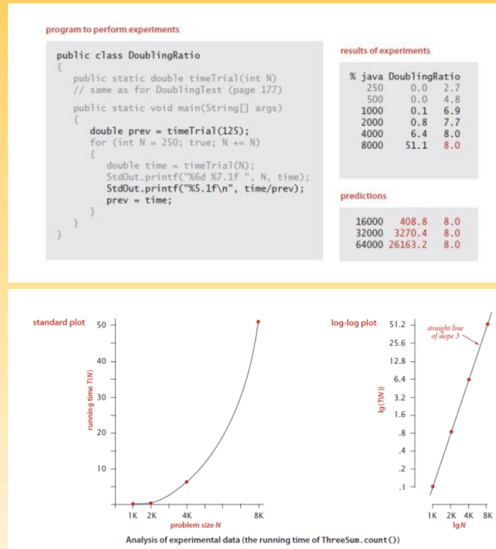
Doubling Ratio

EMBRY-RIDDLE Aeronautical University

The following experiment can be used to predict whether an algorithm that is believed to have polynomial grown of some amount, N^b, the doubling ratio experiment can help approximate b.

$$ratio = \frac{time_{cur}}{time_{prev}}$$

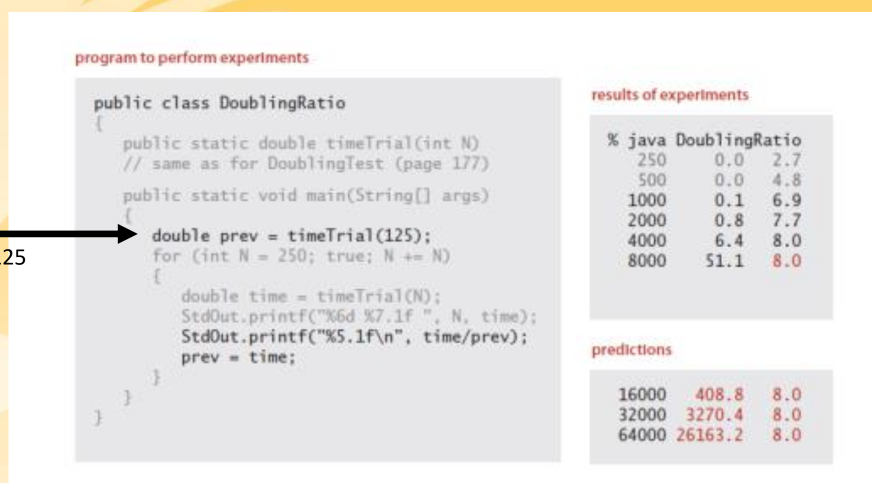


1

Doubling Ratio

EMBRY-RIDDLE Aeronautical University

Runs algorithm with initial N=125
Returns runtime


program to perform experiments

```
public class DoublingRatio
{
   public static double timeTrial(int N)
   // same as for DoublingTest (page 177)
   public static void main(String[] args)
   {
      double prev = timeTrial(125);
      for (int N = 250; true; N += N)
      {
         double time = timeTrial(N);
         StdOut.printf("%6d %7.1f ", N, time);
         StdOut.printf("%5.1f\n", time/prev);
         prev = time;
      }
   }
}
```

results of experiments

```
% java DoublingRatio
  250    0.0   2.7
  500    0.0   4.8
 1000    0.1   6.9
 2000    0.8   7.7
 4000    6.4   8.0
 8000   51.1   8.0
```

predictions

```
16000   408.8   8.0
32000  3270.4   8.0
64000 26163.2   8.0
```


2

Doubling Ratio

Loop from N=250 and doubles itself with each iteration (N+=N)

For each iteration… algorithm is re-ran and the ratio between the new and old runtimes is computed.

program to perform experiments

```
public class DoublingRatio
{
   public static double timeTrial(int N)
   // same as for DoublingTest (page 177)

   public static void main(String[] args)
   {
      double prev = timeTrial(125);
      for (int N = 250; true; N += N)
      {
         double time = timeTrial(N);
         StdOut.printf("%6d %7.1f ", N, time);
         StdOut.printf("%5.1f\n", time/prev);
         prev = time;
      }
   }
}
```

results of experiments

% java DoublingRatio

250	0.0	2.7
500	0.0	4.8
1000	0.1	6.9
2000	0.8	7.7
4000	6.4	8.0
8000	51.1	8.0

predictions

16000	408.8	8.0
32000	3270.4	8.0
64000	26163.2	8.0

Fast at first, but eventually, it will slow down and then converge.

Doubling Ratio

EMBRY-RIDDLE
Aeronautical University

$$ratio = \frac{time_{cur}}{time_{prev}}$$

- The ratio will converge to a ratio of 2^b if the algorithm's true order is N^b, where b is some unknown polynomial.
- e.g. Assume some unknown algorithm is $\sim N^2$, but we don't know that because we don't understand the code, but we can run an experiment.
  - $N = 16, T_1 = 268$
  - $N = 32, T_2 = 1027$
  - Knowing the doubling ratio and having doubled our inputs, we observe a convergence of ratio at
    - $Ratio = \frac{T2}{T1} = \frac{1027}{268} = 3.83$
    - $b = log_2(3.83) = 1.93734$ (round up to 2)
  - Therefore, order is $\sim N^2$

program to perform experiments

```
public class DoublingRatio
{
   public static double timeTrial(int N)
   // same as for DoublingTest (page 177)

   public static void main(String[] args)
   {
      double prev = timeTrial(125);
      for (int N = 250; true; N += N)
      {
         double time = timeTrial(N);
         StdOut.printf("%6d %7.1f ", N, time);
         StdOut.printf("%5.1f\n", time/prev);
         prev = time;
      }
   }
}
```

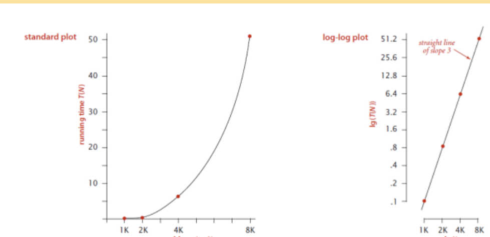
results of experiments

% java DoublingRatio

250	0.0	2.7
500	0.0	4.8
1000	0.1	6.9
2000	0.8	7.7
4000	6.4	8.0
8000	51.1	8.0

predictions

16000	408.8	8.0
32000	3270.4	8.0
64000	26163.2	8.0



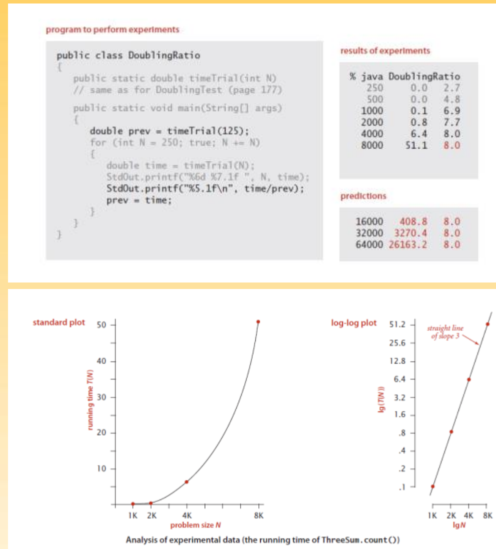
Analysis of experimental data (the running time of ThreeSum.count())

Doubling Ratio

EMBRY-RIDDLE Aeronautical University

$$ratio = \frac{time_{cur}}{time_{prev}}$$

- The ratio will converge to a ratio of 2^b if the algorithm's true order is N^b, where b is some unknown polynomial.
- By forcing the size to be double, we can observe how the result scales.

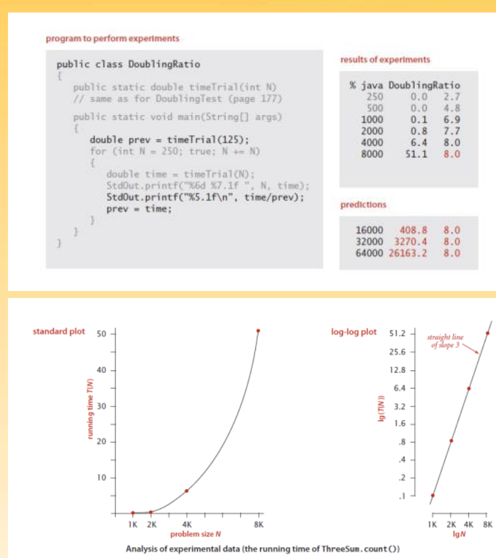


Doubling Ratio

EMBRY-RIDDLE Aeronautical University

$$ratio = \frac{time_{cur}}{time_{prev}}$$

- The ratio will converge to a ratio of 2^b if the algorithm's true order is N^b, where b is some unknown polynomial.
- By forcing the size to be double, we can observe how the result scales.
- Once we have our convergence, we can solve for b.
  - e.g. 2^b = 8 after convergence in the ThreeSum
  - Which means that $b = 3$ AND the algorithm's order of growth is N^3.



Doubling Ratio

EMBRY-RIDDLE Aeronautical University

Proposition C. (Doubling ratio) If $T(N) \sim a N^b \lg N$ then $T(2N)/T(N) \sim 2^b$.

Proof: Immediate from the following calculation:

$$\begin{aligned} T(2N)/T(N) &= a(2N)^b \lg(2N) / a N^b \lg N \\ &= 2^b (1 + \lg 2 / \lg N) \\ &\sim 2^b \end{aligned}$$

program to perform experiments

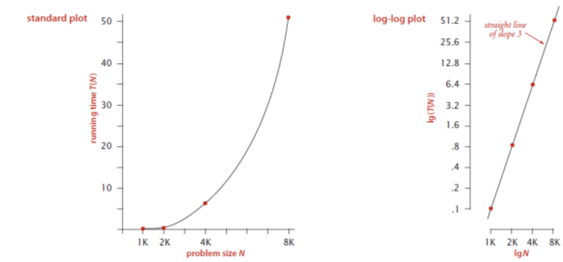
```
public class DoublingRatio
{
   public static double timeTrial(int N)
   // same as for DoublingTest (page 177)

   public static void main(String[] args)
   {
      double prev = timeTrial(125);
      for (int N = 250; true; N += N)
      {
         double time = timeTrial(N);
         StdOut.printf("%6d %7.1f ", N, time);
         StdOut.printf("%5.1f\n", time/prev);
         prev = time;
      }
   }
}
```

results of experiments

```
% java DoublingRatio
   250     0.0   2.7
   500     0.0   4.8
  1000     0.1   6.9
  2000     0.8   7.7
  4000     6.4   8.0
  8000    51.1   8.0
```

predictions

```
 16000   408.8   8.0
 32000  3270.4   8.0
 64000 26163.2   8.0
```



Analysis of experimental data (the running time of ThreeSum.count())

With our model developed, we can compare predict runtime given larger values of N where the experiment would no longer be feasible.