B505 Applied Algorithms
HW 2

2020-09-26
Wesley Liao

1.

2.
given A[1...n] containing all integers from 0 to n except one number:

```
increase size of A by 1
A[n+1] ← -1
m ← n + 1
for i ← 0 to n:
while (A[i + 1] != i) and (A[i + 1] != -1):
swap A[A[i+1] + 1], A[i+1]
if A[i + 1] == -1:
m ← i

return m
```

3.
a)
(1, 5), (2, 5), (3, 4), (3, 5), (4, 5)

b)
The array with the elements in reverse order: {n, ..., 2, 1}.
There are (n * (n - 1)) / 2 inversions.

c)
Every inversion will require a swap and comparison operation.
The running time will be directly proportional to the number of inversions.

4.
given sorted lists A[1...n] and B[1...m]:

```
i ← min(n, k)
j ← min(m, k)

while i + j > k:
if A[i] < B[j]:
j ← j - 1
else if B[j] < A[i]:
i ← i - 1

if A[i] >= B[j]:
```

return A[i]
else:
return B[j]

5.

(a) $T(n) = T(n/2) + n$

$a = 1, n = n, b = 2, d = 1$

$d \stackrel{?}{=} \log_b(a)$

$1 \stackrel{?}{=} \log_2(1)$

$1 > 0$

$T(n) = O(n)$

(b) $T(n) = T(n/5) + n^2$

$a = 1, n = n, b = 5, d = 2$

$d \stackrel{?}{=} \log_b(a)$

$2 \stackrel{?}{=} \log_5(1)$

$2 > 0$

$T(n) = O(n^2)$

(c) $T(n) = T(n/3) +$ constant

$a = 1, n = n, b = 3, d = 0$

$d \stackrel{?}{=} \log_b(a)$

$0 \stackrel{?}{=} \log_3(1)$

$0 = 0$

$T(n) = O(\log(n))$

6.

In case a. an insertion sort will likely be better because it will require only moving the few inverted datapoints.

Using a merge sort in case a. will require still merging the entire array even though it is mostly already sorted.

In case b. a merge sort will be better because there will likely be many inversions.