Problem 1b. Enter the Time and compute the Ratio of Times to two decimal places (x.xx)

Graph Size	Time for Computing Spanning Tree	Ratio of Time: Size 2N/Size N
1,000	0.07839	No ratio for first graph size
2,000	0.16485	2.10294
4,000	0.34483	2.09178
8,000	0.76256	2.2114
16,000	1.59328	2.08938
32,000	3.48097	2.18478
64,000	7.60396	2.18443
128,000	16.69047	2.19497

Approximate the complexity class for the spanning_tree function based on the data above. Briefly explain your reasoning.

Answer: O(N log N) This is the signature for O(N Log N): slightly bigger than 2.

Problem 2b. Answer each of the following question based on the profiles produced when running spanning_tree : use the ncalls information for parts 2 and 3; use the tottime information for parts 1 and 4.

1) What function/method takes the most tottime to execute?

Answer: sorted

2) What non-built in function/method is called the most times?

Answer: __getitem__

3) What method defined in graph.py is called the most times?

Answer: __getitem__

4) What percent of the entire execution time is spent in the 5 functions with the most tottime?

Answer: 78.27%