CS 325 Project 1: Maximum Sum Subarray

Names and group member: Zhouxiang Meng, Jiamin Shen

Theoretical Run-time Analysis:

1. **Algorithm 1: Enumeration:**

   Pseudo code:

   ```
   MaxSubarray(A):
       For each pair(i,j) with 1≤ i < j ≤ n
           Compute a[i]+a[i+1]+...+a[j-1]+a[j]
           Keep max sum found so far
       Return max sum
   ```

   Running time:

   For the for loop, i and j is going n times so the out side loop have $O(n^2)$ time. The inside loop should have O(n) time for computation. Thus, the total time is $O(n^3)$ time.

2. **Algorithm 2: Better Enumeration:**

   Pseudo code:

   ```
   Max subarray(A):
       For i = 1 to n
           Sum = 0
           For j = i to n
               Sum = sum + a[j]
               Keep max sum found so far
       Return max sum
   ```

   Running time:

   This is same analysis with the first algorithm, the outside loop is $O(n^2)$ time. However, the inside loop only need O(1) time for calculation. Thus, the total time is $O(n^2)$ time.

3. **Algorithm 3: Divide and Conquer:**

   Pseudo code:

   ```
   Max subarray(A,l,r)
       Middle = (l+r)/2
       Maxsubarray (A,l,middle)
       Maxsubarray (A,middle+1,r)
       Return Max(Maxsubarray(A,l, middle), Maxsubarray (A,middle+1,r),
               (Maxsubarray(A,l, middle)+Maxsubarray (A,middle+1,r)))
   ```

   Running time:

   The deep of divide part is O(lg(n)) time, for the conquer part, the time should be O(n). Thus, the total time should be O(nlgn) (formula: T(n)=2T(n/2)+n)

4. **Algorithm 4: Linear-time:**

   Pseudo code:

   ```
   Maxsubarray(A)
       For i = 0 to n
           Find the max value of current max value + A[i] and the old max in
   ```

memory

Return the max sum

Running time:

The outside for loop is O(n) time, the inside calculation in for loop should be O(1). Thus the total time should be O(n) (linear) time.

Testing:

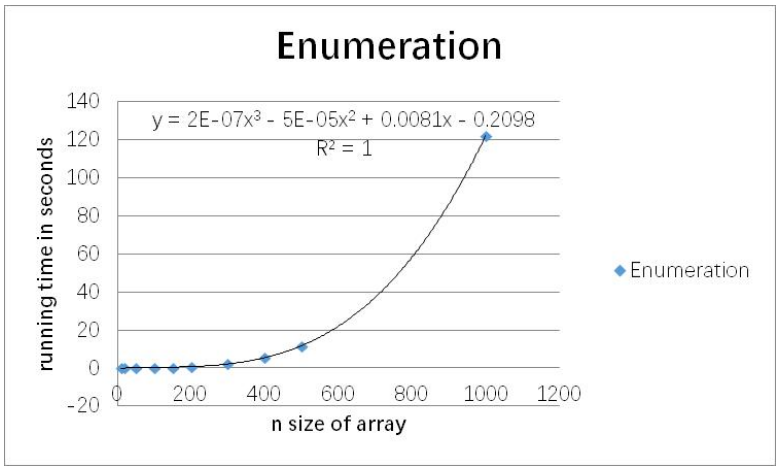
For each algorithm, we put the MSS_Problems.txt that we downloaded from caves in the same path of the .py files, and separately test the correctness of .py files. Comparing the output file with MSS_TestResults.txt file, we find that the result we get is same to the test result file. Thus, our algorithm is correct.

Experimental Analysis:

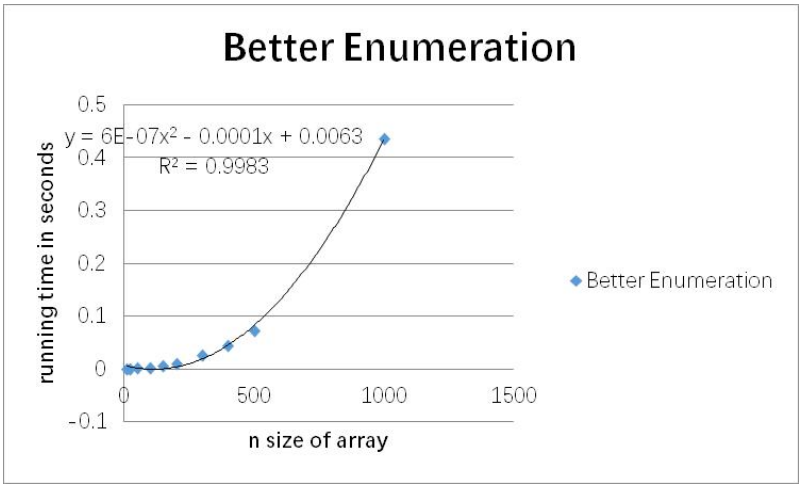
1. We test our algorithm in size n of 10,20,50,100,150,200,300,400,500,1000.

	10	20	50	100	150	200	300	400	500	1000
Enumeration	0.000132	0.000736	0.010304	0.084016	0.278584	0.638047	2.301069	5.606528	11.508116	122.263004
Better Enumeration	0.000037	0.000115	0.000594	0.002367	0.005091	0.009276	0.025993	0.042907	0.071711	0.435835
Divide and Conquer	0.000046	0.000102	0.000268	0.000528	0.001343	0.001296	0.001924	0.002754	0.00321	0.006423
Linear-time	0.000012	0.000032	0.000074	0.000107	0.000171	0.000179	0.000248	0.000384	0.000454	0.000745

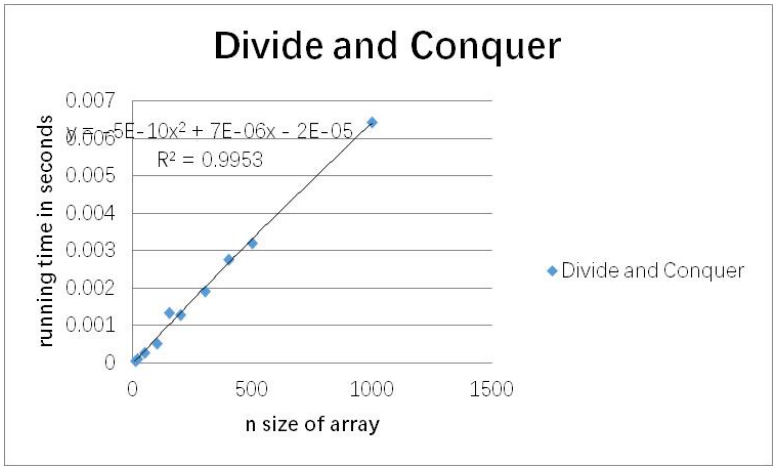
2.Enumeration:



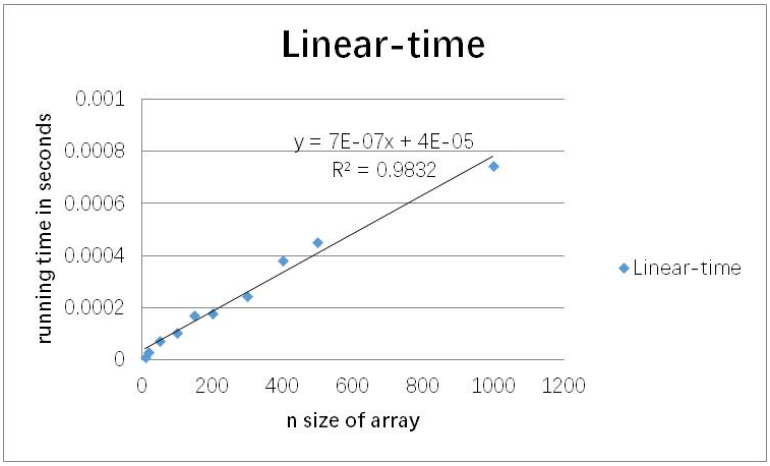
Better Enumeration:



Divide and Conquer:



Linear-time:



3. From the function in the graph in problem 2, we can see the similar function of 4 algorithms.

Enumeration:

The relationship of time and n is cubic(R^2=1). ($O(n^3)$ theoretical)

Better Enumeration:

The relationship of time and n is quadratic(R^2=0.9983).($O(n^2)$ theoretical)

Divide and Conquer:

The relationship of time and n is exponential(R^2=0.9953).(O(nlgn) theoretical)

Linear-time:

The relationship of time and n is linear(R^2=0.9832).(O(n) theoretical)

4. The experimental running time is almost same to the theoretical running time, the little differences might be caused by the test environment, or maybe depends on the random array, for example, the best case of liner time algorithm is always faster than worst case.

5. We can calculate the largest input of this test environment from the regression model in problem 2.

Enumeration:

For 5 second: the largest input is 348.377

For 10 second: the largest input is 430.923

For 1 minute: the largest input is 742.091

Better Enumeration:

For 5 second: the largest input is 2969.47

For 10 second: the largest input is 4165.38

For 1 minute: the largest input is 10083.2

Divide and Conquer:

For 5 second: the largest input is 92754.8996

For 10 second: the largest input is 134248.1504

For 1 minute: the largest input is 339339.4866

Linear-time:

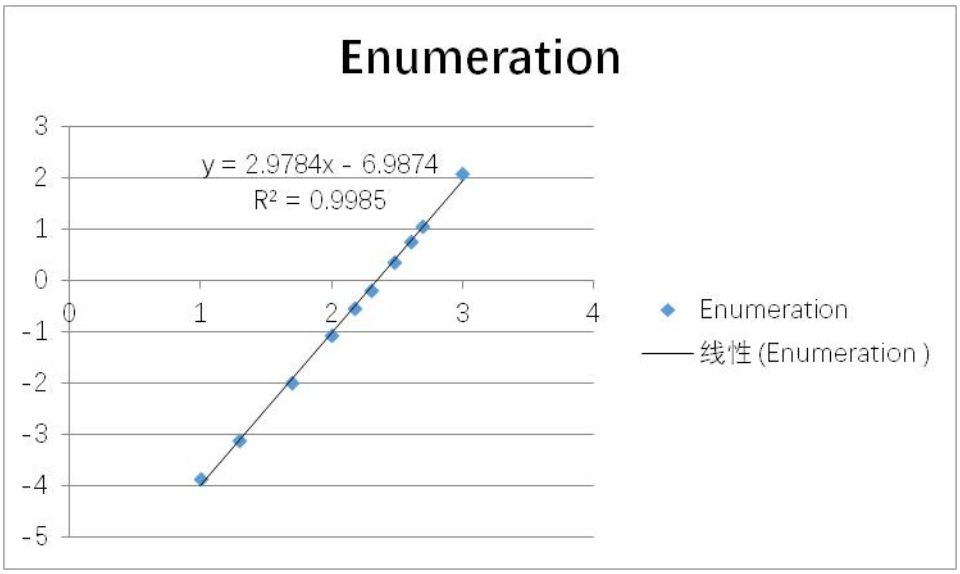
For 5 second: the largest input is 7142800

For 10 second: the largest input is 14285657.14

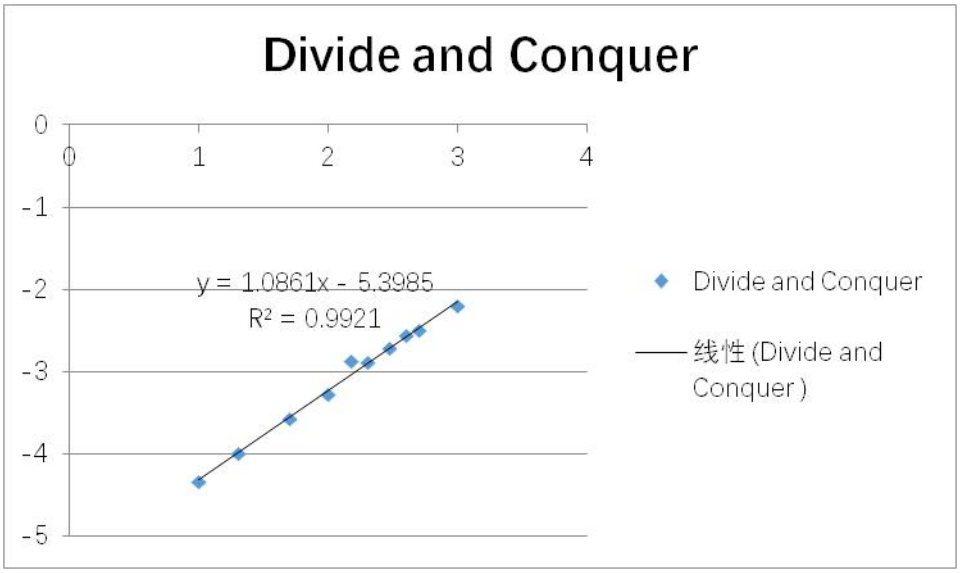
For 1 minute: the largest input is 85714228.57

6. From the value we get from our program, we can construct the log-log plot.

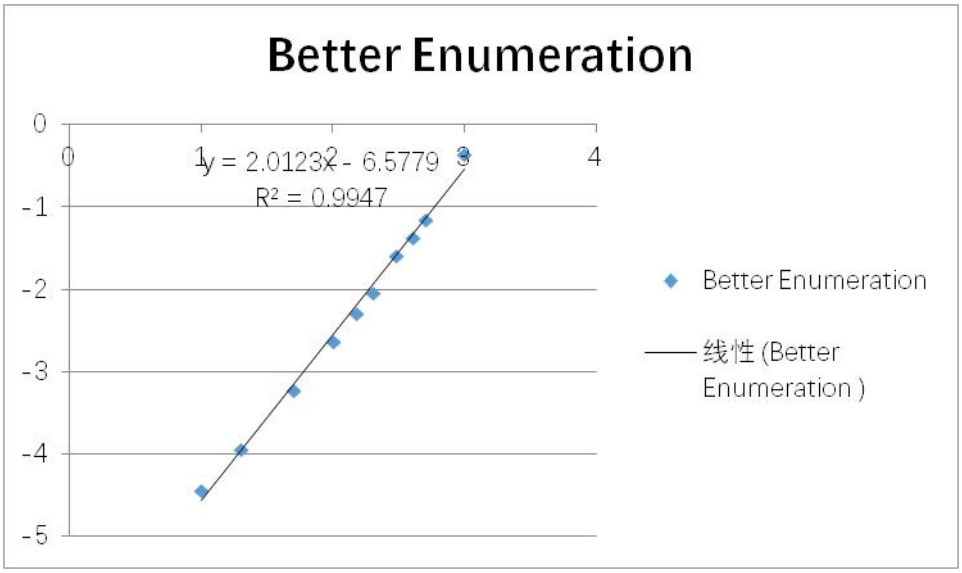
Enumeration:



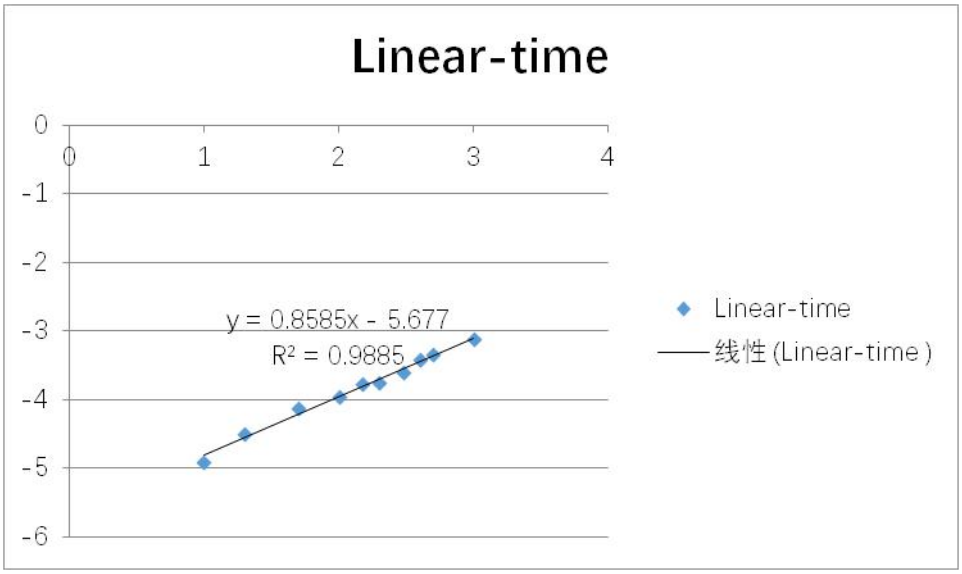
Better Enumeration:



Divide and Conquer:

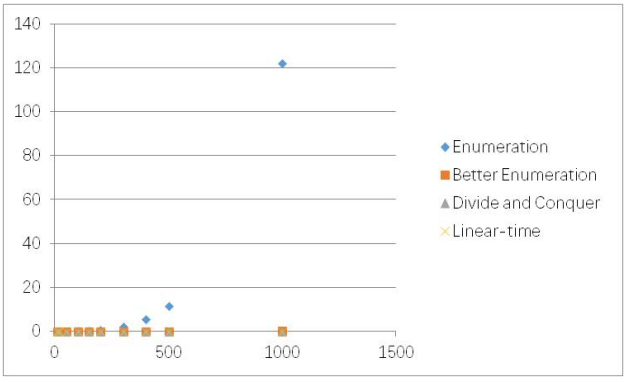


Linear-time:



7.

For the results of all four Algorithms together on a single graph, it is hard to see the difference of last 3 algorithm since cubic algorithm is dominate in the graph:



So, we decided put all result in log graph: