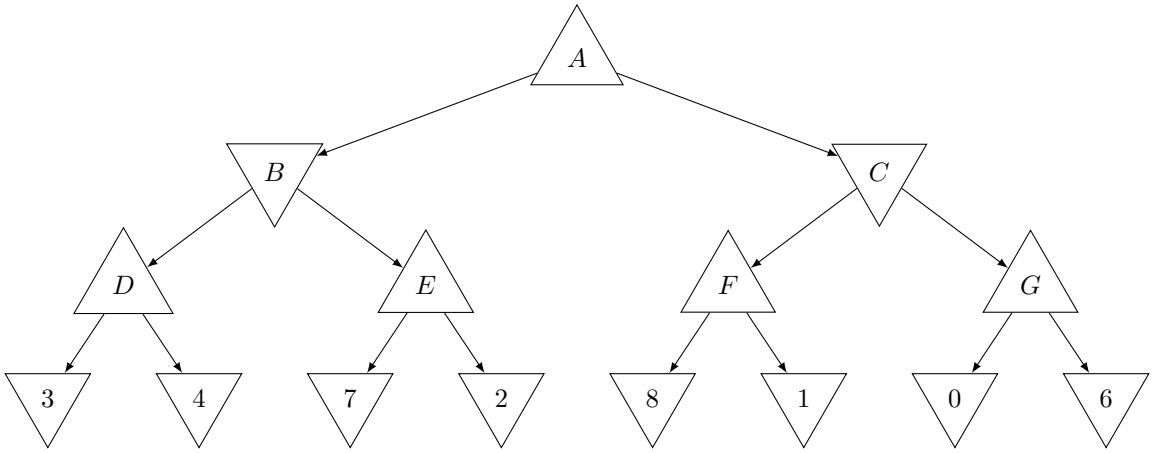
CS 540: Introduction to Artificial Intelligence
Homework Assignment # 5

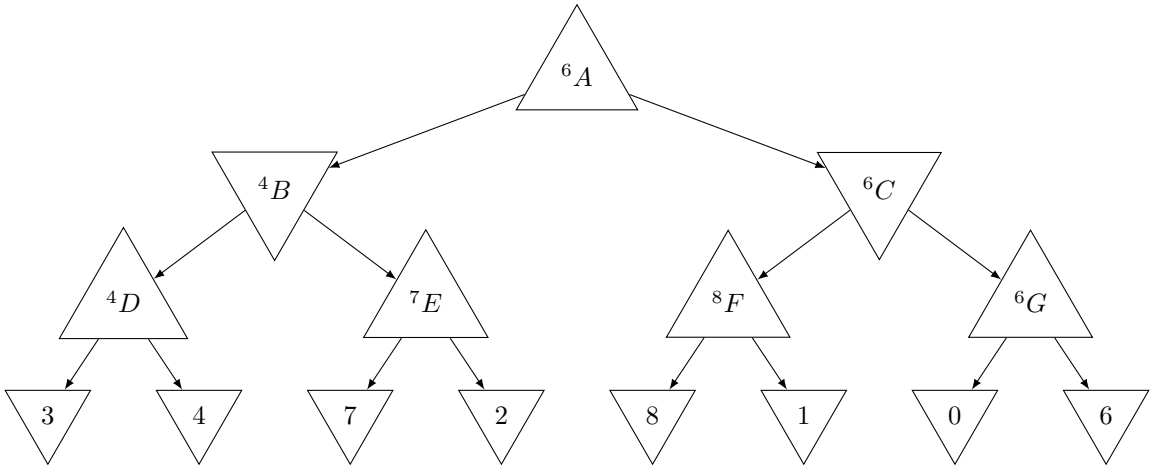
Yien Xu

Question 1: Game Tree Search [60 points]

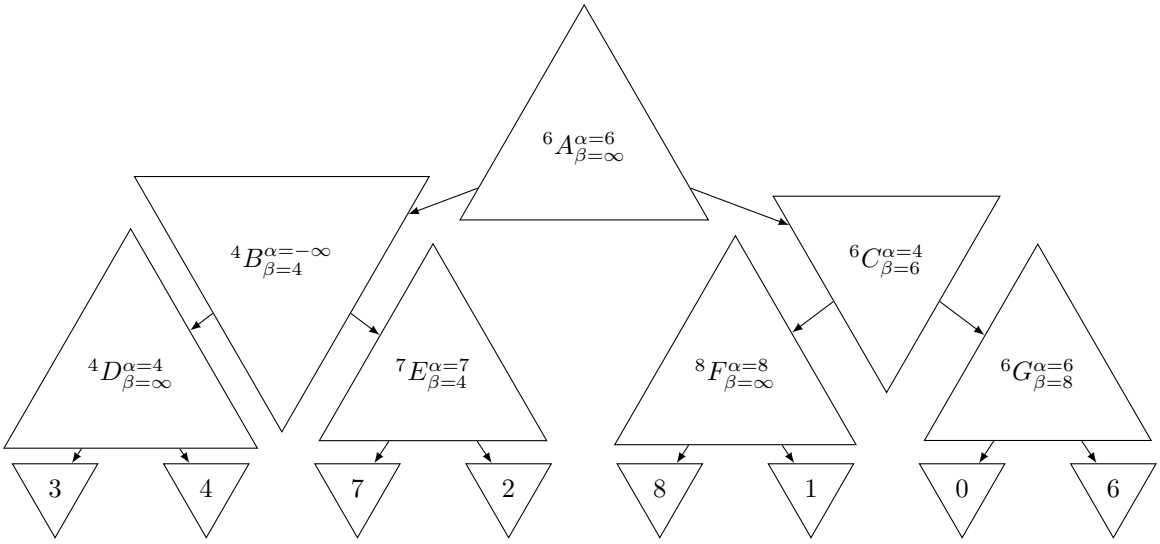
Consider the game tree below. Let Δ and ∇ nodes represent nodes belonging to the maximizing and minimizing player respectively.



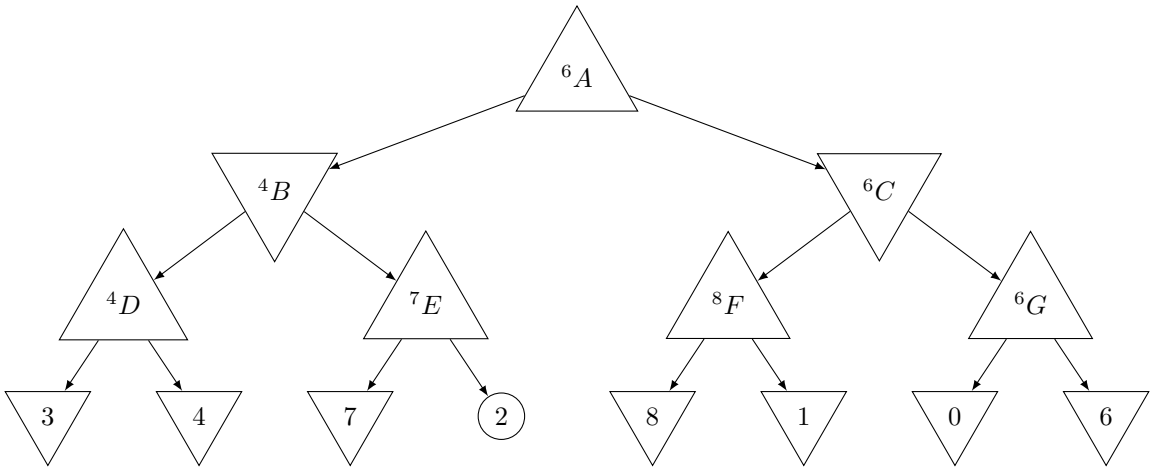
1. (20 points) Suppose we wish to run the **Minimax** algorithm on this game tree. Provide the game theoretic values of nodes $A \ldots G$ through optimal play.



2. (10 points) Use alpha-beta pruning to compute the Minimax value at each node for the original game tree (i.e., deterministic environment). Assume that nodes are visited from left to right. In addition to the Minimax values, show your alpha and beta values at each node after the final time it is visited.



3. (5 points) Which branches, if any, in the game tree are pruned? Show this on the graph above.



The node with circle border (and its subtree) is pruned.

4. (5 points) Explain, in English, why we would want to use alpha-beta pruning.

In some cases, it is unnecessary to search all the game tree to obtain the game theoretic value. Take the previous problem as an exmaple, node E must have chosen the largest of his children. The node with value 7 already exceeds that of node D. So node B must have chosen D if that were the case.

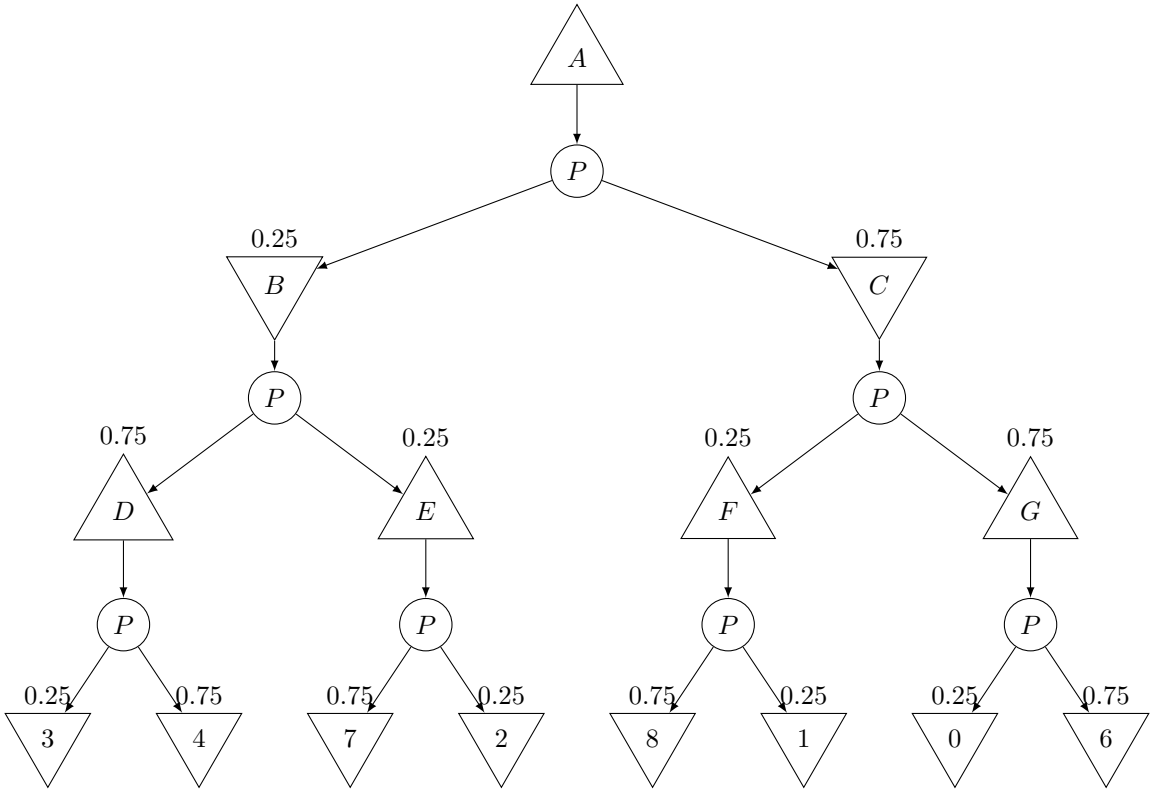
Therefore, the search on the right children of E is not necessary. That's why alpha-beta pruning can help avoid extra work on searching.

5. (5 points) Now suppose we are in a non-deterministic environment with rules as follows. When a player begins their turn they first flip a fair coin (heads probability .5) and if it turns up heads they choose the optimal action. However, if the coin turns up tails they then flip another fair coin to randomly select their action. In this game, what is the probability of a player choosing an optimal action *at each node*?

   Let X_i = Choosing the optimal action at Node i

$$P(X_A) = 0.5 + 0.5 \times 0.5 = 0.75$$
$$P(X_B) = 0.5 + 0.5 \times 0.5 = 0.75$$
$$P(X_C) = 0.5 + 0.5 \times 0.5 = 0.75$$
$$P(X_D) = 0.5 + 0.5 \times 0.5 = 0.75$$
$$P(X_E) = 0.5 + 0.5 \times 0.5 = 0.75$$
$$P(X_F) = 0.5 + 0.5 \times 0.5 = 0.75$$
$$P(X_G) = 0.5 + 0.5 \times 0.5 = 0.75$$

6. (5 points) Given the game described in part 5, redraw the game tree as follows. For each non-terminal node X, introduce a new chance node X'. Let X' be the only successor of X, while the original successors of X are now given to X'. Label the probability on each edge from X' to its successors. These probabilities should sum to 1 for each X'.



7. (10 points) Using the new game tree in part 6, solve the non-deterministic game. Specifically, provide the *expected* game theoretic value of nodes $A \ldots G$ for the new tree. Round your answers to the nearest 2 decimal places (i.e. $X.XX$). Hint: this is explained on slide p.58.

$$\mathbb{E}(D) = 3 \times 0.25 + 4 \times 0.75 = 3.75$$
$$\mathbb{E}(E) = 7 \times 0.75 + 2 \times 0.25 = 5.75$$
$$\mathbb{E}(F) = 8 \times 0.75 + 1 \times 0.25 = 6.25$$
$$\mathbb{E}(G) = 0 \times 0.25 + 6 \times 0.75 = 4.5$$
$$\mathbb{E}(B) = 3.75 \times 0.75 + 5.75 \times 0.25 = 4.25$$
$$\mathbb{E}(C) = 6.25 \times 0.25 + 4.5 \times 0.75 = 4.94$$
$$\mathbb{E}(A) = 4.25 \times 0.25 + 4.9375 \times 0.75 = 4.77$$

Question 2: Natural Language Processing [40 points, 5 points each]

This question will require you to do some minimum coding or scripting to get the answers, but it is *not* a programming question. Do not turn in any code. Instead, simply put your answers in the same pdf file as

the other questions. You may use any programming language or tools.

Download the zip file `http://pages.cs.wisc.edu/~jerryzhu/cs540/handouts/WARC201709.zip`. Unzip it on your computer, you should see 363 text files. These are your essays from homework 1. We call the whole collection the corpus.

1. Briefly describe in English, the most convenient way you can think of for a computer program to break the corpus into "computer words." For example, if you use Java you may consider Scanner; if you use unix commandline you may use wc. We want you to use the simplest method that you can think of. It is OK if these computer words do not fully agree with our natural language definition of words: For example, you may have a computer word like `However,` with that comma attached. Your method is a crude "natural language tokenizer." When we mention "word" we mean your computer word.

   Using Python's `read()` function, I can get the entire text file immediately. Then I converted all words to lower cases using `lower()`, and split words into tokens by breaking up whitespaces. The result I got was a list of more than three hundred texts, and there are word tokens within each text. After that, I iterated through all word tokens and created a vocabulary using Python's dictionary. Details shown in the code below:

```
d = {}
texts = []

# read all files and store them to texts
import_path = 'WARC201709'
for text_file in os.listdir(import_path):
    f = open(os.path.join(import_path, text_file))
    texts.append(f.read().lower())

# generate the vocabulary
num_tokens = 0
for text in texts:
    tokens = text.split()
    for word in tokens:
        num_tokens += 1
        if word in d:
            d[word] += 1
        else:
            d[word] = 1

print("Word tokens: %d" % num_tokens)
print("Word type: %d" % len(d))

words = []
for key, value in d.iteritems():
    words.append((key, value))
```

2. Under your method, how many computer word tokens (occurrences) are there in the corpus? How many computer word types (distinct words) are there in the corpus?

   ```
   Word tokens:  205117
   Word type:  17173
   ```

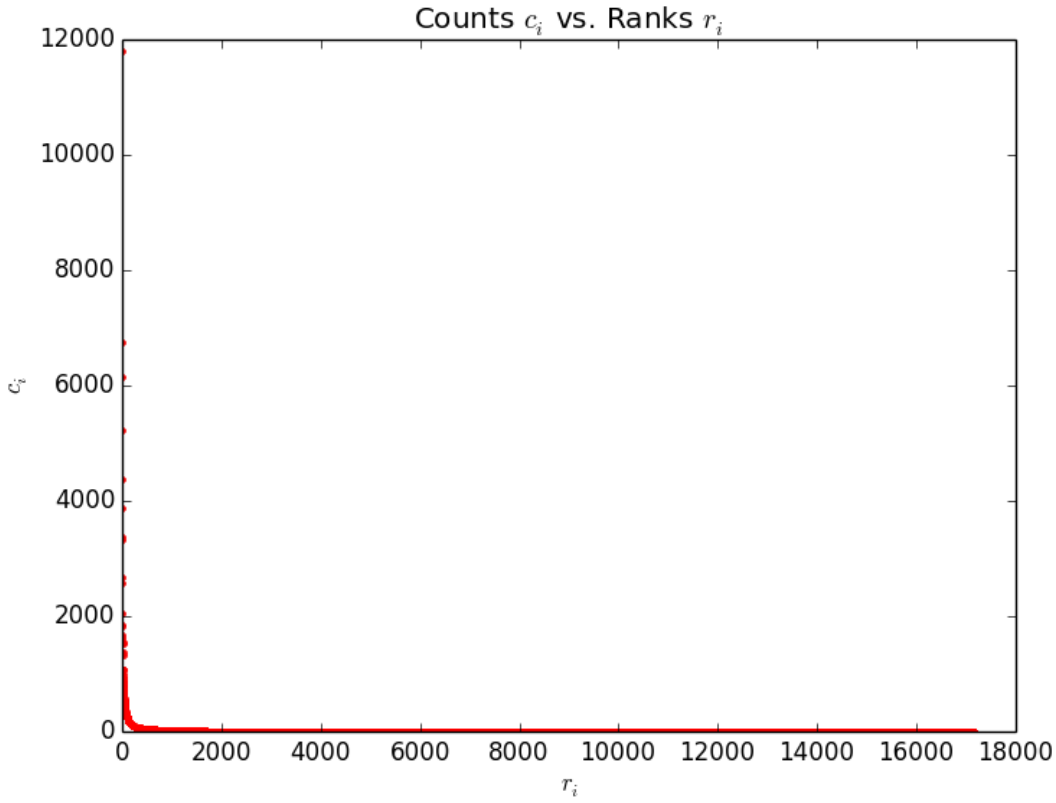
3. Sort the word types by their counts in the corpus, from large to small. List the top 20 word types and their counts.

```
Top 20:
the 11791
to 6760
of 6139
and 5235
in 4386
a 3875
that 3385
is 3325
be 2667
ai 2568
will 2042
for 1838
it 1833
as 1685
are 1616
on 1543
not 1528
this 1371
with 1322
by 1078
```

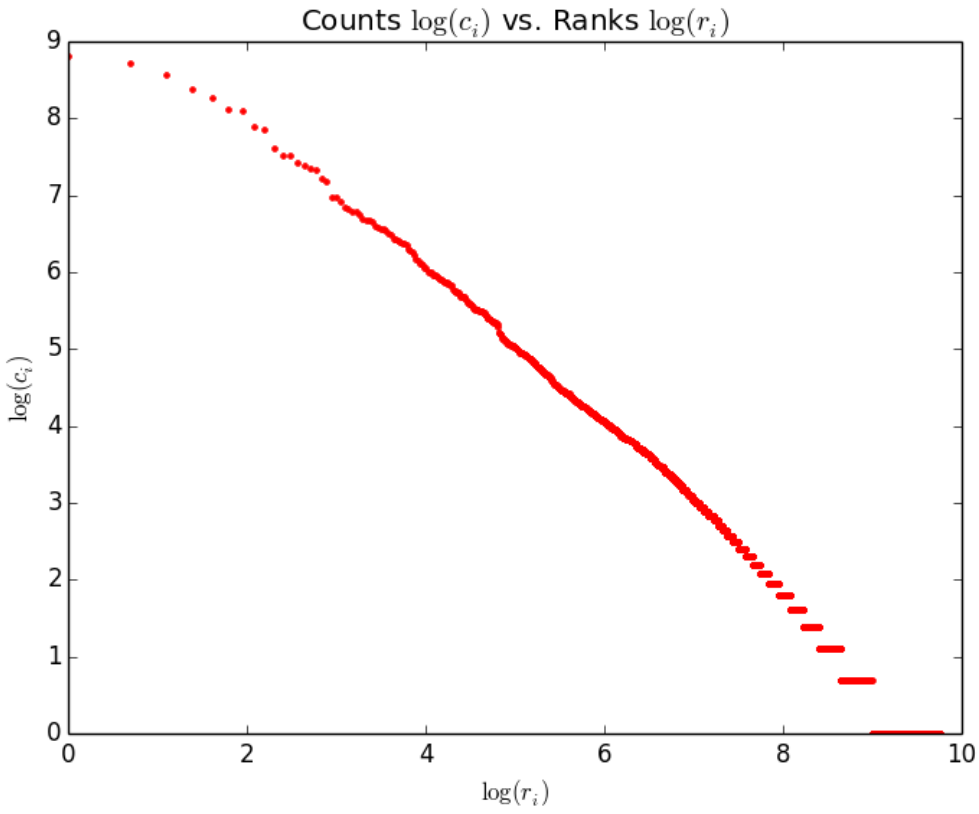
4. Pick 20 bottom word types (they should all have count 1) and list them. You may want to include some strange ones, if any.

```
Bottom 20:
"...  1
"..on 1
"2016 1
"[t]here 1
"achieve" 1
"as 1
"back 1
"bad 1
"been 1
"broad 1
"buffer" 1
"changes 1
"competition 1
"costly" 1
"courts 1
"current 1
"dinosaurs" 1
"education, 1
"enabled 1
"ethical 1
```

5. Let the word type with the largest count be rank 1, the word type with the second largest count be rank 2, and so on. If multiple word types have the same count, you may break the rank tie arbitrarily. This produces $(r_1, c_1), \ldots, (r_n, c_n)$ where r_i is the rank of the ith word type, and c_i is the corresponding count of that word type in the corpus; n is the number of word types. Plot r on the x-axis and c on the y-axis, namely each (r_i, c_i) is a point in that 2D space (you can choose to connect the points or not).



6. Plot $\log(r)$ on the x-axis and $\log(c)$ on the y-axis. You may choose the base.



7. Briefly explain what the shape of the two curves mean.

   The first curve demonstrates the relationship between word counts c_i vs ranks r_i, while the second curve demonstrates that of $\log(c_i)$ vs $\log(r_i)$.

   The shape of the first curve shows that the words with high counts (concentrating on first 500 ranks) are way fewer than those with low counts (the rest of the ranks). The count drops dramatically as the rank increases.

   The shape of the second curve shows that there's linear relationship between $\log(c_i)$ and $\log(r_i)$. We can derive that

   $$\log(c_i) + \log(r_i) \approx 10 \Rightarrow \log(c_i) = 10 - \log(r_i) \Rightarrow c_i = e^{10} \cdot \frac{1}{r_i} \Rightarrow c_i \propto \frac{1}{r_i}.$$

   In addition, the reason why there are line segments in the tail (where $\log(r_i) > 8$) is that there are two many words with counts $1, 2, \ldots$ Therefore, as the rank changes significantly, the count does not change too much.

8. Discuss *two* potential major issues with your computer words, if one wants to use them for natural language processing.

   (a) The method I chose does not filter punctuations. Words like `We,` and `We` are counted as two word types. This leads to potential problems, e.g., not categorizing all words completely.

   (b) I chose to convert all words, no matter what its meaning is, to lower case. This includes changing `US` to `us` and `WHO` to `who`. Two originally different word types are converted to the same word type now. This leads to problems too.