Sample output from my solution to Problem #1:
(yours should match the format: the times depend on your machine's speed).

```
Spanning Tree of size 1000
Analysis of 5 timings
avg = 0.079   min = 0.076  max = 0.084  span = 9.7%

   Time Ranges
7.59e-02<>7.66e-02[ 40.0%]|**************************************************
7.66e-02<>7.74e-02[  0.0%]|
7.74e-02<>7.82e-02[  0.0%]|
7.82e-02<>7.89e-02[  0.0%]|
7.89e-02<>7.97e-02[ 20.0%]|*************************A
7.97e-02<>8.05e-02[  0.0%]|
8.05e-02<>8.12e-02[ 20.0%]|*************************
8.12e-02<>8.20e-02[  0.0%]|
8.20e-02<>8.28e-02[  0.0%]|
8.28e-02<>8.35e-02[  0.0%]|
8.35e-02<>8.43e-02[ 20.0%]|*************************

Spanning Tree of size 2000
Analysis of 5 timings
avg = 0.166   min = 0.161  max = 0.172  span = 6.1%

   Time Ranges
1.61e-01<>1.62e-01[ 20.0%]|*************************
1.62e-01<>1.63e-01[  0.0%]|
1.63e-01<>1.64e-01[  0.0%]|
1.64e-01<>1.65e-01[ 40.0%]|**************************************************
1.65e-01<>1.66e-01[  0.0%]|A
1.66e-01<>1.68e-01[  0.0%]|
1.68e-01<>1.69e-01[ 20.0%]|*************************
1.69e-01<>1.70e-01[  0.0%]|
1.70e-01<>1.71e-01[  0.0%]|
1.71e-01<>1.72e-01[  0.0%]|
1.72e-01<>1.73e-01[ 20.0%]|*************************

Spanning Tree of size 4000
Analysis of 5 timings
avg = 0.351   min = 0.342  max = 0.359  span = 4.9%

   Time Ranges
3.42e-01<>3.44e-01[ 20.0%]|**************************************************
3.44e-01<>3.45e-01[  0.0%]|
3.45e-01<>3.47e-01[  0.0%]|
3.47e-01<>3.49e-01[ 20.0%]|**************************************************
3.49e-01<>3.50e-01[ 20.0%]|**************************************************
3.50e-01<>3.52e-01[  0.0%]|A
3.52e-01<>3.54e-01[  0.0%]|
3.54e-01<>3.55e-01[ 20.0%]|**************************************************
3.55e-01<>3.57e-01[  0.0%]|
3.57e-01<>3.59e-01[  0.0%]|
3.59e-01<>3.61e-01[ 20.0%]|**************************************************

Spanning Tree of size 8000
Analysis of 5 timings
avg = 0.733   min = 0.723  max = 0.749  span = 3.5%

   Time Ranges
7.23e-01<>7.26e-01[ 40.0%]|**************************************************
```

```
7.26e-01<>7.28e-01[  0.0%]|
7.28e-01<>7.31e-01[ 20.0%]|*************************
7.31e-01<>7.34e-01[  0.0%]|A
7.34e-01<>7.36e-01[  0.0%]|
7.36e-01<>7.39e-01[  0.0%]|
7.39e-01<>7.41e-01[ 20.0%]|*************************
7.41e-01<>7.44e-01[  0.0%]|
7.44e-01<>7.46e-01[  0.0%]|
7.46e-01<>7.49e-01[  0.0%]|
7.49e-01<>7.51e-01[ 20.0%]|*************************

Spanning Tree of size 16000
Analysis of 5 timings
avg = 1.582   min = 1.544  max = 1.642  span = 6.2%

   Time Ranges
1.54e+00<>1.55e+00[ 20.0%]|*************************
1.55e+00<>1.56e+00[  0.0%]|
1.56e+00<>1.57e+00[ 40.0%]|**************************************************
1.57e+00<>1.58e+00[  0.0%]|A
1.58e+00<>1.59e+00[ 20.0%]|*************************
1.59e+00<>1.60e+00[  0.0%]|
1.60e+00<>1.61e+00[  0.0%]|
1.61e+00<>1.62e+00[  0.0%]|
1.62e+00<>1.63e+00[  0.0%]|
1.63e+00<>1.64e+00[  0.0%]|
1.64e+00<>1.65e+00[ 20.0%]|*************************

Spanning Tree of size 32000
Analysis of 5 timings
avg = 3.432   min = 3.377  max = 3.514  span = 4.0%

   Time Ranges
3.38e+00<>3.39e+00[ 20.0%]|**************************************************
3.39e+00<>3.40e+00[ 20.0%]|**************************************************
3.40e+00<>3.42e+00[ 20.0%]|**************************************************
3.42e+00<>3.43e+00[  0.0%]|A
3.43e+00<>3.45e+00[  0.0%]|
3.45e+00<>3.46e+00[  0.0%]|
3.46e+00<>3.47e+00[ 20.0%]|**************************************************
3.47e+00<>3.49e+00[  0.0%]|
3.49e+00<>3.50e+00[  0.0%]|
3.50e+00<>3.51e+00[  0.0%]|
3.51e+00<>3.53e+00[ 20.0%]|**************************************************

Spanning Tree of size 64000
Analysis of 5 timings
avg = 7.470   min = 7.358  max = 7.680  span = 4.3%

   Time Ranges
7.36e+00<>7.39e+00[ 40.0%]|**************************************************
7.39e+00<>7.42e+00[  0.0%]|
7.42e+00<>7.45e+00[  0.0%]|
7.45e+00<>7.49e+00[ 40.0%]|**************************************************A
7.49e+00<>7.52e+00[  0.0%]|
7.52e+00<>7.55e+00[  0.0%]|
7.55e+00<>7.58e+00[  0.0%]|
7.58e+00<>7.62e+00[  0.0%]|
7.62e+00<>7.65e+00[  0.0%]|
7.65e+00<>7.68e+00[  0.0%]|
7.68e+00<>7.71e+00[ 20.0%]|*************************
```

```
Spanning Tree of size 128000
Analysis of 5 timings
avg = 16.294   min = 16.071  max = 16.665  span = 3.6%

   Time Ranges
1.61e+01<>1.61e+01[ 20.0%]|**************************************************
1.61e+01<>1.62e+01[ 20.0%]|**************************************************
1.62e+01<>1.62e+01[ 20.0%]|**************************************************
1.62e+01<>1.63e+01[  0.0%]|A
1.63e+01<>1.64e+01[  0.0%]|
1.64e+01<>1.64e+01[ 20.0%]|**************************************************
1.64e+01<>1.65e+01[  0.0%]|
1.65e+01<>1.65e+01[  0.0%]|
1.65e+01<>1.66e+01[  0.0%]|
1.66e+01<>1.67e+01[  0.0%]|
1.67e+01<>1.67e+01[ 20.0%]|**************************************************
```

Sample output from my solution to Problem #2:
(yours should match the format: the times/counts depend on your machine's speed and the random graph created).

```
Tue Mar 14 11:53:24 2017    profile5K

         773841 function calls (768840 primitive calls) in 0.527 seconds

   Ordered by: call count

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
   199461    0.010    0.000    0.010    0.000 {built-in method builtins.len}
   104730    0.085    0.000    0.162    0.000 graph.py:23(__getitem__)
   103632    0.063    0.000    0.063    0.000 equivalence.py:28(_compress_to_root)
    99731    0.047    0.000    0.289    0.000 graph_goody.py:26(<genexpr>)
    99731    0.056    0.000    0.236    0.000 graph.py:125(__iter__)
    99730    0.072    0.000    0.077    0.000 graph.py:12(legal_tuple)
    46817    0.021    0.000    0.078    0.000 equivalence.py:60(in_same_class)
   5002/1    0.137    0.000    0.407    0.407 {built-in method builtins.sorted}
     5000    0.002    0.000    0.002    0.000 equivalence.py:19(add_singleton)
     4999    0.005    0.000    0.011    0.000 equivalence.py:68(merge_classes_containing)
     4999    0.000    0.000    0.000    0.000 {method 'add' of 'set' objects}
        2    0.000    0.000    0.000    0.000 graph.py:73(all_nodes)
        2    0.000    0.000    0.000    0.000 {method 'keys' of 'dict' objects}
        1    0.021    0.021    0.521    0.521 graph_goody.py:24(spanning_tree)
        1    0.001    0.001    0.003    0.003 equivalence.py:8(__init__)
        1    0.006    0.006    0.526    0.526 <string>:1(<module>)
        1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler' objects}
        1    0.000    0.000    0.527    0.527 {built-in method builtins.exec}


Tue Mar 14 11:53:26 2017    profile10K

         1561667 function calls (1551666 primitive calls) in 1.114 seconds

   Ordered by: internal time

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
  10002/1    0.309    0.000    0.850    0.850 {built-in method builtins.sorted}
   209760    0.172    0.000    0.326    0.000 graph.py:23(__getitem__)
   199760    0.143    0.000    0.154    0.000 graph.py:12(legal_tuple)
   215396    0.133    0.000    0.133    0.000 equivalence.py:28(_compress_to_root)
   199761    0.109    0.000    0.473    0.000 graph.py:125(__iter__)
   199761    0.096    0.000    0.580    0.000 graph_goody.py:26(<genexpr>)
        1    0.050    0.050    1.095    1.095 graph_goody.py:24(spanning_tree)
    97699    0.044    0.000    0.166    0.000 equivalence.py:60(in_same_class)
   399521    0.021    0.000    0.021    0.000 {built-in method builtins.len}
        1    0.019    0.019    1.114    1.114 <string>:1(<module>)
     9999    0.011    0.000    0.021    0.000 equivalence.py:68(merge_classes_containing)
    10000    0.003    0.000    0.003    0.000 equivalence.py:19(add_singleton)
        1    0.002    0.002    0.005    0.005 equivalence.py:8(__init__)
     9999    0.001    0.000    0.001    0.000 {method 'add' of 'set' objects}
        2    0.001    0.000    0.001    0.000 graph.py:73(all_nodes)
        1    0.000    0.000    1.114    1.114 {built-in method builtins.exec}
        2    0.000    0.000    0.000    0.000 {method 'keys' of 'dict' objects}
        1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler' objects}
```