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	0.087	No ratio for first graph size
2,000	0.202	2.32
4,000	0.404	2
8,000	0.872	2.15
16,000	1.922	2.19
32,000	4.109	2.14
64,000	8.688	2.11
128,000	18.316	2.11

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

Answer:

As the number of nodes doubles, the time doubles. This implies a linear relationship which is represented by O(n)

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: {built-in method builtins.sorted}

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

Answer: graph.py:23(__getitem__)

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

Answer: graph.py:23(__getitem__)

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

Answer: 75.56%