Algorithms and Data Structures: Homework #7

Due on March 25, 2020 at 23:00

Henri Sota

Problem 7.1

f) Given n 2D points that are uniformly randomly distributed within the unit circle, design and write down an algorithm (only pseudocode) that sorts in linear time the points by increasing Euclidean distance to the circle's origin. Write also a pseudocode function for the computation of the Euclidean distance between two 2D points.

Algorithm 1 Euclidean Distance[point]

return $\sqrt{\text{point}.x^2 + \text{point}.y^2}$

Algorithm 2 Euclidean Distance of Points[FirstPoint, SecondPoint]

return $\sqrt{(\text{FirstPoint}.x - \text{SecondPoint}.x)^2 + (\text{FirstPoint}.y - \text{SecondPoint}.y)^2}$

Algorithm 3 Insertion Sort on Distance[Bucket]

for $j = 1$ **to** Bucket.$length - 1$ **do**
  $key = \text{Bucket}[j].d$
  $i = j - 1$
  while $i > 0$ and $\text{Bucket}[i].d > key$ **do**
    $\text{Bucket}[i + 1] = \text{Bucket}[i]$
    $i = i - 1$
  end while
  $\text{Bucket}[i + 1] = key$
end for

Algorithm 4 Point Sort[A]

Declare B: Array with n elements
$n = A.length$
for $i = 0$ **to** $n - 1$ **do**
  $A[i].d = \text{Euclidean Distance(A[i])}$
end for
for $i = 0$ **to** $n - 1$ **do**
  Declare $B[i]$: **List of Point Structs**
end for
for $i = 1$ **to** n **do**
  Insert $A[i]$ into list $B[\lfloor n(A[i].d)^2 \rfloor]$
end for
for $i = 0$ **to** $n - 1$ **do**
  Insertion Sort on Distance($B[i]$)
end for
Concatenate the lists $B[0], B[1], \ldots, B[n - 1]$ together in order

Point Sort algorithm is based of off Bucket Sort algorithm, because the points being considered are within the unit circle, which means their Euclidean Distance is ≤ 1 always. In order to solve it in linear time, the points do not fall into the bucket $nA[i].d$. In this case, they need to fall on the areas between

the radii $\sqrt{\frac{1}{n}}, \sqrt{\frac{2}{n}}, \ldots, \sqrt{\frac{n}{n}}$. The bucket on which point (x, y) should be decided using $\lfloor n * (x^2 + y^2) \rfloor$.
* Each algorithm works on a Point structure that has properties (x, y, d) which refer to the x-coordinate, y-coordinate and Euclidean Distance from the origin respectively.

Problem 7.2

Consider Hollerith's original version of the Radix Sort, i.e., a Radix Sort that starts with the most significant bit and propagates step by step to the least significant bit (instead of vice versa).

c) Write down the pseudocode for an algorithm which sorts n integers in the range 0 to $n^3 - 1$ in $O(n)$ time.
In order to sort the list of integers in $O(n)$ time, there is the need for the modification of an algorithm that runs in linear time. The steps taken in order to solve this:

- Traverse the array of integers and convert each one to base n. This has to be done in order to make sure that all of the integers will have at most 3 digits ($\log_n n^3 = 3$).
- Consider the first digit (Least Significant Digit) and perform Counting Sort for each integer. There are n possible values to consider during Counting Sort. Perform Counting Sort on the second and third digit.

Counting Sort runs on linear time and since the algorithm only has to pass at most 3 times in the worst case, the time complexity is $O(n)$.

Algorithm 5 Counting Sort[A, B, i]

Declare C: Array with n elements
for $i = 0$ **to** k **do**
  $C[i] = 0$
end for
for $j = 1$ **to** $A.length$ **do**
  $C[A[j].i] = C[A[j].i] + 1$
end for
for $j = 1$ **to** $A.length$ **do**
  $C[i] = C[i] + C[i - 1]$
end for
for $j = A.length$ **to** 1 **do**
  $B[C[A[j].i]] = A[j]$
  $C[A[j].i] = C[A[j].i] - 1$
end for

* Accessing an integers i-th property in this pseudocode means to get its i-th digit from right to left.

Algorithm 6 N To 3 Radix Sort[A]

Convert all integers to base n
for $i = 1$ **to** 3 **do**
  $A[i]$ = Convert to Base $n(A[i])$
  Declare B: Array with $A.length$ elements
  Counting Sort(A, B, i)
end for