Vector vs. List

Submit Assignment

Due Tuesday by 11:59pm **Points** 100 **Submitting** a file upload **File Types** zip, doc, docx, and pdf
Available Jan 27 at 4pm - Feb 8 at 7:59am 12 days

Vector vs. List

Exercise (as posed to me by John Bentley many years ago):

- Generate N random integers and insert them into a sequence so that each is inserted in its proper position in the numerical order. **5 1 4 2** gives:
  - **5**
  - **1 5**
  - **1 4 5**
  - **1 2 4 5**
- Remove elements one at a time by picking a random position in the sequence and removing the element there. Positions **1 2 0 0** gives
  - **1 2 4 5**
  - **1 4 5**
  - **1 4**
  - **4**
- For which N is it better to use a linked list than a vector (or an array) to represent the sequence?
- The sequence grows incrementally
-

Use **std::vector** and **std::list**. Use the same algorithms for both vector and list. Do not use **std::advance()** to traverse a sequence, write an explicit loop (**std::advance()** will do optimizations that will perturb your results). The N random numbers should come from a uniform distribution, but the numbers need to be distinct (i.e., duplicate values not are allowed; that implies that you'll have to generate the sequence first and then use it. That also saves you from the mistake of measuring the cost of generating the initial random sequence as part of inserting and removing its elements from the ordered sequence).

You can use **<random>** to generate random integers and **<chrono>** for timing (or just use Unix "time" or a "wall clock" or equivalent). Run each experiment three times with different seeds for the random number generator to make it likely that your numbers are not perturbed by external factors.

Draw a graph of the times for various numbers of elements (e.g., for 100K, 200K, … elements). You can hand-draw if you don't have a suitable way of drawing from a program.

Do the experiment again using a **std::map**, and add its performance to the graph (to see the effect of using an O(log n) algorithm).

Run at least one complementary experiment to help you understand what you see. For example, store the integers in a large struct to see the effects of size, pre-allocate the list elements to see the cost of allocation, or something.

Write an explanation of why your results are the way they are – this write-up is essential, do not just send code and a graph.

The code needs to be submitted together with the results. Try not to write C or Java in C++ - if you do, you'll end up writing too much code.

Due **midnight Tuesday February 7**. I recommend to start early; people trying to do this exercise the last evening have often been surprised about the time needed.