CSCI3104: Algorithms, Fall 2018

Final Exam Questions

Notes:

- *Due date: 2pm on December 8, 2018*
- *Submit a pdf file of your written answers and one py file with your Python codes.* ***Your written answers must be typed to receive credit.*** *All Python solutions should be clearly commented. Your codes need to run to get credit for your answers.*
- *You should not ask about the exam during the office hours or post any question about it on Piazza. However, classes on Wednesday, November 28 and Wednesday, December 5 will be reserved for questions about the exam.*
- *All work on this exam needs to be independent. You may consult the textbook, the lecture notes and homework sets, but you should not use any other resources. If we suspect that you collaborated with anyone in the class or on the Internet, we will enforce the honor code policy strictly. If there is a reasonable doubt about an honor code violation, you will fail this course.*

Problems:

1. (15 pts) Doctor Jean needs your help! She's been relabeling her collection of chromosome sequences and has found three different sequences that were displaced from their original locations in the lab. She knows that two of the sequences represent two chromosomes from a human (Homo Sapiens) and the other sequence represents a chromosome from a soybean (Glycine Max).

   Help Doctor Jean by parsing in the following sequences into memory (be careful of newlines!) and applying the Longest Common Subsequence algorithm learned in recitation to each unique pair of sequences.

   Unzip the "sequence_data.zip" file located on Canvas to access the data. When backtracing through the computed two dimensional matrix to find the longest common subsequence, break ties uniformly at random. Compare the results from your implementation to determine which species the sequences pertain to (try to come up with a sensible metric that will compare the results).

(a) Show a table that maps the sequence to the species.

(b) Submit your separate python (.py) file along with your PDF submission.

Note: the data provided is minified data (first 50 lines of each sequence) from the National Center for Biotechnology Information. The data was originally in FASTA format but it was modified when removing the descriptions of sequences from the files. You can treat the file as a text file. Below are useful links for information relating to this question.

NCBI Genome Data: ftp://ftp.ncbi.nlm.nih.gov/genomes/

FASTA Format Specification: http://genetics.bwh.harvard.edu/pph/FASTA.html

2. (15 pts) Draco Malfoy is struggling with the problem of making change for n cents using the smallest number of coins. Malfoy has coin values of $v_1 < v_2 < \cdots < v_r$ for r coins types, where each coin's value v_i is a positive integer. His goal is to obtain a set of counts $\{d_i\}$, one for each coin type, such that $\sum_{i=1}^{r} d_i = k$ and where k is minimized.

   (a) A greedy algorithm for making change is the **wizard's algorithm**, which all young wizards learn. Malfoy writes the following pseudocode on the whiteboard to illustrate it, where n is the amount of money to make change for and v is a vector of the coin denominations:

```
wizardChange(n,v,r) :
   d[1 .. r] = 0       // initial histogram of coin types in solution
   while n > 0 {
      k = 1
      while ( k < r and v[k] > n ) { k++ }
      if k==r { return 'no solution' }
      else { n = n - v[k] }
   }
   return d
```

   Hermione snorts and says Malfoy's code has bugs. Identify the bugs and explain why each would cause the algorithm to fail.

   (b) Sometimes the goblins at Gringotts Wizarding Bank run out of coins,[1] and make change using whatever is left on hand. Identify a set of U.S. coin denominations

[1] It's a little known secret, but goblins like to *eat* the coins. It isn't pretty for the coins, in the end.

for which the greedy algorithm does not yield an optimal solution. Justify your answer in terms of optimal substructure and the greedy-choice property. (The set should include a penny so that there is a solution for every value of n.)

(c) On the advice of computer scientists, Gringotts has announced that they will be changing all wizard coin denominations into a new set of coins denominated in powers of c, i.e., denominations of $c^0, c^1, \ldots, c^\ell$ for some integers $c > 1$ and $\ell \geq 1$. (This will be done by a spell that will magically transmute old coins into new coins, before your very eyes.) Prove that the wizard's algorithm will always yield an optimal solution in this case.

Hint: first consider the special case of $c = 2$.

3. In the two-player game "Pandas Peril", an even number of cards are laid out in a row, face up. On each card, is written a positive integer. Players take turns removing a card from either end of the row and placing the card in their pile. The player whose cards add up to the highest number wins the game. One strategy is to use a greedy approach and simply pick the card at the end that is the largest. However, this is not always optimal, as the following example shows: (The first player would win if she would first pick the 4 instead of the 5.)

   4 2 10 5

   (a) (10 pts) Write a dynamic programming algorithm for a strategy to play Pandas Peril. Player 1 will use this strategy and Player 2 will use a greedy strategy of choosing the largest card.

   (b) (10 pts) Prove that your strategy will do no worse than the greedy strategy for maximizing the sum of each hand.

   (c) (10 pts) Implement your strategy and the greedy strategy in Python and simulate a game, or two, of Pandas Peril. Your simulation should include a randomly generated collection of cards and show the sum of cards in each hand at the end of the game.

4. A common problem in computer graphics is to approximate a complex shape with a bounding box. For a set, S, of n points in 2-dimensional space, the idea is to find the smallest rectangle, R. with sides parallel to the coordinate axes that contains all the points in S. Once S is approximated by such a bounding box, computations involving S can be sped up. But, the savings is wasted if considerable time is spent constructing R, therefore, having an efficient algorithm for constructing R is crucial.

(a) (10 pts) Design a divide and conquer algorithm for constructing R in O($\frac{3n}{2}$) comparisons.

(b) (10 pts) Implement your algorithm in Python. Generate 50 points randomly and show that your bounding box algorithm correctly bounds all points generated. Your code should output the list of points, as well as the coordinates of R. You should include an explanation of the results in your pdf file with your algorithm.

5. (10 pts) Professor Voldemort has designed an algorithm that can take any graph G with n vertices and determine in $O(n^k)$ time whether G contains a clique of size k. Has the Professor just shown that $P = NP$? Why or why not?

6. (10 points) Consider the special case of TSP where the vertices correspond to points in the plane, with the cost defined for an edge for every pair (p, q) being the usual Euclidean distance between p and q. Prove that an optimal tour will not have any pair of crossing edges.