**Task 1**

a)

Test A1 – generate n = 30000 random values from -35500 to 36600

Test A2 – generate n = 40000 random values from -35500 to 36600

Test A3 – generate n = 50000 random values from -35500 to 36600

Test A4 – generate n = 60000 random values from -13800 to 96800

Test A5 – generate n = 70000 random values from -13800 to 96800

Test A6 – generate n = 80000 random values from -13800 to 96800

Test B1 – generate n = 30000 unique, random values from -35500 to 36600

Test B2 – generate n = 40000 unique, random values from -35500 to 36600

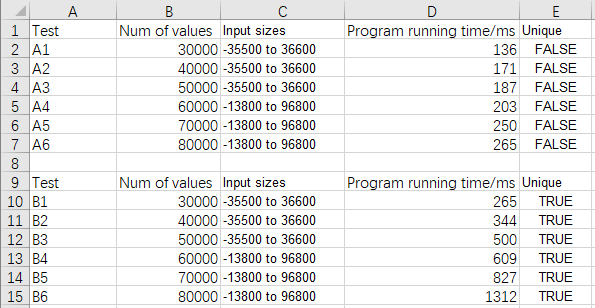
Test B3 – generate n = 50000 unique, random values from -35500 to 36600

Test B4 – generate n = 60000 unique, random values from -13800 to 96800

Test B5 – generate n = 70000 unique, random values from -13800 to 96800

Test B6 – generate n = 80000 unique, random values from -13800 to 96800

b)

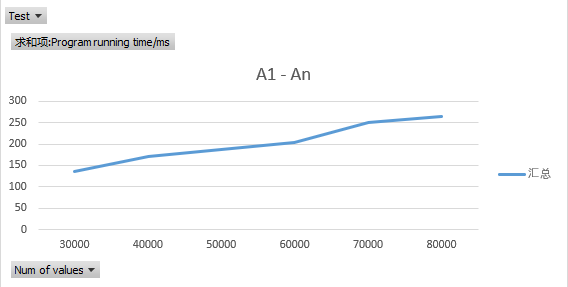


The Java application is in He Mingli A1 file.

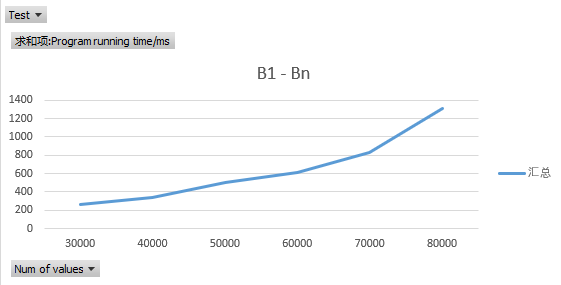
Print results are stored in Test result file.

c)

Graph #1 – the timing results for Test A1 – An (non-unique random values)



Graph #2 – the timing results for Test B1 – Bn (unique random values)



d)

Following the above results, we can clear to know that the large number of values cost more time to generate numbers. In addition, for A1 to An, these random numbers are only needed to generate once. But unique output needs to traverse the output in the existing list to determine whether it is repeated, so that we need to add loop function, that led to costed high time to algorithm. Furthermore, if n (Num of values) goes bigger, the time cost of unique outputs is much more that A1 – An.

Therefore, A1 – An Big-O characterization:

f(n) = n

B1 – Bn Big-O characterization:

f(n) = n^2

**Task 2**