CS3000: Algorithms & Data — Summer 2024 — Laney Strange

APP 1
Due: May 8th, 2025 @ 11:30am via Gradescope

Name: Sample Solution

- APPs will be assigned towards the end of roughly two lectures each week. You'll put together a solution to a short problem that we'll all use in the following lecture. We'll have time set aside to do these in class, or you can work on your own.
- You may handwrite your solutions, or typeset them in LaTeXor another system.
- APPs will be graded on completeness. They must be submitted by 11:30am (just before lecture) on the due date. They will not be accepted late, but we drop 3 of them (out of 8 total).
- Collaboration is strongly encouraged for APPs!

Problem 1.

Below is the pseucode for INSERTIONSORT as we saw today in class.

INSERTIONSORT(A, n)

```
1  for i = 2 to n
2      key = A[i]
3      j = i − 1
4      while j > 0 and A[j] > key
5          A[j + 1] = A[j]
6          j = j − 1
7      A[j + 1] = key
```

- Give an example of an array that would result in the worst-case run-time for Insertion Sort.

  Solution:

  Anything in reverse-sorted order, such as $< 9, 8, 7, 6, 5, 4 >$

- Give an example of an array that would result in the best-case run-time for Insertion Sort.

  Solution:

  Anything in sorted order, such as $< 2, 3, 4, 5, 6, 7 >$

- In line 5, when we set $A[j+1] = A[j]$, what is keeping us from overwriting and therefore losing the value at $A[j+1]$?

  Solution:

  Line 2 helps us with this, by saving the value in $A[i]$ in *key*, and then line 3 sets $j = i - 1$, so that $j + 1 = i$.

- What is the best-case running time $T(n)$ for Insertion Sort on an input of size n? Assume that each execution of the kth line takes c_k time where c_k is a constant.

  Solution:

  $T(n) = c_1 n + c_2(n-1) + c_3(n-1) + c_4(n-1) + c_7(n-1)$
  $= (c_1 + c_2 + c_3 + c_4 + c_7)n - (c_2 + c_3 + c_4 + c_7)$

- Give a tight bound on the best-case run-time.

  Solution:

  $\Theta(n)$.