CS 5200 F 19 Take-Home Prelim 1

George Markowsky

2019-09-16

Due Monday, September 23, 2019 By 11:59 pm

I ran out of time to customize the exam so I will just release one exam for everyone. Use the the ID that you received as part of Task 1 to act as your digital signature. We will use the ID and Password in the near future.

You are not permitted to discuss the take-home exam with anyone other than me. If you have a question, email me and I will try to answer it. You may consult your textbook, class notes, and class handouts for this exam. You may also consult reference documents for Python. You may look up general facts and standard definitions, but you should not search for solutions to these problems on the Internet. You should certainly not copy solutions from the Internet. You may also use tools such as spreadsheets. If you use ideas from sources or tools that are not standard for this course, please cite them in a proper manner.

When you write proofs, clearly state what you are proving and how you are proving it. If you have any questions or are confused, please ask for clarification. You may ask questions about material presented during the semester either by email as long as it is not directly about any of the problems.

When you finish your exam, your answer should consist of a single PDF file containing your answers typed (no handwritten answers will be accepted). The PDF file should include any code that you are asked to write and any output that you are asked to supply. No screenshots will be accepted. There is no need for screen shots since Python code is written in ASCII and all output can be produced in ASCII. You just select the text you want and copy it into your answer file.

The standard reason that I get about why students want to submit written text is because it is faster. This is not necessarily the case. Microsoft Word has a very powerful equation editor that is very easy to use and fast. Similarly, writing equations in LaTeX or Libre Office is also fast. Given that once you have typed an equation you can easily reuse it using standard cut and paste you will find that in some cases typing is faster than handwriting.

Your PDF should contain the following statement at the beginning:

> *I, <your name + ID given to you>, certify that all the material in this PDF file is my original work, that I did not discuss these questions with anyone other than my instructor, and that I did not copy work from anyone for this examination.*

Of course, <your name + ID given to you> needs to be replaced by your name and the ID that was in the ASCII file that was given to you. If your name is John Doe, the statement should look something like, *I, John Doe 12345678, certify....*

Please write clearly and try to keep grammatical and spelling errors to a minimum.

1. (10 points) **Big O Notation** You are not permitted to use calculus or limits to solve this problem. You must work directly from the definitions of O, Θ, and Ω.

   (a) (6 points)**Without using limits, but only the definition of O** prove that $81n^3+1300n^2+300n \in O(n^5 - 15000n^4 - 10n^3)$ but that $n^5 - 15000n^4 - 10n^3 \notin O(81n^3 + 1300n^2 + 300n)$. Show all work.

   (b) (4 points) Let $f(x) = 2\sin^3(x) - 4\sin^2(x)$ and $g(x) = 2\cos^4(x) + 5\cos(x)$. Determine whether $f(x) \in O(g(x))$ or $g(x) \in O(f(x))$. You must show all work and base it directly on the definition of O. **You may not use limits.**

2. (15 points) **Simple Graph Algorithms**

   Write a Python program to answer a series of questions about a graph that is given in the file GraphData.txt. **The file GraphData.txt is available on Canvas.** GraphData.txt is an ASCII file that each line is given in the form

   Vertex1,Vertex2

   To simplify things, the vertices are given as integers, but not necessarily consecutive integers. Your Python program should start by reading in all the data in GraphData.txt and finish by producing a text file called GraphDataOut.txt which is structured as described below. You should include your Python program as a .py file and GraphDataOut.txt in your submitted PDF file. You do not need to include GraphData.txt in the PDF file. If you can't do some parts of this problem place the text "I could not do this part" in the appropriate place in the output file.

   (a) (2 points) The first line should read "The number of vertices in the graph is NumVert." where NumVert is the number of distinct vertices in the graph.

   (b) (2 points) The second line should read "The number of edges in the graph is NumEdge." where NumEdge is the number of distinct edges in the graph. Note that GraphData.txt might contain duplicate edges.

   (c) (1 point) The third line should read "Below is the adjacency list for this graph with the vertices sorted."

   (d) (3 points) The next NumVert lines of the text file should contain the adjacency list for the graph with one list element per line in the form

   Vertex,Neighbor1,Neighbor2,...

   where all neighbors of the vertex are given. The lines of the adjacency list must be sorted by initial vertex and the list of neighbors must also be sorted. Make the lines as long as necessary to include all neighbors.

   (e) (4 points) Following the adjacency list, the next line of the file should read "The number of connected components of this graph is NumComp." where NumComp is the number of connected components. Number the connected components using the numbers from range(#ConnectedComponents).

   (f) (1 point) The next line should read "The number of connected components of the graph is NumComponents." where NumComponents is the number of connected components of the graph.

   (g) (2 points) The next section of the file should have a line for each connected component. For each connected component you should start with the lowest numbered node, followed by the number of nodes in the component, followed by the number of edges. The connected components must be listed by increasing order of the smallest vertex. For example, consider the graph that has nodes 21, 14, 5, 9, 2, 6, and 8 and edges (21,5), (21,14), (14,5), (2,9), (9,6), (6,8), and (8,2). This graph has two components so the entries should read
   2,4, 4
   5,3, 3

3. **(15 points) Proof by Induction**

   Let function f be as below.

```
def f(n):
   if n <= 0:
      return 1000
   elif n < 2:
      return 7000
   else:
      return f(n-1) + f(n-2)
```

   Let function sf be as below.

```
def sf(n):
   if n < 0:
      return 0
   else:
      return sf(n-1) + f(n)
```

   (a) (5 points) Conjecture a very simple linear relationship between f and sf. By simple relationship I mean something that looks like

   $$\text{sf(n)} = \text{SimpleExpression(n)}$$

   where SimpleExpression(n) involves no more than 4 terms which could be various values of f and constants. You may not use summation notation or recursion on the right side of the equation.

   (b) (10 points) Prove, using induction, that the relationship that you conjectured in part (a) is correct. Be sure to set your proof up correctly and to list explicitly the steps of a proof by induction.

4. **(10 points) Probability**

   Let Vec_n be the set of all vectors of length n each component of which comes from range(n). For example, (3, 0, 2, 2) $\in \text{Vec}_4$. If v $\in \text{Vec}_n$, a *quirk* is defined as a pair (i, j) such that $0 \leq i < j \leq n-1$, but v[i] $>$ v[j].

   (a) (4 points) Create a sample space consisting of the elements of Vec_3. List all the elements of this space. Turn it into a probability space by using the uniform distribution. Finally, compute the average number of quirks in members of Vec_3.

   (b) (6 points) Generalize the results of Part (a) to determine the average number of quirks in members of Vec_n. You do not need to list the elements of Vec_n.

5. **(10 points) Probability**

   Which is less likely: obtaining no heads when you flip a fair coin n times, or obtaining fewer than n heads when you flip the coin 4n times? Provide a rigorous justification for your answer.

6. **(20 points) Combinatorics**

   In this problem, we investigate the effect of various assumptions on the number of ways of placing n blocks into b distinct boxes. C(n,k) stands for n choose k, also called the binomial coefficient n choose k.

(a) (4 points) Suppose that the n blocks are distinct and that their order within a box does not matter. Argue that the number of ways of placing the blocks in the boxes is b^n.

(b) (4 points) Suppose that the blocks are distinct and that the blocks in each box are ordered. Prove that there are exactly (b + n - 1)!/(b - 1)! ways to place the blocks in the boxes. (Hint: Consider the number of ways of arranging n distinct blocks and b - 1 indistinguishable sticks in a row.)

(c) (4 points) Suppose that the blocks are identical, and hence their order within a box does not matter. Show that the number of ways of placing the blocks in the boxes is C(b+n-1,n). (Hint: Of the arrangements in part (b), how many are repeated if the blocks are made identical?)

(d) (4 points) Suppose that the blocks are identical and that no box may contain more than one block. Show that the number of ways of placing the blocks is C(b,n).

(e) (4 points) Suppose that the blocks are identical and that no box may be left empty. Show that the number of ways of placing the blocks is C(n-1,b-1).

7. **(10 points) Recursion**

Consider the Python functions shown below:

```
def f(n):
    return 3*n*(3*n-1)*(3*n-2)

def s(n):
    if n <= 0:
        return 0
    else:
        return s(n-1) + f(n)
```

Express s(n) as a polynomial of fixed degree and constant coefficients. Prove rigorously that your polynomial equals s(n).

8. **(10 points) Logical Reasoning**

Suppose there are n supposedly identical computer chips that are capable of testing each other. Suppose our test jig accommodates two chips at a time. When the jig is loaded, each chip tests the other and reports whether it is good or bad. A good chip always reports accurately whether the other chip is good or bad, but the answer of a bad chip cannot be trusted. Thus, the four possible outcomes of a test are as follows:

Chip A says	**Chip B says**	**Conclusion**
B is good	A is good	both are good, or both are bad
B is good	A is bad	at least one is bad
B is bad	A is good	at least one is bad
B is bad	A is bad	at least one is bad

Be sure to give solid reasons for your conclusions. If you cannot completely solve this problem, you can still earn points for showing how you went about solving the problem as long as your explanation is clear.

(a) (4 points) Show that if more than n/2 chips are bad, the professor cannot necessarily determine which chips are good using any strategy based on this kind of pairwise test. Assume that the bad chips can conspire to fool the professor.

(b) (3 points) Consider the problem of finding a single good chip from among n chips, assuming that more than n/2 of the chips are good. Show that floor(n/2) pairwise tests are sufficient to reduce the problem to one of nearly half the size.

(c) (3 points) Show that the good chips can be identified with Θ(n) pairwise tests, assuming that more than n/2 of the chips are good. Give and solve the recurrence that describes the number of tests.