Algorithm speed over time

- Computers double in speed every $x$ years
- Running time in $y$ years:
  \[
  T(y) = \Theta\left((1/2)^{y/x}\right) = \Theta(2^{-y/x})
  \]
Algorithm dataset size

- For many applications, dataset size $n$ is also doubling every $z$ years.
Implications

- Assume speed doubling time = dataset doubling time \((y = z)\)
- then in \(y\) years =
  \[ T(n,y) = \Theta(n) \times \Theta(2^{-y/x}) \]
- Dataset size \(n\) in \(x\) years = \(\Theta(2^{y/x})\)
- For a linear algorithm
  \[ T(n,y) = \Theta((2^{y/x})^2 \times 2^{-y/x}) = \Theta(1) \]
- Increase in CPU speed roughly offsets increase in dataset size
Why quadratic algorithms are too slow

• For a quadratic algorithm, running time in \( y \) years =

\[
T(n,y) = \Theta((2^{y/x})^2) \times \Theta(2^{-y/x}) = \Theta(2^{y/x})
\]

• Running time doubles every

• If \( y=3 \), in 8 years algorithm will run

\[2^3 = 8 \text{ times as slow}\]