# Project 3 Solutions

1. *Let* $f(n) = n(n + 1) / 2$. Of the following possibilities, state which are true about $f$ and of those, which one best describes the asymptotic class of $f$ ? (Be sure to argue your answers.)

| | | |
|---|---|---|
| a. $\Theta(n^2)$ | b. $O(n^2)$ | c. $\Omega(n^2)$ |
| d. $\Theta(n(n + 1) / 2)$ | e. $O(n(n + 1) / 2)$ | f. $\Omega(n(n + 1) / 2)$ |
| g. $\Theta(n^3)$ | h. $O(n^3)$ | i. $\Omega(n^3)$ |
| j. $\Theta(n \log n)$ | k. $O(n \log n)$ | l. $\Omega(n \log n)$ |
| m. $\Theta(n)$ | n. $O(n)$ | o. $\Omega(n)$ |

   Answer: all are true except for
   g,i : because $n^2 \le O(n^3)$ but NOT $\ge \Omega(n^3)$
   j,k : because $n^2 \ge \Omega(n \log n)$ but NOT $\le O(n \log n)$
   m,n : because $n^2 \ge \Omega(n)$ but NOT $\le O(n)$
   The best representative of $\Theta(n(n + 1) / 2)$ is (IMO) $n^2$ because it is the simplest.

2. Which of these statements are best supported by data obtained using search_spy (argue your answer):

   a. fsu::g_lower_bound has asymptotic runtime $\Theta(\log n)$.
   b. fsu::g_lower_bound has asymptotic runtime $O(\log n)$ but not $\Omega(\log n)$.
   c. seq::g_lower_bound has asymptotic runtime $\Theta(n)$.
   d. seq::g_lower_bound has asymptotic runtime $O(n)$ but not $\Omega(n)$.

   Answer: a and d.
   a and not b because the data for the fsu:: versions indicates that all searches require about log $n$ comparisons
   d and not c because the data for alt:: versions indicate some searches require very few comparisons while others require almost $n$ comparisons.

3. State an asymptotic runtime for each sort algorithm that is best supported by data gathered with sort_spy. Argue your answer using collected data, and also discuss characteristics of the algorithm body that support your answer.

   g_selection_sort = $\Theta(n^2)$
   g_insertion_sort $\le O(n^2)$
   g_heap_sort $\le O(n \log n)$
   List::Sort $\le O(n \log n)$

Note: These are the best you could do with the data at this time. However, there is a theoretical result all comparison sorts can have worst-case runtime better ≥ $\Omega(n \log n)$, which leads to these more precise statements:

g_selection_sort = $\Theta(n^2)$
g_insertion_sort ≤ $O(n^2)$
g_heap_sort = $\Theta(n \log n)$
List::Sort = $\Theta(n \log n)$

4. Describe two scenarios, one under which the namespace fsu search algorithms are appropriate and one under which the namespace seq versions are appropriate.

   Because the fsu:: and seq:: algorithms have the same assumptions on the data - namely that it is sorted - whenever we can use the fsu:: versions we are better off because they are much faster. So the only situation where we would "choose" the seq:: versions is when we are forced to because the iterators defining the search range are not powerful enough to work with the fsu:: versions. In other words, when the iterators are not random access iterators. The classic case for this is List::Iterator - which are bidirectional iterators but do not have a bracket operator and "pointer" arithmetic.

   Short answer: when the data is in a List (not an array, Deque, or Vector)