CS2500 Homework 1

Evan Wilcox

Due January 31, 2019

1.1-4
Both algorithms are trying to find the most efficient path for the situation but the traveling-salesman starts and ends at the same location and the shortest-path ends at a different location than it starts.

1.1-5
Sorting is a problem that only the best solution will do because something that is mostly sorted won't work. Finding the fastest path between two places in a city is an example of a problem that an "approximate" is good enough because even if its not the fastest it will sill get you there.

1.2-2
Insertion sort out performs Merge sort when n = [2, 43].

1-1

	1 second	1 minute	1 hour	1 day	1 month	1 year	1 century
$\lg n$	$9.9\text{x}10^{301029}$	$5.5\text{x}10^{18061799}$	$2.5\text{x}10^{1083707984}$				
$\sqrt{n}$	$1\text{x}10^{12}$	$3.6\text{x}10^{15}$	$1.3\text{x}10^{19}$	$7.5\text{x}10^{21}$	$6.7\text{x}10^{24}$	$9.6\text{x}10^{27}$	$9.96\text{x}10^{30}$
n	$1\text{x}10^{6}$	$6\text{x}10^{7}$	$3.6\text{x}10^{9}$	$8.6\text{x}10^{10}$	$2.6\text{x}10^{12}$	$3.1\text{x}10^{13}$	$3.1\text{x}10^{15}$
$n\lg n$	$6.3\text{x}10^{4}$	$2.6\text{x}10^{6}$	$1.3\text{x}10^{8}$	$6.8\text{x}10^{8}$	$7.2\text{x}10^{10}$	$7.9\text{x}10^{11}$	$6.8\text{x}10^{13}$
n^2	$1\text{x}10^{3}$	$7.7\text{x}10^{3}$	$6\text{x}10^{4}$	$2.9\text{x}10^{5}$	$1.6\text{x}10^{6}$	$5.6\text{x}10^{6}$	$5.6\text{x}10^{7}$
n^3	$1\text{x}10^{2}$	$3.9\text{x}10^{2}$	$1.5\text{x}10^{3}$	$4.4\text{x}10^{3}$	$1.4\text{x}10^{4}$	$3.1\text{x}10^{4}$	$1.5\text{x}10^{5}$
2^n	19	25	31	36	41	44	51
$n!$	9	11	12	13	15	16	17

2.1-3

```
linearSearch(A, v)
1  for j = 1 to A.length:
2      if A[j] == v:
3          return j
```

At the start of each iteration of the for loop, the subarray $A[1...j-1]$ consists of elements that are not equal to v.

2-2

a) We need to prove that for any input it has the same output every time it is run.

b) At the start of each iteration of the for loop, the subarray $A[j{+}1...A.length{-}1]$ consists of elements that are greater that $A[j]$.

Initialization: We start by showing that the loop invariant holds before the first loop iteration, when $j = A.length$. The subarray, therefore, consists of no elemets. This subarray shows that the loop invariant holds prior to the first iteration of the loop.

Maintenance: The program checks if $A[j] < A[j-1]$ and swaps them if so. Essentially, moving numbers greater than $A[j]$ to the right of it so the only numbers right of $A[j]$ are greater than it. This holds true with the loop invariant.

Termination: The condition causing the for loop to terminate is that $j < i+1$. Because each loop iteration decreases j by 1, we must have $j = i-1$ at that time. Substituting $i-1$ for j in the wording of loop invariant, we have that the subarray $A[i...A.length-1]$ consists of the elements greater than A[i].

c) At the start of each iteration of the for loop, the subarray $A[1...i-1]$ consists of sorted elements.

Initialization: We start by showing that the loop invariant holds before the first loop iteration, when $i = 1$. The subarry is an empty array so it is sorted.

Maintenance: The program increments i in order to begin sorting the next element.

Termination: The condition causing the for loop to terminate is that $i > A.length - 1$. Because each loop iteration increase i by 1, we must have $i = n + 1$ at that time. Substituting $n + 1$ for i in the wording of loop invariant, we have that the subarray $A[1...n$ consists of the sorted elements.