Problem Set 1

Big O

---

1. Exercise 3.7 of the textbook.

   An algorithm prints the following pattern:

   ```
   *
   * *
   * * *
   * * * *
   * * * * *
   ```

   a. What are the basic operations performed by the algorithm that you would count towards its running time?
   b. Count the number of these basic operations for the specific output shown above.
   c. The number of lines printed in the preceding pattern is 5. Assume that the algorithm can extend this pattern for any number of lines (line number *i* has *i* stars). If the number of lines, *n* is the input to the algorithm, how many basic operations are performed as a function of *n*?
   d. Write your answer to the above question as a big *O* order.

---

2. Exercise E3.10 of the textbook.

   A spreadsheet keeps track of student scores on all the exams in a course. Each row of the spreadsheet corresponds to one student, and each column in a row corresponds to his/her score on one of the exams. There are *r* students and *c* exams, so the spreadsheet has *r* rows and *c* columns.

   Consider an algorithm that computes the total score on all exams for each student, and the average class score on each exam. You need to analyze the running time of this algorithm.

   a. What are the basic operations you would count toward the running time?
   b. What is the worst-case running time as a total count (not big *O*) of these basic operations?
   c. What is the big *O* running time?
   d. Is your algorithm linear, quadratic, or some other order?

---

3. * Exercise 3.14 of the textbook

   A card game program keeps a deck of cards in an array. Give an algorithm to "unshuffle" the deck so that all the cards of a suit are grouped together, and within each suit, the cards are in ascending order or rank--consider the ace as the lowest ranked card. Since there are only 52 cards, space is not an issue so you can use extra space if needed. The only constraint is that your algorithm be as fast as possible.

   What is the worst case big O running time of your algorithm? What are the basic operations you used in your analysis? Is the average big O running time different from the worst case?

---

4. * Exercise E3.11 of the textbook.

   Two people compare their favorite playlists for matching songs. The first playlist has *n* songs, and the second has *m*. Each is stored in an array, in no particular order.

   1. Describe an algorithm to find the common songs in these lists (intersection), WITHOUT sorting or making any other changes to either list.

a. What is the worst-case big *O* running time of your algorithm? Make sure to state the basic operations used in your analysis of running time.
b. What is the best-case big *O* running time of your algorithm? Explain clearly, including all book-keeping needed to achieve this best case.

2. Now suppose you could sort either or both arrays (as part of your algorithm). Repeat the worst-case and best-case analysis for the big *O* running time. (The running time must include the time to sort.)

---

5. ** Exercise E3.15 of the textbook.

There is a highway with *n* exists numbered 0 to *n* - 1. You are given a list of the distances between them. For instance:

```
Exits:     1   2   3   4   5   6
Distances: 5   3   4   2   8   6
```

The distance from the start of the highway to exit `1` is `5`, from exit `1` to `2` is `3`, and so on.

You have to devise an algorithm that would calculate the distance between any two exits. Imagine that this distance function is called millions of times by applications that need this information, so you need to make your algorithm as fast as possible.

Describe your algorithm. What is the worst-case big *O* running time? Make sure to state all the parameters you use in the analysis and relate them to the algorithm.