CMPS 101: Winter 2016: Programming HW 3

Due: 26th February 2016

- The assignment is to be attempted in groups of two. If you choose to not work with a partner, one point will be automatically deducted from your score.
- Each group needs to submit only one set of solutions.
- The names of the group members, and their UCSC ID (@ucsc.edu email address) should prominently be written in the README file.
- The solutions need to be submitted via git as outlined in this piazza post `https://piazza.com/class/iizg7fr5ykq6ot?cid=30`. However, if there is any conflict between the Piazza guide and the instructions in this homework, the instructions in this homework have higher priority and must be adhered to.
- A test script will be provided later. You need to follow the instructions in the test script, and check that your assignment passes all the tests before submission. Otherwise you are liable to lose points.
- The submission must be tagged and submitted before 3:30 pm on the due date.
- You are required to read the homework and start working on it soon after it is assigned.
- Although no points are given for coding style, unreadable or messy code can be penalized at the grader's option.
- Clearly acknowledge sources, and mention if you discussed the problems with other students or groups. In all cases, the course policy on collaboration applies, and you should refrain from getting direct answers from anybody or any source. If in doubt, please ask the instructors or TAs.

In the class we discussed about priority queues. Here you will use priority queues for a discrete event simulation problem.

- Download the code from `https://users.soe.ucsc.edu/~vishy/winter2016/hw/discrete_event.txt`. Rename it as `discrete_event.py`.
- The code is closely based on this page: `http://algs4.cs.princeton.edu/61event/`. Read the page carefully to understand the math, and to download and play around with some of the data files.
- Currently, the code uses an inefficient array based priority queue (look for the class `ArrayPQ`) which does a brute-force $O(n^2)$ search to find the next collision event.
- Your goal is to replace the array based priority queue with an efficient heap based priority queue. Note that you are not allowed to use any of the native priority queue implementations that Python provides (e.g., `heapq` or `PriorityQueue`).
- You may modify any part of the code that you wish. However, the place where we expect the majority of your changes to happen, are clearly marked in the code.
- Write a short document summarizing the changes you made to the code. Call your document `changes.txt`. Submit it along with your code.
- Create another file `observations.txt`, and comment on whether you observed any significant benefits of switching from an Array to a Heap based Priority Queue.

What You Need to Submit Only one submission is required per group.

- A README file which contains the names of the team members and their UCSC ID.
- Submit **one** python file which is named `discrete_event.py`, where you have replaced the `ArrayPQ` class with a heap based priority queue implementation.
- Submit **two** text files, which document the changes you made to the code and your observations about whether switching to a heap made any noticeable difference to the speed.

Learning Outcomes After this homework you should

- Be able to read moderately sophisticated Python code and understand it enough in order to make changes to it

- Gained an understanding of abstract data structures. In particular you will realize the importance of encapsulating the data structure, so that it can be replaced with another one with minimal code changes.
- Developed an appreciation for how discrete event simulations work, and how efficient data structures can help researchers in fields beyond computer science.

Fun things to try (not graded):

- Try to change the physics model and see what happens to the simulations
- Create new data files and share with classmates on Piazza.