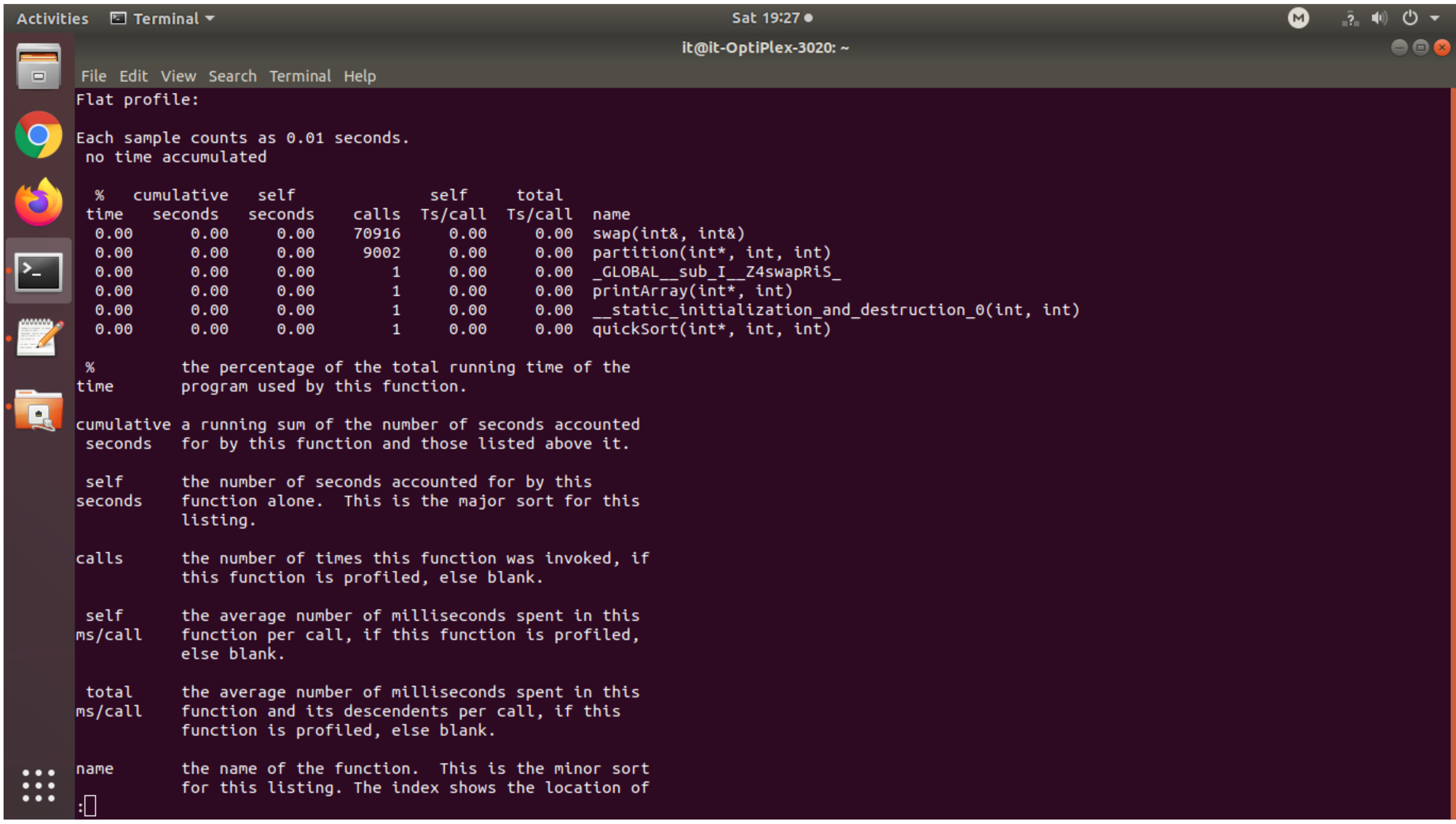
PRN: 2020BTEIT00006

Gprof Output for the Quicksort Algorithm

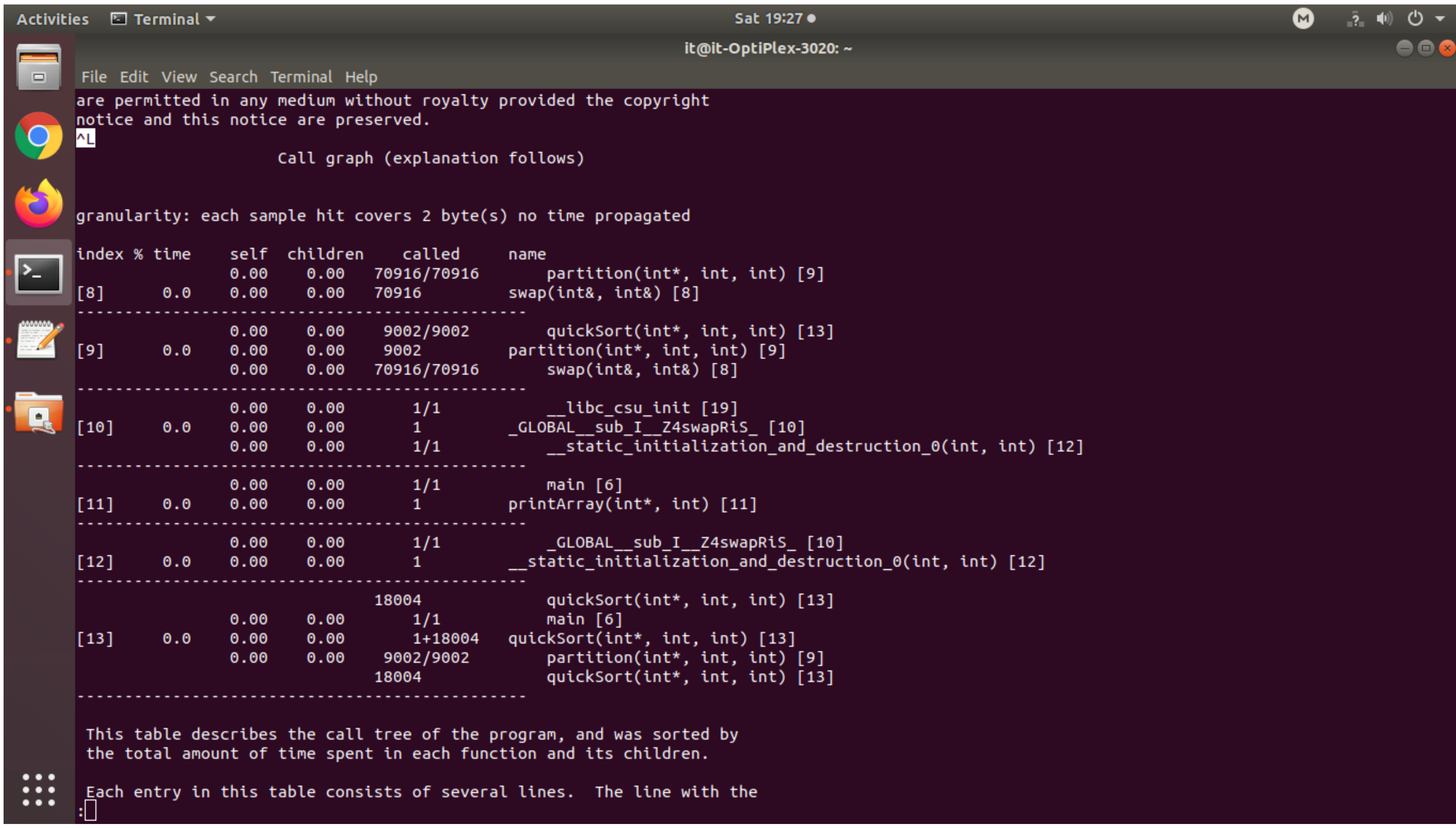
The value for the size of the array is 10000 i.e n = 10000.

Flat Profile: Below is the flat profile table for the quicksort algorithm.



```
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    70916     0.00     0.00  swap(int&, int&)
  0.00      0.00     0.00     9002     0.00     0.00  partition(int*, int, int)
  0.00      0.00     0.00        1     0.00     0.00  _GLOBAL__sub_I__Z4swapRiS_
  0.00      0.00     0.00        1     0.00     0.00  printArray(int*, int)
  0.00      0.00     0.00        1     0.00     0.00  __static_initialization_and_destruction_0(int, int)
  0.00      0.00     0.00        1     0.00     0.00  quickSort(int*, int, int)

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

cumulative a running sum of the number of seconds accounted
 seconds   for by this function and those listed above it.

 self      the number of seconds accounted for by this
seconds    function alone.  This is the major sort for this
           listing.

calls      the number of times this function was invoked, if
           this function is profiled, else blank.

 self      the average number of milliseconds spent in this
ms/call    function per call, if this function is profiled,
           else blank.

 total     the average number of milliseconds spent in this
ms/call    function and its descendents per call, if this
           function is profiled, else blank.

name       the name of the function.  This is the minor sort
           for this listing. The index shows the location of
:
```

Graph Explanation: Below is the graph table for the quicksort algorithm



```
are permitted in any medium without royalty provided the copyright
notice and this notice are preserved.
^L
                     Call graph (explanation follows)


granularity: each sample hit covers 2 byte(s) no time propagated

index % time    self  children    called     name
                0.00    0.00  70916/70916      partition(int*, int, int) [9]
[8]      0.0    0.00    0.00  70916          swap(int&, int&) [8]
-----------------------------------------------
                0.00    0.00    9002/9002      quickSort(int*, int, int) [13]
[9]      0.0    0.00    0.00    9002         partition(int*, int, int) [9]
                0.00    0.00  70916/70916      swap(int&, int&) [8]
-----------------------------------------------
                0.00    0.00       1/1           __libc_csu_init [19]
[10]     0.0    0.00    0.00       1         _GLOBAL__sub_I__Z4swapRiS_ [10]
                0.00    0.00       1/1         __static_initialization_and_destruction_0(int, int) [12]
-----------------------------------------------
                0.00    0.00       1/1         main [6]
[11]     0.0    0.00    0.00       1         printArray(int*, int) [11]
-----------------------------------------------
                0.00    0.00       1/1          _GLOBAL__sub_I__Z4swapRiS_ [10]
[12]     0.0    0.00    0.00       1         __static_initialization_and_destruction_0(int, int) [12]
-----------------------------------------------
                              18004            quickSort(int*, int, int) [13]
                0.00    0.00       1/1         main [6]
[13]     0.0    0.00    0.00       1+18004   quickSort(int*, int, int) [13]
                0.00    0.00    9002/9002      partition(int*, int, int) [9]
                              18004            quickSort(int*, int, int) [13]
-----------------------------------------------

 This table describes the call tree of the program, and was sorted by
 the total amount of time spent in each function and its children.

 Each entry in this table consists of several lines.  The line with the
:
```