CS 540: Introduction to Artificial Intelligence
Homework Assignment # 5

Assigned: 10/9
Due: 10/16 8:50am

Hand in your homework:

If a homework has programming questions, please hand in the Java program. If a homework has written questions, please hand in a PDF file. Regardless, please zip all your files into hwX.zip where X is the homework number. Go to UW Canvas, choose your CS540 course, choose Assignment, click on Homework X: this is where you submit your zip file.

Late Policy:

All assignments are due at the beginning of class on the due date. One (1) day late, defined as a 24-hour period from the deadline (weekday or weekend), will result in 10% of the total points for the assignment deducted. So, for example, if a 100-point assignment is due on a Wednesday 9:30 a.m., and it is handed in between Wednesday 9:30 a.m. and Thursday 9:30 a.m., 10 points will be deducted. Two (2) days late, 25% off; three (3) days late, 50% off. No homework can be turned in more than three (3) days late. Written questions and program submission have the same deadline.

Assignment grading questions must be raised with the instructor within one week after the assignment is returned.

Collaboration Policy:

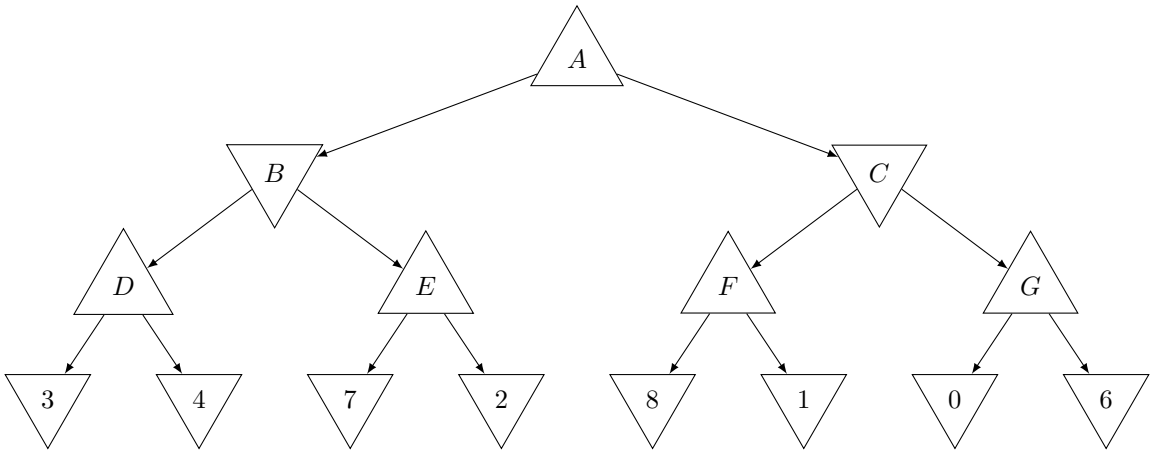
You are to complete this assignment individually. However, you are encouraged to discuss the general algorithms and ideas with classmates, TAs, and instructor in order to help you answer the questions. You are also welcome to give each other examples that are not on the assignment in order to demonstrate how to solve problems. But we require you to:

- not explicitly tell each other the answers
- not to copy answers or code fragments from anyone or anywhere
- not to allow your answers to be copied
- not to get any code on the Web

In those cases where you work with one or more other people on the general discussion of the assignment and surrounding topics, we suggest that you specifically record on the assignment the names of the people you were in discussion with.

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.

   From bottom to top, the theoretic values of each node could be calculated as follows.

   - $v(D) = \max(3, 4) = 4$.
   - $v(E) = \max(7, 2) = 7$.
   - $v(F) = \max(8, 1) = 8$.
   - $v(G) = \max(0, 6) = 6$.
   - $v(B) = \min(v(D), v(E)) = \min(4, 7) = 4$.
   - $v(C) = \min(v(F), v(G)) = \min(8, 6) = 6$.
   - $v(A) = \max(v(B), v(C)) = \max(4, 6) = 6$.

   For grading:

   - **A, B, C**: 4 points for each node.
   - **D, E, F, G**: 2 points for each node.

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.

   From bottom to top,

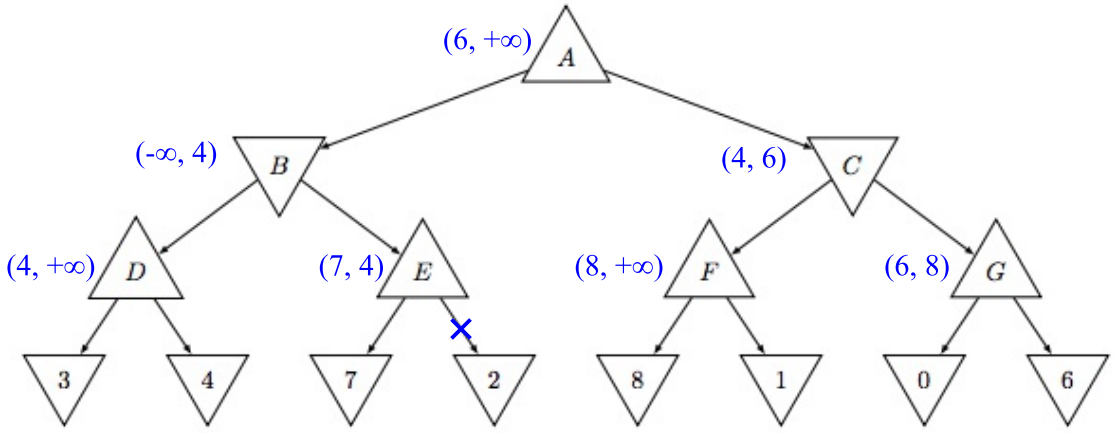
   - **A**: $\alpha(A) = 6,\ \beta(A) = +\infty$.

- **B**: $\alpha(B) = -\infty,\ \beta(B) = 4$.
- **C**: $\alpha(C) = 4,\ \beta(C) = 6$.
- **D**: $\alpha(D) = 4,\ \beta(D) = +\infty$.
- **E**: $\alpha(E) = 7,\ \beta(E) = 4$.
- **F**: $\alpha(F) = 8,\ \beta(F) = +\infty$.
- **G**: $\alpha(G) = 6,\ \beta(G) = 8$.

For grading:

- **A, B, C**: 2 points for each node. One α or β value corresponds to 1 point.
- **D, E, F, G**: 1 points for each node. One α or β value corresponds to 0.5 point.

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



Let (α, β) be a pair of α, β values for each node (shown on the above figure), observe that

$$\alpha(E) = 7 > 4 = \beta(E)$$

after node 7 was visited, then the remaining branch E-2 would be pruned.

For grading:

- earn 5 points if branch E-2 was correctly shown.
- earn 2 points if $\alpha(E) > \beta(E)$ was mentioned but the branch was incorrect.

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

Similar to A^* search that avoids touching the whole tree or in CSP that we could backtrack early or focus on the good part of the tree, alpha-beta pruning help to reduce the searching space so as to skip unnecessary work. This is also the common motivation for all pruning techniques.

For grading:

- award 5 points if the answer is correct.
- deduct 2 points if they said time complexity could be reduced.

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*?

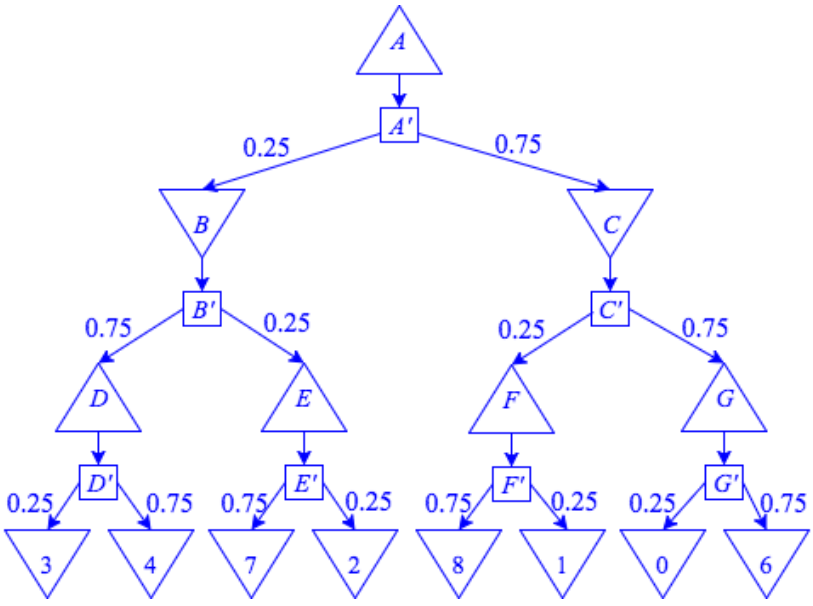
   When the player is currently at a particular node,

$$
\begin{aligned}
& P(optimal) && (1)\\
= \; & P(head) + P(tail) * P(optimal \mid tail) && (2)\\
= \; & 0.5 + 0.5 * 0.5 && (3)\\
= \; & 0.75. && (4)
\end{aligned}
$$

   For grading:

   - deduct 2 points if derivation is missing.

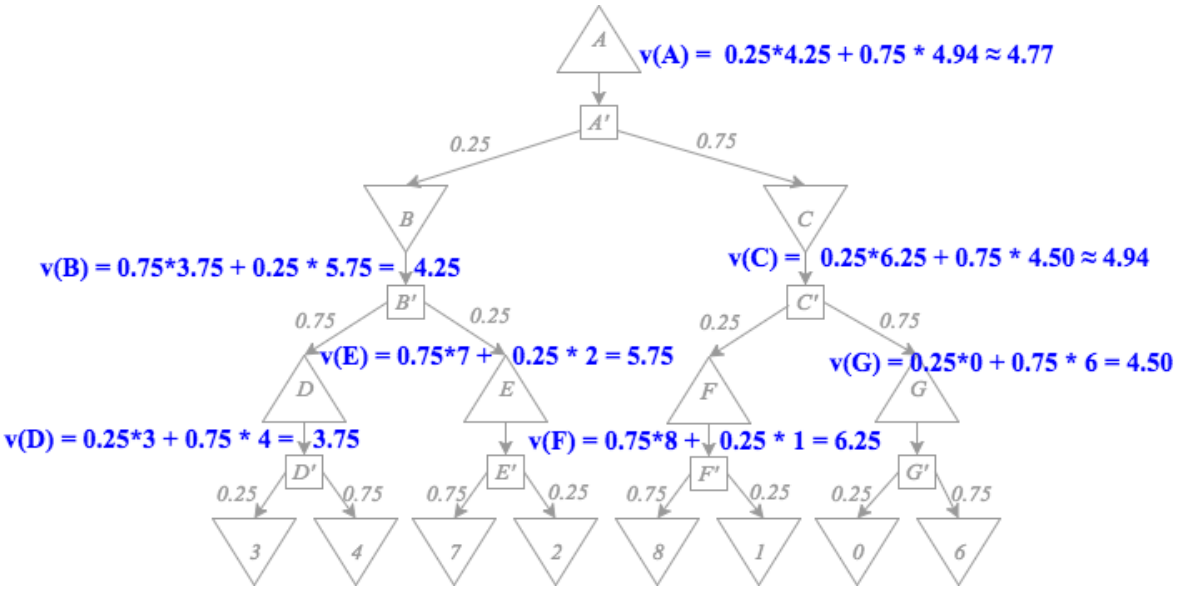
6. (5 points) Given the game described in part 5, redraw the game tree as follows. For each non-terminal node X, introduce a new chance node X'. Let X' be the only successor of X, while the original successors of X are now given to X'. Label the probability on each edge from X' to its successors. These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



   For grading:

- each value of the probability correspond to 0.5 point.
- award 0 points if more than 10 values are incorrect.

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.



For grading:

- **A, B, C**: 2 points for each node.
- **D, E, F, G**: 1 points for each node.

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.

Read the original files, use space-separated strings to represent words and convert all the letters to lowercases. In Java, this can be simply done by calling String's split() or Scanner's next(), then use String's toLowerCase() to convert all the letters to lowercases.

For grading:

- earn 5 points if their strategy used space to separate words.
- they could also ignore non-alphabetic characters and convert uppercases to lowercases, but no extra points would be awarded for such cases.

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 occurrences: $205,557$, and there are $17,100$ distinct words.

   For grading:

   - **word occurrences**: 3 points if the number is between $180,000$ and $220,000$.
   - **# distinct words**: 2 points if the number is between $15,000$ and $20,000$.

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

   the: 11790

   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

For grading: No need to get the exactly same output.

- **most frequent words** : 4 points if ≥ 15 words are in the above list, 3 points if the number is $\in [10, 14]$, 2 points if the number is $\in [5, 9]$, 1 points if the number is $\in [1, 4]$, 0 points if none.
- **word count**: 1 point the count approximate the correct count.

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.

   misinterpreted: 1

   capacity,: 1

   salary: 1

   ipad,: 1

   mistreat: 1

   spontaneous: 1

   analytic,: 1

   pollution: 1

   ?i: 1

   abrupt: 1

   gps-equipped: 1

   heading.: 1

   pretend: 1

   counters: 1

   10.4: 1

   undermining: 1

   beckon: 1

   mercedes-benz: 1

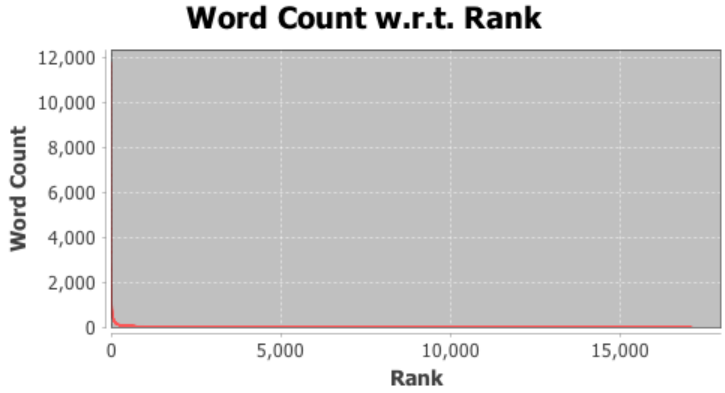
   karjakin,: 1

   tailor-made: 1

   For grading: No need to get the exactly same output.

   - 5 points if no frequent words appears.
   - do not deduct any points if they didn't show the word count.

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).

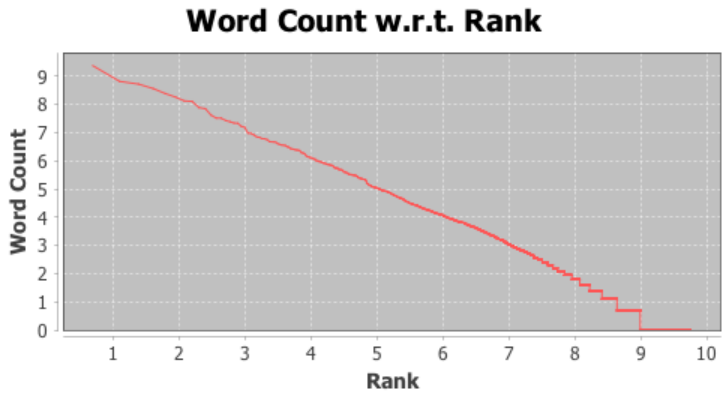


For grading:

- 3 points if the shape looks similar.
- 2 points if the scale of the x-axis and y-axis is correct.

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

   Using e as base, the plot could be redrawn as below.



For grading:

- 3 points if the shape looks similar.

- 2 points if the scale of the x-axis and y-axis is correct.

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

   Some observation:

   - The top few frequency ranks are taken up by function words i.e. "the", "a" and "and".
   - The bottom-ranked words display lots of ties in frequency.
   - Most of the words in the corpus have a frequency well below the mean.

   The curve drawn at 5 has a long tail and the shape is close to an exponential function, meaning the words are super unbalanced. The curve drawn at 6 reshaped the previous one and its shape approximate a linear function.

   For grading:

   - 5 points if reasonable.

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

   Issues:

   - punctuations attached.
   - failed to correct misspelled words.
   - treated different tenses of a same word as different words (e.g. undermining and undermine).
   - works might contains numbers (i.e.: "10.4")
   - works might contains non-ASCII characters.

   For grading:

   - 5 points if 2 issues were pointed out.
   - 2 point if only 1 issue was pointed out.