ADS Homework 6

Aashish Paudel

March 26, 2019

Problem 1

(a)

See the code counting_sort.py

(b)

See the code bubble_sort.py

(c)

Soultion:

(d)

The algorithm is implemented in sort_words.py.

(e)

During bucket sort, adding the elements in their respective buckets can be done in $\theta(\mathbf{n})$.
Consider, we sort all the elements of bucket individually using insertion sort. Say, elements at each bucket is $\mathbf{n_i}$.
Then, the time complexity of sorting elements of all buckets is: $\sum_{\mathbf{i=1}}^{\mathbf{n}} \mathbf{n_i}^*$.
Thus, the combined time complexity: $\theta(\mathbf{n}) + \sum_{\mathbf{i=1}}^{\mathbf{n}} \mathbf{n_i}$.
At worst case, all the elements are in the same bucket. Thus from $*$,

$$\begin{aligned} &= \theta(n) + O(n^2) + 1 + 1 + 1... \\ &= O(n^2)) \end{aligned}$$

Thus, the time complexity: $\mathbf{O(n^2)}$.
eg- An array [1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 1.0] will be placed in the

Algorithm 1 Count the elements

// This function is used to preprocess the array Time complexity = $\theta(n+k)$

procedure PREPROCESS(arr, n)
// Determine the value of k
$k \leftarrow arr[0]$
for $i \leftarrow 1$ to n **do**
if $k < arr[i]$ **then**
$k \leftarrow arr[i]$
end if
end for
// Initialize B
for $i \leftarrow 1$ to k **do**
$B[i] \leftarrow 0$
end for
// Given the indices as respective value in arr, add the no of repetitions in B
for $i \leftarrow 1$ to k **do**
$B[arr[i]] += 1$
end for
// Get the cumulative sum of all indices left to current index
for $i \leftarrow 1$ to k **do**
$B[arr[i]] += B[arr[i-1]]$
end for
return B
end procedure
=0

This function takes preprocessed array B: Time complexity = $O(1)$

procedure COUNT(B, a, b)
return $B[b] - B[a-1]$
end procedure
=0

same bucket resulting in worst case thus giving the above time complexity.

(f)

The maximum distance between the origin and boundary of circle is 1. Thus, all the values of distance $d_1, d_2, d_3... \subset [0, 1]$. Hence, we will use bucket sort to sort these distances.

Algorithm 2 Sort the distances in unit circle

```
0: procedure EUCLIDEAN_DISTANCE(x_1, y_1, x_2, y_2)
   return sqrt((x_2 - x_1)^2 + (y_2 - y_1)^2)
0: end procedure=0
0: procedure SORT(x, y, n)
   // Intialize A and fill it with distances
0:   for i ← 1 to n do
0:     A[i] = EUCLIDEAN_DISTANCE(x, y, 0, 0)
0:   end for
   // Initialize B
0:   for i ← 1 to n do
0:     B[i] = 0
0:   end for
   // Fill elements in bucket
0:   for i ← 1 to n do
0:     B[n[A[i]]] = A[i]
0:   end for
   // Sort the elements in each bucket
0:   for i ← 1 to n do // Sort the array using any sorting algorithm
0:   end for
   return B
0: end procedure=0
```

Problem 2

(a)

The code is in the file holerith_sorting.py.

(b)

Time complexity

For individual lines of complexity please refer to holerith_sorting.py.

We have, $\mathbf{T(n)} = \sum_{\mathbf{i=0}}^{\mathbf{k}} \theta(\mathbf{n}) + \theta(\mathbf{1})$, where
$\mathbf{n}$ = number of elements
$\mathbf{k}$ = length of largest element
Time complexity $\mathbf{T(n)} = \theta(\mathbf{kn})$
For very large n, $\mathbf{k} << \mathbf{n}$. Thus,
Time complexity $T(n) = \theta(n)$

Storage Complexity

We store two large lists (ideally arrays): **buckets and output**
buckets is the recursive array where we store our numbers for sorting.
output is the final array to store the sorted numbers.
Storage complexity of output: $\mathbf{n}$ Storage complexity of bucket: $\mathbf{\Pi_{i=0}^{k} 10} = 10^k$.
Thus, the storage complexity of algorithm: $\mathbf{n + 10^k}$

(c)

We do it using radix sort.

Algorithm 3 Sort n elements from 0 to $n^3 - 1$

```
0: procedure SORT(arr, n, index)
0:     for i ← index to 0 do
0:         // Sort elements index wise using counting sort.
0:
```

Time complexity = $\theta(\mathbf{kn})$
For very large n: $\mathbf{n} >> \mathbf{k}$
Thus, Time complexity = $\theta(\mathbf{n})$, where
k = length of individual digit - 1