Lab 5

Collaboration level 0 (no restrictions). Open notes.

1 Sorting lower bound and CountingSort

1. Which of the following is NOT an assumption of Counting sort:

   A. Counting-sort operates on integers

   B. Counting-sort assumes that you know the range of elements

   C. Counting-sort assumes the elements are unique

2. Which of the following is true about Counting-sort?

   I. Counting-sort runs in linear time and is always more efficient than Quicksort

   II. Counting-sort is efficient when the range of elements to be sorted is small

   III. Counting-sort operates on integers

   A. II and III B. I, II and III C. I and III

3. In which scenario would you use Counting-sort over Quicksort?

   A. sorting 25 integers in the range 0 to 20,000

   B. sorting 10,000 integers in the range 0 to 100

   C. sorting 1,000 social security numbers (integers on 9 digits, 000000000 to 999999999)

4. Suppose that we were to rewrite the last for loop in CountingSort as: for j=1 to A.length (instead of: for j=A.length down to 1). Would the algorithm sort properly? What would change?

5. Assume you have n elements in the range $\{-20, -19, ..., ..., 19, 20\}$. How would you modify Counting-sort to sort this array? What if the range was $\{10,000, ..., 12,000\}$?

6. Assume you have $n = 1,000$ integers in the range $\{0, .., 50\}$ and you need to sort them. Would you use Counting-sort or Quicksort, and why?

7. Describe one scenario when CountingSort is more efficient than Quicksort.

8. Describe one scenario when Quicksort is more efficient than CountingSort.

9. Describe one scenario when you cannot use CountingSort.

2 Sorting in practice

On the class website you will find three files, `python-insertionSort.ipynb, python-mergesort.ipynb` and `python-quicksort.ipynb`. A Jupyter notebook contains Python code interleaved with comments and plots, so the whole experience is interactive. **How to run a Jupyter notebook:**

- Go to JupyterLab (`https://jupyter.org`). On the website, you'll see two buttons: one that says *Install JupyterLab*: download and install JupyterLab on your computer. Afterwards, you can run it from the terminal with `./jupyter-notebook mergesort.ipynb`.
- Another option is to use VisualStudio which has an extension for Jupyter Notebooks, which you could install.
- Another option is to login in to Bowdoin (`login.bowdoin.edu`) and use Bowdoin's JupyterLab server.

Once you are able to run the notebooks, step through them cell by cell, read the explanations, run the code and generate the plots. Interact with the code: make changes, re-run the cells and see what happens. After stepping through all notebooks you should have seen how the sorts that we discussed in class perform in practice and compared to each other, and how seemingly small changes in the implementation lead to significant differences in running time. Overall the results will validate everything we leant in class so far and the idea that once you know the theory you can usually predict practice.