MCS 320 Project 4

Problem One. In this problem we will explore how eigenvalues of *minors* of symmetric matrices are related.

(a) First define a Sage function RSM(d) which takes as input an integer $d \geq 1$ and outputs a $d \times d$ *random symmetric matrix* as follows: the entries on and above the diagonal are real numbers between -1 and 1 chosen uniformly at random and the entries below the diagonal are set so that the matrix is symmetric, i.e. so that the element in place $[i, j]$ is equal to the element in place $[j, i]$. In order for this problem to work well with the second half of this problem, it will be helpful if the entries are part of the *real double field* RDF in Sage.

Here is a sample:

```
sample input: RSM(4)

sample output:

[-0.40482564988963143   -0.925268159250711  -0.9466460098221481 -0.13880541404271063]
[  -0.925268159250711  0.24631464819411786  0.06858343814596823   0.4984169133094949]
[ -0.9466460098221481  0.06858343814596823   -0.830519246481698  -0.5345568389257767]
[-0.13880541404271063   0.4984169133094949  -0.5345568389257767  -0.9933571710996867]
```

Include a sample of RSM(5) as part of your answer.

(b) Write a sage function evalArray(d) which takes in an integer $d \geq 1$ and outputs a list L as follows. Generate a random symmetric matrix M using the function from part (a); we will use this same matrix M for the entirety of the function. For each $1 \leq j \leq d$, consider $j \times j$ square matrix M_j given by top left $j \times j$ entires of M. Append to the list L the pairs (λ, j) for all eigenvalues λ of M_j.

Here is a sample:

```
sample input: evalArray(3)

sample output:

[[-0.41098068588816417, 1],
[-0.47965569366818306, 2],
[0.3504327303018379, 2],
[1.08493819390883, 3],
[-0.44931220816173656, 3],
[-0.7329542806790152, 3]]
```

Include the plot list_plot(evalArray(20)) as part of your answer.

Problem Two. Recall a *permutation* of length n is a list of the integers $1, 2, \ldots, n$ rearranged. In this problem we will explore two statistics known as *inversions* and *major index.*

(a) The *Major index* of a permutation is defined as follows: find all indices j for which the entry at j is larger than the entry at $j+1$ and take the sum of all such j. Here, the indexing starts at 1 rather than 0. For instance the major index of $[1, 4, 2, 5, 3]$ is $2 + 4 = 6$ since at positions 2 and 4 we have $4 > 2$ and $5 > 3$. Write a Sage function myMaj(perm) which takes in a permutation perm and outputs the major index. Include as part of your answer the output for myMaj([1,3,2,5,4]).

(b) Given a permutation of n, the number of *inversions* is the number of pairs $1 \le i < j \le n$ so that the ith entry is larger than the jth entry. For instance, the number of inversions of $[1, 4, 2, 5, 3]$ is 3. Write a Sage function called myInv(perm) which computes the number of inversions of the permutation perm. Include as part of your answer the output for myInv([1,3,2,5,4]).

(c) Define two Sage functions majorArray(d) and invArray(d) which both take in an integer $d \ge 1$. The function majorArray(d) outputs a list whose jth entry is equal to the number of permutations of length d whose major index is equal to j. Similarly, the function invArray(d) outputs a list whose jth entry is the number of permutations of length d with exactly j inversions. For example:

```
sample input: majorArray(4)
sample output: [1, 3, 5, 6, 5, 3, 1]

sample input: invArray(5)
sample output: [1, 4, 9, 15, 20, 22, 20, 15, 9, 4, 1]
```

Include as your answer the output to majorArray(7) and invArray(7).

Hint: It may be useful to note that for a permutation of length d, the major index and number of inversions are both between 0 and $d(d-1)/2$ inclusively.

Problem Three. At a university of 35000 people, Professor A has been informed that he will win a Nobel prize. He tells some close colleagues, and they tell their friends and so on; the information begins to spread. The news spreads at a rate proportional to the number of people who have heard it times log of the ratio of those who have heard it. At first, only one person—Professor A—knows; after 5 days, 4 people know. If we let $y = y(t)$ denote the number of people who know on day t, we may model the spread of the information by the ordinary differential equation

$$y' = py(\log 35000 - \log y)\,.$$

(a) Set up and solve the initial value problem above in Sage. Use the fact that on day 5 there are 4 people who know to estimate the constant p.

(b) For the value of p found in part (a) and plot the slope field of the differential equation along with the solution. For full credit, you must plot the slope field and the solution in a single plot.

(c) How long does it take until 25% have heard?

(d) At some point, the rate of spread of information begins to slow down. At what time t does this occur?

(e) In the beginning, only 1 person knew, but after enough time everyone in the university will know. Thus, at some point t_{2000}, there will be 2000 people who know. Similarly, at some time t_{4000} there will be 4000 people who know. Calculate the time that passed in between these two times, i.e. compute $t_2 - t_1$.

Project Guidelines, Submission Details, and Plagiarism Warning

This project is due on **Friday, November 6, 2020 at 9 AM**. No late submissions will be accepted and no extensions will be given!

Your solution to this project must consist of a single, computer typed (not hand-written), PDF document, called **project3.pdf**, containing the **Sage code and the answers** for the four problems. For each problem, write a Sage function, titled *def problem1(), def problem2(), etc*, with the appropriate input arguments. Do not change the function names. Upload the file **project3.pdf** through Blackboard. We must be able to copy and paste your code in order to evaluate it!

This project must be solved **individually**. Under no circumstances are you allowed to copy or to collaborate with anyone else. All submitted files will be automatically checked for plagiarism. Regardless of who copied from whom, all caught in the act of plagiarism will be penalized.

In particular, using internet resources of any kind is **not** allowed. Internet sites are routinely checked for similarity to your submission, both for code content and code logic. Changing code order or variable names will not prevent plagiarism detection. In addition, do not post any content of this project to any internet sites or make it public in any other form. **The content of this project is not in the public domain!**

You are free, however, to use our course resources, such as lecture notes, text books, and official Sage documentation during the solving of this project.

If you have questions about this project, come to my online office hours using the usual Blackboard link.