בalgorithm.h וalgorithm.cpp יש את החותמות ואת המתודות ריצה עבור זמני ריצה לינארי, ריבועי ובשלישית.
בex5.cpp יש את המיין שמקבל 2 משתנים seed וn, עבור הsrand וגודל המערך.

בקובץ makefile:

- יש הרצה לתכנית
- יש הרצה לprofiling

עבור הרצת n= 100

```
ariela@ariela-VirtualBox:~/Desktop/opSys_1/ex5$ make
clang++ -Wall -pg -O0 -c ex5.cpp -o ex5.o
clang++ -Wall -pg -O0 -c algorithm.cpp -o algorithm.o
clang++ -Wall -pg -O0 -o ex5 ex5.o algorithm.o
ariela@ariela-VirtualBox:~/Desktop/opSys_1/ex5$ ./ex5 42 100
Running of O(n) max sub array sum: 2075

Running of O(n^2) max sub array sum: 2075

Running of O(n^3) max sub array sum: 2075
```

הרצנו את ex5 עם הנתונים הפלט זה התוצאה של הסכום המקסימלי לכל אחד מזמני הריצה.

הרצנו make profiling ויצר קובץ חדש בשם analyst.txt עם זמני הריצה והקריאות.

```
Flat profile:

Each sample counts as 0.01 seconds.
 no time accumulated

  %   cumulative   self              self     total
 time   seconds   seconds    calls  Ts/call  Ts/call  name
  0.00      0.00     0.00    10300     0.00     0.00  int const& std::max<int>(int const&, int const&)
  0.00      0.00     0.00        1     0.00     0.00  run_algorithms(int*, int)
  0.00      0.00     0.00        1     0.00     0.00  max_subarray_sum_n(int*, int)
  0.00      0.00     0.00        1     0.00     0.00  max_subarray_sum_n2(int*, int)
  0.00      0.00     0.00        1     0.00     0.00  max_subarray_sum_n3(int*, int)
  0.00      0.00     0.00        1     0.00     0.00  generate_random_array(int, int)
```

נראה שעבור n קטן כזה זמני הריצה די דומים הקריאות עבור n^2 ועבור n^3 די דומים.

```
index % time    self  children    called     name
                0.00    0.00     200/10300       max_subarray_sum_n(int*, int) [10]
                0.00    0.00    5050/10300       max_subarray_sum_n3(int*, int) [12]
                0.00    0.00    5050/10300       max_subarray_sum_n2(int*, int) [11]
[8]      0.0    0.00    0.00   10300         int const& std::max<int>(int const&, int const&) [8]
-----------------------------------------------
                0.00    0.00       1/1           main [6]
[9]      0.0    0.00    0.00       1         run_algorithms(int*, int) [9]
                0.00    0.00       1/1           max_subarray_sum_n(int*, int) [10]
                0.00    0.00       1/1           max_subarray_sum_n2(int*, int) [11]
                0.00    0.00       1/1           max_subarray_sum_n3(int*, int) [12]
-----------------------------------------------
                0.00    0.00       1/1           run_algorithms(int*, int) [9]
[10]     0.0    0.00    0.00       1         max_subarray_sum_n(int*, int) [10]
                0.00    0.00     200/10300       int const& std::max<int>(int const&, int const&) [8]
-----------------------------------------------
                0.00    0.00       1/1           run_algorithms(int*, int) [9]
[11]     0.0    0.00    0.00       1         max_subarray_sum_n2(int*, int) [11]
                0.00    0.00    5050/10300       int const& std::max<int>(int const&, int const&) [8]
-----------------------------------------------
                0.00    0.00       1/1           run_algorithms(int*, int) [9]
[12]     0.0    0.00    0.00       1         max_subarray_sum_n3(int*, int) [12]
                0.00    0.00    5050/10300       int const& std::max<int>(int const&, int const&) [8]
-----------------------------------------------
                0.00    0.00       1/1           main [6]
[13]     0.0    0.00    0.00       1         generate_random_array(int, int) [13]
-----------------------------------------------
```

שאלה 5

עבור n=100 הפונקציה הלניארית קראה ל200 השוואות.

הפונקציה הרבועית ובחזקת 3 עשו 5050 השוואות.

(אשאיר את הקובץ הזה בשם profile_n_100.txt למקרה הצורך)

הרצה עבור n=1000

```
ariela@ariela-VirtualBox:~/Desktop/opSys_1/ex5$ ./ex5 30 1000
Running of O(n) max sub array sum: 25517

Running of O(n^2) max sub array sum: 25517

Running of O(n^3) max sub array sum: 25517

ariela@ariela-VirtualBox:~/Desktop/opSys_1/ex5$ make profile
gprof ex5 gmon.out > analysis.txt
Profiling data saved to analysis.txt
```

בקובץ analyst נוכל לראות את השינוי בקריאות:

```
Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  ms/call  ms/call  name
100.00      0.55     0.55        1   550.00   550.00  max_subarray_sum_n3(int*, int)
  0.00      0.55     0.00  1003000     0.00     0.00  int const& std::max<int>(int const&, int const&)
  0.00      0.55     0.00        1     0.00   550.00  run_algorithms(int*, int)
  0.00      0.55     0.00        1     0.00     0.00  max_subarray_sum_n(int*, int)
  0.00      0.55     0.00        1     0.00     0.00  max_subarray_sum_n2(int*, int)
  0.00      0.55     0.00        1     0.00     0.00  generate_random_array(int, int)

 %         the percentage of the total running time of the
```

ניתן לראות שאכן רוב הריצה היה בפונקיה עם זמן ריצה o(n^3)

```
granularity: each sample hit covers 4 byte(s) for 1.82% of 0.55 seconds

index % time    self  children    called     name
                                                 <spontaneous>
[1]     100.0    0.00    0.55                 main [1]
                 0.00    0.55       1/1           run_algorithms(int*, int) [2]
                 0.00    0.00       1/1           generate_random_array(int, int) [13]
-----------------------------------------------
                 0.00    0.55       1/1           main [1]
[2]     100.0    0.00    0.55       1         run_algorithms(int*, int) [2]
                 0.55    0.00       1/1           max_subarray_sum_n3(int*, int) [3]
                 0.00    0.00       1/1           max_subarray_sum_n(int*, int) [11]
                 0.00    0.00       1/1           max_subarray_sum_n2(int*, int) [12]
-----------------------------------------------
                 0.55    0.00       1/1           run_algorithms(int*, int) [2]
[3]     100.0    0.55    0.00       1         max_subarray_sum_n3(int*, int) [3]
                 0.00    0.00  500500/1003000     int const& std::max<int>(int const&, int const&) [10]
-----------------------------------------------
                 0.00    0.00    2000/1003000     max_subarray_sum_n(int*, int) [11]
                 0.00    0.00  500500/1003000     max_subarray_sum_n3(int*, int) [3]
                 0.00    0.00  500500/1003000     max_subarray_sum_n2(int*, int) [12]
[10]      0.0    0.00    0.00 1003000         int const& std::max<int>(int const&, int const&) [10]
-----------------------------------------------
                 0.00    0.00       1/1           run_algorithms(int*, int) [2]
[11]      0.0    0.00    0.00       1         max_subarray_sum_n(int*, int) [11]
                 0.00    0.00    2000/1003000     int const& std::max<int>(int const&, int const&) [10]
-----------------------------------------------
                 0.00    0.00       1/1           run_algorithms(int*, int) [2]
[12]      0.0    0.00    0.00       1         max_subarray_sum_n2(int*, int) [12]
                 0.00    0.00  500500/1003000     int const& std::max<int>(int const&, int const&) [10]
-----------------------------------------------
                 0.00    0.00       1/1           main [1]
[13]      0.0    0.00    0.00       1         generate_random_array(int, int) [13]
-----------------------------------------------
```

אחנו רואים שבפונקציה הלניארית יש 2000 קריאות ובריבועית ובחזקת 3 יש את רותן כמות קריאות לmax אבל עדיין רוב הריצה בילינו ב(o(n^3.

(גם שמרתי בתור קובץ profile_n_1000.txt).

עבור n=10000 (שרפתם לי 15 דקות מהחיים)

```
Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls   s/call   s/call  name
 99.71    529.71   529.71        1   529.71   530.12  max_subarray_sum_n3(int*, int)
  0.16    530.54     0.83 100030000     0.00     0.00  int const& std::max<int>(int const&, int const&)
  0.11    531.15     0.61        1     0.61     1.02  max_subarray_sum_n2(int*, int)
  0.02    531.25     0.10                             _init
  0.00    531.25     0.00        1     0.00   531.15  run_algorithms(int*, int)
  0.00    531.25     0.00        1     0.00     0.00  max_subarray_sum_n(int*, int)
  0.00    531.25     0.00        1     0.00     0.00  generate_random_array(int, int)

 %         the percentage of the total running time of the
time       program used by this function.
```

זמני הריצה משתנים ממש, בקושי בילינו ב(O(n לעומת (o(n^2 שם הוא היה כשניה לעומת ב(o(n^3 ששם הוא בילה 530 שניות.
מבחינתן אחוזים ב(o(n^3 בילינו 99.71% מהזמן.

```
granularity: each sample hit covers 4 byte(s) for 0.00% of 531.25 seconds

index % time    self  children    called     name
                                                 <spontaneous>
[1]    100.0    0.00  531.15                 main [1]
                0.00  531.15       1/1           run_algorithms(int*, int) [2]
                0.00    0.00       1/1           generate_random_array(int, int) [14]
-----------------------------------------------
                0.00  531.15       1/1           main [1]
[2]    100.0    0.00  531.15       1         run_algorithms(int*, int) [2]
              529.71    0.41       1/1           max_subarray_sum_n3(int*, int) [3]
                0.61    0.41       1/1           max_subarray_sum_n2(int*, int) [4]
                0.00    0.00       1/1           max_subarray_sum_n(int*, int) [7]
-----------------------------------------------
              529.71    0.41       1/1           run_algorithms(int*, int) [2]
[3]     99.8  529.71    0.41       1         max_subarray_sum_n3(int*, int) [3]
                0.41    0.00 50005000/100030000     int const& std::max<int>(int const&, int const&) [5]
-----------------------------------------------
                0.61    0.41       1/1           run_algorithms(int*, int) [2]
[4]      0.2    0.61    0.41       1         max_subarray_sum_n2(int*, int) [4]
                0.41    0.00 50005000/100030000     int const& std::max<int>(int const&, int const&) [5]
-----------------------------------------------
                0.00    0.00   20000/100030000     max_subarray_sum_n(int*, int) [7]
                0.41    0.00 50005000/100030000     max_subarray_sum_n3(int*, int) [3]
                0.41    0.00 50005000/100030000     max_subarray_sum_n2(int*, int) [4]
[5]      0.2    0.83    0.00 100030000         int const& std::max<int>(int const&, int const&) [5]
-----------------------------------------------
                                                 <spontaneous>
[6]      0.0    0.10    0.00                 _init [6]
-----------------------------------------------
                0.00    0.00       1/1           run_algorithms(int*, int) [2]
[7]      0.0    0.00    0.00       1         max_subarray_sum_n(int*, int) [7]
                0.00    0.00   20000/100030000     int const& std::max<int>(int const&, int const&) [5]
-----------------------------------------------
                0.00    0.00       1/1           main [1]
[14]     0.0    0.00    0.00       1         generate_random_array(int, int) [14]
-----------------------------------------------
```

למרות שכמות ההשוואות כמעט זהות ב (o(n^2 וב(O(n^3 כמות הסכומים שאחננו בודקים הם שונים לכן יש פער כל כך גדול בזמני ריצה.