

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	0.00004	No ratio for first graph size
2,000	0.00004	1
4,000	0.00004	1
8,000	0.00004	1
16,000	0.00004	1
32,000	0.00004	1
64,000	0.00004	1
128,000	0.00005	0.8

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

Answer: If each of these operations for `spanning_tree` is about  $O(1)$  and we do  $N$  of each, the complexity class is  $O(N)$ .

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: `equivalence.py:28(_compress_to_root)`

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: 73%