Deepanshu
2019CS50427
Homework 1.1

Page No.
Date

Insertion Sort

Describing Insertion sort in the way of describing algorithms shown during flipped lectures.

Problem

Sorting a given list of integers

Input specifications (given):

An array of n numbers

Output specifications (output):

1. An array of n numbers.

2. for any chosen subsequence of output array, the number appearing before in the array should not be larger than the number appearing afterwards in the array.
i.e $b_1, b_2 \text{ —— } b_k$ be a sequence from output array.
then $b_1 \leq b_2 \leq b_3 \text{ —— } \leq b_k$ where $1 \leq k \leq n$

3. Output should be one of the permutation of the input.

Instance

Instance 1 : $A = [10, 12, 4, 3, 7]$
Instance 2 : $A = [1]$

Deepanshu
2019CS50427

Page No.
Date

Pseudo code of Insertion sort

Algorithm insertionSort (int A[], int n)
(1-based indexing)
i ← n-1 // Starting from second last and considering last // element to be sorted (single element is sorted

while i > 0 // Outer loop
do temp ← A[i] // preserving A[i] so that it is not lost during swapping
j ← i+1
while j < n+1 and A[j] < temp // Inner loop
do A[j-1] ← A[j] // Swap
j++
a[j-1] ← temp
i--

Edge cases

following cases should be taken care of while writing the Insertion sort Algorithm

1. That array is already sorted.
2. That array is reversed sorted.
3. If array contains negative numbers.
4. If all elements of array are same.
5. If all numbers are prime. (I don't know why this is considered a special case, but I read it somewhere).

Deepanshu
2019CS50427

Page No.
Date

Space and time complexity

1. Space complexity

The algorithm is an "in place" algorithm. This means that no space depending on n is required.

Space complexity is of constant order. $O(1)$

2. Time complexity

Outer loop is executed n times. Therefore, complexity depends on the number of times inner loop runs.

worst case analysis (reverse sorted)

When the array is reverse sorted, the number of swaps are maximum since every number should be brought to first index for every iteration of outer loop.

Let c_1 be the number of primitive operations in inner most loop.

Total operations

$$\sum_{i=n}^{1} \sum_{j=i-1}^{0} c_1$$

$$\sum_{i=1}^{n} c_1 (i-1)$$

$$c_1 \left[\frac{n(n+1)}{2} - n \right]$$

$$c_1 \left(\frac{n(n-1)}{2} \right) \qquad O(n^2)$$

Deepanshu
2019CS50427

Page No
Date

Best case analysis (sorted)

When the array is sorted, the number of swaps are zero for every iteration of first loop.

Let c_1 be the number of primitive steps in inner loop.

Total operations $\sum_{i=1}^{n} c_1 = c_1 n$ or

$O(n)$

Remarks.

(In Doubt) → * The sorted array is not always the best case for sorting. Similarly, reverse sorted array is also not always worst case for sorting.

This is true in case of Insertion sort Algorithm

Doubts.

1. Point 5 of edge cases.
2. Is sorted array always the best case for sorting?

Note.

We can also do Binary search to put elements in partially sorted array.

Time complexity in that case: $O(n \log n)$ for all cases.

Deepanshu
2019CS50427

1/10/20

Homework

Flipped lecture 1.2

Pseudo code for descending order:

while i > 0
do temp ← A[i]
(1-based indexing)
Algorithm insertionSort (int A[], int n)

i ← n-1

while i > 0 // Outer loop
do Temp ← A[i]
j ← i+1
while j < n+1 and (A[j] > temp) // Inner loop
do A[j-1] ← A[j] // Swap
j++
a[j-1] ← temp
i--

only change.

We need to only change the comparison sign.
Therefore, loops will enter
All swaps will take place in other cases and resulting array will be reverse sorted (decreasing order)

Deepanshu

2019CS50427

Page No.
Date

Homework 1.3

Q. Formally prove that there is no n such that $n^2 \leq cn$ for all $n > n_0$.

Ans: We will prove by contradiction.

Assumption: c is positive

Suppose there exist $n = n^*$ (let say) greater than n_0 such that

$n^{*2} < cn^*$ for all $n^* > n_0$.

$n^* < c$ for all $n^* > n_0$. — (A)

~~But since n~~

Clearly $c > n^*$

$\Rightarrow 2c > n^*$ — (1)

Since $2c > n^* > n_0$, it should satisfy (A)

$2c < c$

which is a contradiction for positive c.

Therefore, our assumption ~~that~~ is wrong.

$\Rightarrow$ There is no n such that $n^2 \leq cn$ for all $n > n_0$.

Hence Proved.

Homework 1.4

Algorithm prefixAverages(a)

```
sum ← 0                          // variable to store partial sums upto i'th value
for i ← 0 to n-1 do
    sum ← sum + a[i]
    A[i] ← (double) sum / (double) (i+1)      // Avg = sum / freq.
return array A
```

We have a single loop running n times.
Running time: O(n)