CSE 6140 Assignment 3
due Oct 24, 2017 at 6pm on T-Square

(First submission due Oct 22, updates can be done afterwards until deadline)

Please read and follow all instructions carefully.

Submission Instructions

- You should submit 3 separate files on TSquare:
  - A single typed PDF for the cover letter and your solutions to Problems 1 and 2, titled `<GTusername>_HW3_pbs12.pdf` (e.g. `enathan3_HW3_prob1.pdf`)
  - A single typed PDF for your report for Problem 3 titled `<GTusername>_HW3_report.pdf` (e.g. `enathan3_HW3_report.pdf`)
  - A single zip file as is explained in the "Deliverables" section of Problem 2" for the programming portion of Problem 3 (note: do not put the other PDFs in this zip file). This file should be named `<GTusername>_HW3_program.zip` (e.g. `enathan3_HW3_program.zip`)
- Do not submit .pngs, .pages files, .docx files, .jpgs, .rtfs or any other formats in nested zipped files. Your solutions should be typed, and should be in PDF files.
- Do not submit .rars, .tars, .tar.bz2s, .tar.gzs, .7zs, or any other type of archive file. Your programming assignment should be in a zip file.
- Failure to comply with ALL of these instructions will cause you to lose up to all of the points on this assignment.
- Remember the academic honor code: you are not allowed to copy from other students, you may not use material from prior classes, and googling questions on the internet is not allowed. Any evidence of cheating will be reported and the proper procedures for academic misconduct will be followed.

1 Fiber Connectivity

You have been hired as a consultant for Google. They wish to install fiber optic cables throughout the state of California, where a cable would go from one city to another. Imagine we are provided with a map of the state where we are given the locations of all the cities and all the roads connecting the cities. Google has asked you to determine the most efficient placement of these cables along the roads such that all cities are connected by these cables. However, you must also keep in mind one very important piece of information: due to the nature of the technology, a city cannot have more than k cables connecting through it.

Formally, suppose we have: n cities, $C = \{c_1, c_2, \cdots, c_n\}$; m roads, $R = \{r_1, r_2, \cdots, r_m\}$, each connecting a pair of cities; and a constant $k \in \{2, ..., |C| - 1\}$. If we are given a subset T of roads, $T \subseteq R$, then define $d_T(u)$ as the number of roads that city u is connected to using only roads in T. We want to find a subset T such that $|T| \leq |C| - 1$ such that $1 \leq d_T(u) \leq k$ for all nodes $u \in V$, and all cities in C are connected together.

Prove that this problem is NP-Complete.

2 "In Hartford, Hereford, and Hampshire...

... hurricanes hardly *ever* happen." Unfortunately, though, hurricanes do happen elsewhere. As such, you've been formally requested by the British prime minister, Sir Henry Higgins, to lead the project on setting up emergency bunkers throughout England in hopes of reducing casualties during a hurricane. In particular, inhabitants of every town in England should be able to reach a bunker in a reasonable amount of time.
One proposal is to build a bunker in each and every town, thus eliminating travel time and ensuring safety for all. However, due to deployment and maintenance costs, this solution turns out to be outside of the allocated budget. A much more cost effective strategy is to build a single bunker at a reasonable distance from all the towns, but that will lead to long travel times and congestion along the roads, which is expected to increase casualties.

Instead, you propose a happy medium: build a set of bunkers such that a town either has a bunker built in it, or is directly connected to a town which has a bunker. This reduces the number of bunkers needed, while still allowing all inhabitants ample time to get to safety. Satisfied with your proposition, PM Higgins allocates funds for this project (code-named Pygmalion) and asks you to come up with a method to solve it.

You formally set up the problem, `PYGMALION`, as follows: Given an undirected graph G, where nodes represent towns and edges represent roads, and given a number k, is there a way to build k bunkers at k different towns so that every town either has its own bunker, or is connected by a (direct) road to a town that does have a bunker?
Thinking about a solution, you sadly realize what you've gotten yourself into: Show that `PYGMALION` is NP-Complete. Follow all the steps we have outlined in class for a complete proof.

3 Programming

Emily has recently earned \$10,000 in cash by gambling (lucky her!). She wants to buy a brand new car. However, she doesn't have enough money yet and of course she is wise enough to not take the risk of gambling again. She usually takes the bus home while dreaming of her own car. "My dark gray car, you are the most beautiful car in the world", she imagines.

One day, while day-dreaming, her eyes fell on an advertisement which changed her life:

Do you want money to buy a car? or a home?

Join us today. Tomorrow is too late!

Pool-o-Pale Investment Co. Visit www.pool-o-pale.com.

She visits the website as soon as she gets home and reads the rules and regulations. She finds that she has to invest her money. They will pay her daily interest, a typical banking approach. She then finds a very appealing rule:

The interest rates are known in advance!

For example, tomorrow's interest rate is 3.5 %. This means that tomorrow, the bank will pay Emily 0.035×10000. The interest rate on the day after tomorrow is -2.1%, which means that they will claim 0.021×10000 of her money on that day. Emily gets excited about this: she can invest on the days with positive interest rates only! "That's great!", she thought, feeling that she was closer than ever to buying the car. But then, she goes on to read the next rule:

Every person can join the Pool-o-Pale once!

"What a bad rule!" she whispered disappointedly. This means that she has to join the plan on one given day, and remain so till some later day. Then, she will earn money at the rate equal to the summation of the interest rates on those days. "How can I earn as much money as possible? I wish I knew of an algorithm that finds the best investment period for me", she thinks. That night she slept while driving her dark gray car in her dreams.

Let's help Emily buy a car! Tomorrow morning, she is going to a branch of Pool-o-Pale. The manager will give her a spreadsheet containing the fixed interest rates from now until some days later. She has less than ten seconds to decide the interval she is going to invest within. You are going to help her find an efficient algorithm to accomplish this. You can, because you know how to design and analyze efficient algorithms! ☺

Suppose the manager will give her a text file containing on its first line, n, the total number of days in the plan. Then, at line i she will receive a (positive or negative) real number indicating the daily interest rate, say a_i. If you really want to help her, you have to find the indices j and k such that $\sum_{j \le i \le k} a_i$ is maximum. Assume that there exists at least one day where the interest rate is positive (otherwise, there is no reason why Emily should invest). Remember you have only ten seconds.

You, as an algorithm expert, should try and analyze the following proposals:

- A brute-force approach for this problem seems very naive. You can design a faster algorithm. Believe in yourself! Implement a *divide-and-conquer* approach by splitting the array into two halves. The best solution will either be:

  - fully contained in the first half
  - fully contained in the second half
  - such that its start point is in the first half and its end point in the second half

  The first two cases are handled recursively. The third one is a linear search.

- Secondly, you will implement a more clever solution by *dynamic programming*: Assume that the days are indexed by the set $I = \{1, ..., n\}$. Let $B(j)$ denote the maximum sum of interest rates that can be obtained if $j \in I$ is Emily's last day of investment. Then, it is easy to see that $B(j)$ can be computed using the following recurrence relation:

  $$B(j) = \begin{cases} 0 & j = 0 \\ \max\{B(j-1) + a_j, 0\} & \text{otherwise} \end{cases}$$

  In words, if j is Emily's last day of investment, the maximum interest rate she can obtain up to j is either: the maximum interest rate she can obtain up to $j-1$, plus that of the j-th day (if $B(j-1) + a_j \geq 0$); or zero (if $B(j-1) + a_j < 0$, i.e. there is no investment interval ending on day j that is profitable).
  You can easily (and efficiently) compute $B(j), \forall j \in I$ using a bottom-up approach. Once you have computed all the necessary $B(j)$ values, you can solve Emily's problem by returning the the best i and j values (Hint: the best i and j correspond to the $B(j)$ that is maximum among all $j \in I$).

You will help Emily by implementing the two aforementioned algorithms (divide-and-conquer and dynamic programming). Your algorithms should take inputs and generate outputs as follows.

Input: The first line contains two numbers. The first one, n, is the number of days. The second one, k, is the number of instances of the problem you should solve. Then, the next k lines contain n comma separated values. The input files are named `<n>.txt`, e.g. 7.txt and have the form:

```
7,3
-1.5,3.4,-3.1,1.7,2.7,-4.8,3.4
-1.2,3.8,-6.1,9.7,2.8,-5.8,1.4
-3.5,6.4,-3.1,1.7,1.7,-1.8,3.2
```

Output: For each algorithm and each input file there should be an output file with k lines. Each line has four numbers. The first one is the maximum value of interest rate Emily will receive. The next two numbers are the indices of the the start and end days of optimal investment, and are in the range $[1, n]$. The last value is the running time of the corresponding algorithm in milliseconds. Values are separated by commas, and non-integer values are output with two decimal digits.

```
4.78,2,6,1554.21
...
...
```

Your should submit the output for both algorithms corresponding to the input files provided. Your outputs should be named as

```
<GTusername>_output_<algorithm>_<n>.txt
```

where `<algorithm>` should be either `dc` (for divide and conquer) or `dp` (for dynamic programming). Put all output files in a folder named `output`.

Sample data for debugging: We provide a sample input file with ($n = 10, k = 10$) in `10.txt`, and the corresponding sample output file in `enathan3_output_dc_10.txt`. If your algorithms are correct, they will have the same values for the first three columns of the sample output file. Note that you should not submit your output file for this sample dataset.

Deliverables: You should create a zip file for the programming portion of this assignment named, `<GTusername>_HW3_program.zip`, that includes the following:

- Source for the two algorithms, well structured and commented. Similar to Assignment 1, you are allowed to use Python, Java, C, or C++.
- A `README` file explaining how to run your codes. If you use Java, C, or C++, please also include the command(s) you use to compile your code (we should be able to compile (if necessary) and run your code and see the output files generated).
- A folder named `output` containing the corresponding output of the input files provided in. The output should be in the format described in the "Output" section above.

You should also create a report and submit it as a PDF titled `<GTusername>_HW3_report.pdf` (e.g. `enathan3_HW3_report.pdf`) on t-square. The report should include:

- A description of your divide and conquer algorithm.
- A description of your dynamic programming algorithm.
- Time and space complexity analysis of both algorithms.
- A single graph with two lines that shows how the average running time of your algorithms grows with n. For each input size, n, average the running time over the k instances in the input file for that value of n.
- Observations about your empirical results that tie back into your time and space complexity analysis.
- Discussion on how the two algorithms compare with each other in terms of the complexity and empirical performance.