Homework Assignment 3

CS 430 Introduction to Algorithms

Name : Karthik Mahesh

CWID : A20383027

1.Solution:

```
a ← sequence of values
count ← 1
max ← 1

for  i = 1 to a.length
   if (a[i]-a[i - 1]<=1) then
      count++;
   else
      count ← 1;
   end if

   if (count > max) then
      max ← count;
   end if
   i++
end for
```

Max = maximum sized streak of similar weather days

Time complexity is O(n)

2.Solution:

Consider an array A= {1,2,3,4,5,6,7} which is already sorted.
Heapsort first takes linear time to convert the array into max-heap i.e, O(n) time.
Then in for loop, there are n-1 calls to max-heapify. So the process generated by for loop is $\Theta(n \log n)$.
So the heapsort procedure have an order of growth $\Theta(n \log n)$
Therefore heapsort for the array a requires $c.n\log n$ steps where c is a constant.

3.Solution:

The best way to merge the k sorted lists is to first merge arrays 1 and 2, arrays 3 and 4, and so on. Here there are k/2 array merges of 2n which is total work of kn.

Then we can merge arrays (1,2) and (3,4) , arrays (5,6) and (7,8), and so on. Here there are k/4 merges of 4n which is total work kn.

We have to repeat this procedure till we get only one array.

There will be log(k) such merges, each with kn work. Hence total work done will be O(k.n.log(k))

4.Solution:

```
num sort(L)
   {
   make list of pairs (i,num[i])
   bucket sort pairs by num[i]
   bucket sort pairs by i (giving lists digit[i])
   bucket sort num by length
   i = max length
   L = empty
   while (i > 0)
   {
      concatenate num of length = i before start of L
      distribute into buckets by digits in position i
      coalesce by concatenating buckets in digits[i]
      i--
   }
   concatenate list of empty num at start of L
   return L
   }
```

The time for the first three bucket sorts is O(n+k). Each remaining pass takes time O(n_i) where n_i is the number of num having a digit in position i, so the total time for the while loop is O(n) again. Overall, the algorithm's total time is O(n + k), where n is the total length of all num and k is number of digits.

5.Solution:

Assume log n = k
There are n/k sub sequences and every sub sequence can be ordered in k! ways. So this makes $(k!)^{n/k}$ outputs
$(k!)^{n/k} <= 2^h$
Taking log on both sides
$h >= lg(k!)^{n/k}$
$>=(n/k) lg (k!)$
$>= (n/k)k lg k$
$= omega(n lg k)$
$= omega(n lg lg n)$ since k = log n

References used:

Analysis and Design of algorithms by Padma reddy
https://www.ics.uci.edu/~eppstein/161/960123.html
cyberzhg.gitbooks.io
http://cs.uno.edu/people/faculty/bill/k-way-merge-n-sort-ACM-SE-Regl-1993.pdf