate word tokens and word types. (WHITESPACE → \t, \n, \r). Also, force encoded all the strings to UTF-8 encoding for uniformity (& simplicity)

2) Number of computer word tokens (occurrences) $\}$ = 213494

Number of computer word types (Distinct words) $\}$ = 17719

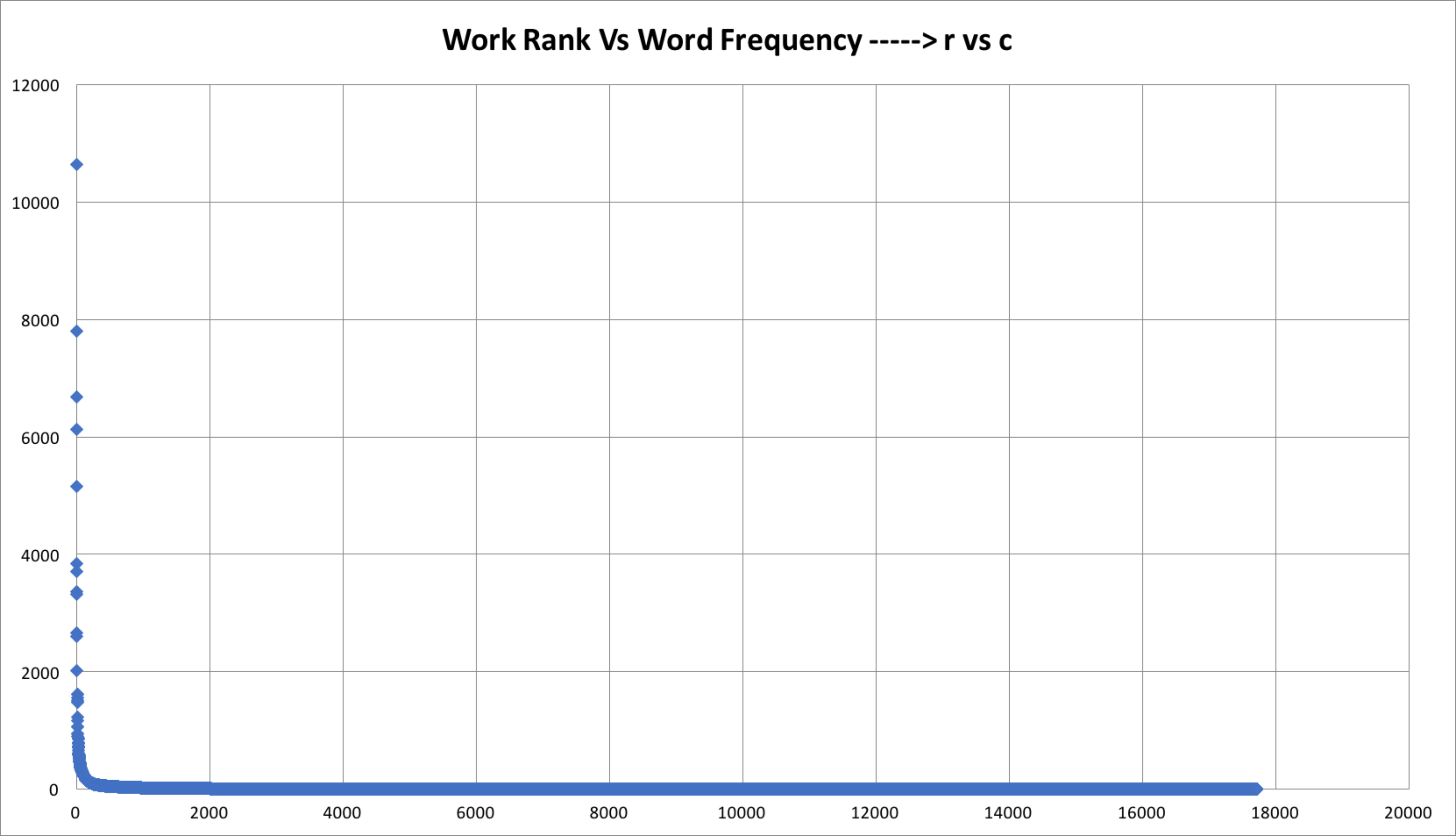
3. The top 20 word types by their counts are

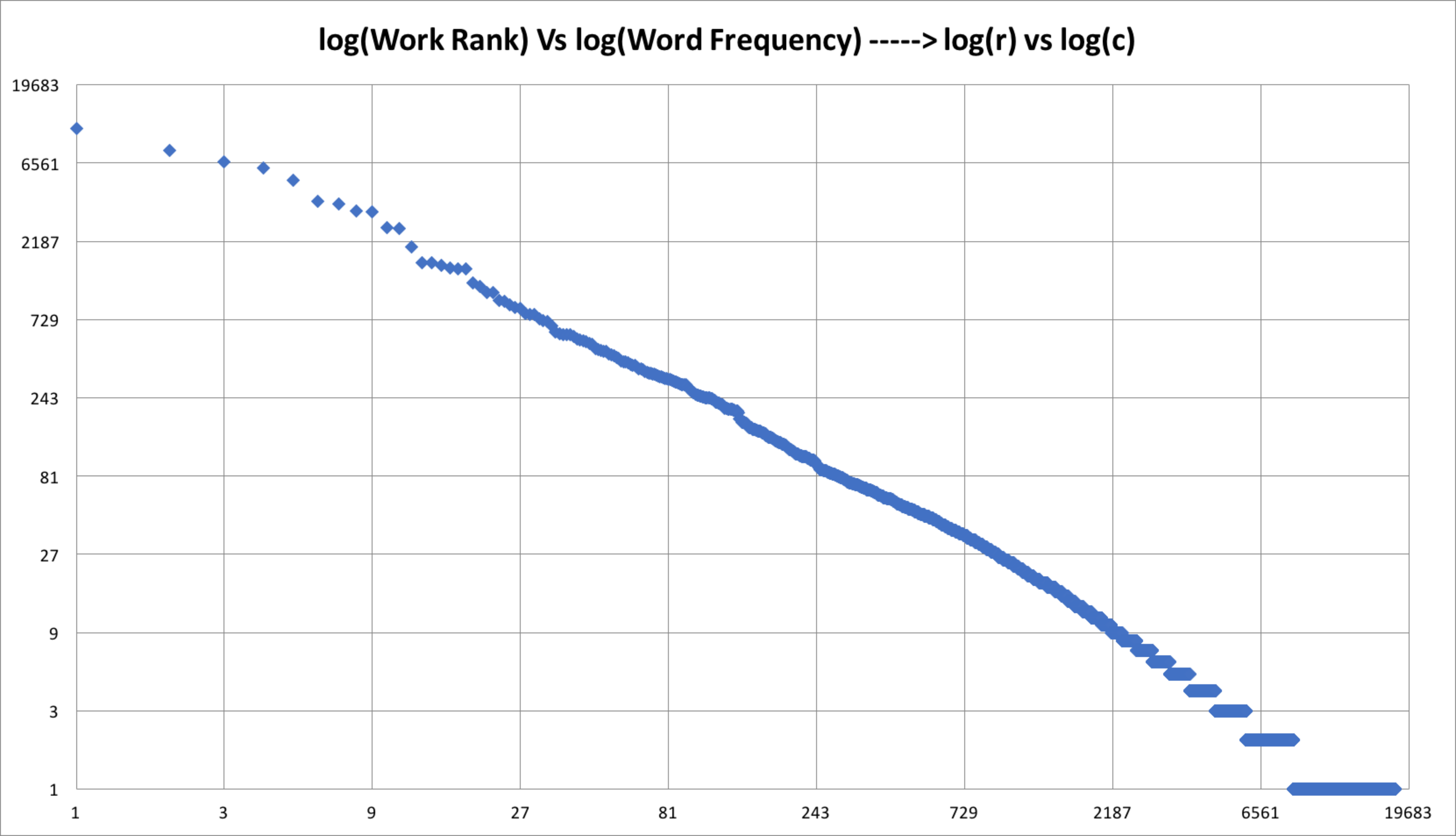
1) the - 10647
2) - 7803
3) to - 6687
4) of - 6128
5) and - 5158
6) in - 3850
7) a - 3705
8) that - 3360
9) is - 3313
10) be - 2663
11) AI - 2610
12) will - 2024
13) are - 1616
14) for - 1615
15) it - 1557
16) not - 1506
17) as - 1486
18) on - 1483
19) with - 1222
20) The - 1162

---

4. The bottom 20 word types all with count 1 are,

1) undetectable,
2) imagination,
3) line,
4) service?
5) Primarily,
6) employment/
7) wages.
8) sophistications
9) hindering
10) note,
11) Unless,
12) convinced
13) mentally
14) physically,
15) bleak
16) unwarranted.
17) travelled.
18) Automating
19) unstated
20) above.


Work Rank Vs Word Frequency -----> r vs c
12000
10000
8000
6000
4000
2000
0
0
2000
4000
6000
8000
10000
12000
14000
16000
18000
20000



log(Work Rank) Vs log(Word Frequency) -----> log(r) vs log(c)
19683
6561
2187
729
243
81
27
9
3
1
1
3
9
27
81
243
729
2187
6561
19683

7) The rank vs count (r, c) graph shows the following

1) As the rank of word type increases, the count of the word type falls sharply initially. (overall, the count falls exponentially)

2) Words with rank in the starting range say (1 to 20) have very high word counts. Most word types have very small counts.

3) As the rank of the word type increases, the word frequency almost becomes flat towards the later end of the graph.

The logarithmic $(\log r, \log c)$ graph shows ~~a~~ the following

1) The graph almost follows a steady slope.

2) There is an almost linear relation between $\log r$ and $\log c$. (line equation) → ZIPF'S LAW

3) As $\log r$ increases, $\log c$ falls proportionally.

---

8) TWO POTENTIAL ISSUES WITH MY COMPUTER WORDS

1) CASE SENSITIVITY: The tokenizer cannot ~~distinguish~~ relate between words of different cases. i.e. 'the' and 'The' are recognized as two separate words. Thus, when used for NLP, this tokenizer cannot perform a case insensitive matching.

2) PUNCTUATION: Stripping off of punctuation is not performed by the tokenizer and hence the same word prefixed (or) suffixed with different punctuation marks are treated as separate word tokens (i.e. distinct).

eg- intelligent.	Service
Intelligent,	service?
intelligent-	service.

3) WORD SEPARATOR / DELIMITER: Though whitespaces are ideal to separate word tokens, there are special cases where word tokens are to be grouped and treated as a compound word rather than being split using whitespaces.

eg. co operation split as 'co' and 'operation'

4) NUMERIC CHARACTERS: No processing is done on numbers and hence each number occurring on the corpus gets added as a new word token. Also, the correlation numeric and their textual representation is not present.

i.e. '100' and 'hundred' are two separate word tokens.