**ADS Homework 7(BONUS)**

**Problem 7.1: Sorting in Linear Time**

**F**)(Bonus)

Euclidean distance pseudocode:

*// there exists a class: class point {*

*private:*

*float x, y; } ;*

euclideanDistance(a, b)

x = a.x – b.x;

y = a.y – b.y;

distance = sqrt(pow(x, 2)+pow(y, 2));

**return** distance;

end function

**Sorting of the points:**

pointSorting(arr, n)

let d be point (0,0)

let G[0..n-1] be a new array

map<double, point> mymap;

for i=0 to n-1 do:

G[i] = euclideanDistance(arr[i], d) ;

Then set mymap[G[i]] = arr[i]

// since the points are in a unit circle, the distance to the circle’s origin is less than 1, so we apply //bucket sort

Create new array C[0..n-1]

for i=0 to n-1 do:

C[i] = 0;

for i=1 to n

insert G[i] into list C[(int)n\*arr[i]]

for i=0 to n-1

sort C[i]

then concatenate C[0] up to C[n-1] together in order

for i=0 to n-1

arr[i] will be equal to mymap[G[i]] *//* so it has been sorted

**Problem 7.2: Radix Sort**

**C**)(Bonus)

*// A function to do counting sort of array according to the digit represented by the exponent.*

**countingSort**(arr, n, exp)

let B[n] be the output array

new array C[n] *//count array*

for i=0 to n do:

C[i] = 0;

*// Store number of occurrences in C[]*

for i=0 to n do:

C[(arr[i]/exp)%n ]++;

*// make C[i] so that it contains actual position of a specific digit in B[]*

for i=0 to n do:

C[i] += C[i - 1];

*// Build the output array*

for i=0 to n do:

B[C[ (arr[i]/exp)%n] - 1] = arr[i];

C[(arr[i]/exp)%n] = C[(arr[i]/exp)%n] - 1;

*// Copy the output array to arr[]*

for i=0 to n do:

arr[i] = B[i];

end function

*// The main function sorts array arr of size n using Radix Sort*

**radixSort**(arr, n)

*// Do counting sort for first digit in base n (instead of passing digit number, n^0 = 1 is passed)*

countingSort(arr, n, 1);

*// Do counting sort for second digit in base n (instead of passing digit number, n^1 = n is passed)*

countingSort(arr, n, n);

*// Since there can be 3 digits in base n, we need to call counting sort 3 times for range 0 to n3-1, so we do counting sort for third digit in base n (instead of passing digit number, n^2 = n2 is passed)*

countingSort(arr, n, n^2);

end function

Reference:

[www.geeksforgeeks.org](http://www.geeksforgeeks.org)