CSE 548: Algorithms

Fall 2022

R. Sekar

Topics

The topic of this course . . .

- Theoretical study of correctness and performance of (abstract) programs
  - **Performance = run time** + memory use
- What we will learn:
  - **How to design algorithms that perform better**
  - How to prove their correctness
  - How to analyze their performance
- Performance is important, *but not at the expense of:*
  - Functionality
  - Correctness, reliability
  - Programmer effort, maintainability, extensibility, modularity, simplicity

Administrivia

- Course web page:
  - http://seclab.cs.sunysb.edu/sekar/cse548/
    - Redirected from the department page for CSE 548
  - General information, lecture schedule and notes, etc.
- Blackboard:
  - Handouts, assignment submission
- Piazza:
  - All important announcements
  - Discussion forum and emails

Academic Integrity

- Do not copy from any one, or any source (on the Internet or elsewhere)
- The penalty for cheating is an F-grade, plus referral to graduate school. *No exception*, regardless of the “amount” of copying involved.
- In addition, if you cheat, you will be unprepared for the exams, and will do poorly.
- To encourage you to work on your own, we scale up assignment scores by about 30% to a maximum 100%
- Ethics homework will be distributed today, due before

Insertion Sort

procedure *InsSort*(**int** $A[n]$, **int** n)

for $(j = 1; j < n; j{+}{+})$

  $i = j - 1;$

  $key = A[j];$

  while $(i \geq 0 \;\&\&\; A[i] > key)$

    $A[i + 1] = A[i];$

    $i = i - 1;$

  $A[i + 1] = key;$

Insertion Sort

procedure *InsSort*(**int** $A[n]$, **int** n)

for $(j = 1; j < n; j{+}{+})$ /***Invariant:** $A[0 \ldots j-1]$ is sorted (ascending)*/

$i = j - 1;$

$key = A[j];$

while $(i \geq 0$ && $A[i] > key)$

$A[i+1] = A[i];$

$i = i - 1;$

$A[i+1] = key;$

Insertion Sort

procedure *InsSort*(**int** $A[n]$, **int** n)

for $(j = 1; j < n; j{+}{+})$ /***Invariant:** $A[0 \ldots j-1]$ is sorted (ascending)*/

$i = j - 1;$

$key = A[j];$

while $(i \geq 0$ && $A[i] > key)$

/*$A[i \ldots j-1] < key$ */

$A[i+1] = A[i];$

$i = i - 1;$

$A[i+1] = key;$

Insertion Sort

procedure *InsSort*(**int** $A[n]$, **int** n)

for $(j = 1; j < n; j{+}{+})$ /* **Invariant:** $A[0 \ldots j-1]$ is sorted (ascending) */

  $i = j - 1;$

  $key = A[j];$

  while $(i \geq 0$ && $A[i] > key)$

    /* $A[i \ldots j-1] < key$, location $A[i+1]$ is "free" */

   $A[i+1] = A[i];$

   $i = i - 1;$

  $A[i+1] = key;$

Running time of Insertion Sort

- Given by a function $T(n)$ where n is the array size

Running time of Insertion Sort

- Given by a function $T(n)$ where n is the array size
- Outer loop is executed n times

Running time of Insertion Sort

- Given by a function $T(n)$ where n is the array size
- Outer loop is executed n times
- For the j^{th} iteration of outer loop, the inner loop is executed at most j times

Running time of Insertion Sort

- Given by a function $T(n)$ where n is the array size
- Outer loop is executed n times
- For the j^{th} iteration of outer loop, the inner loop is executed at most j times
- Runtime

$$\sum_{j=1}^{n} j = \frac{n(n+1)}{2}$$

Running time of Insertion Sort

- Given by a function $T(n)$ where n is the array size
- Outer loop is executed n times
- For the j^{th} iteration of outer loop, the inner loop is executed at most j times
- Runtime

$$\sum_{j=1}^{n} j = \frac{n(n+1)}{2}$$

- Often, we focus on *asymptotic complexity:* a function that matches the growth rate of $T(n)$ as $n \to \infty$

Asymptotic Complexity

- Expressing complexity in terms of "number of steps" is a simplification
  - Each such operation may in fact take a different amount of time
  - But it is too complex to worry about the details, esp. because they differ across programming languages, processor types, etc.

Asymptotic Complexity

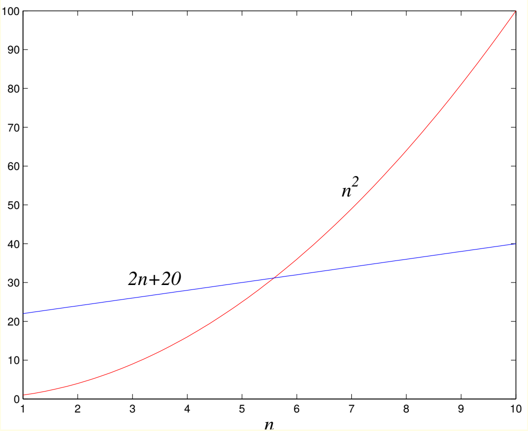
- Expressing complexity in terms of "number of steps" is a simplification
  - Each such operation may in fact take a different amount of time
  - But it is too complex to worry about the details, esp. because they differ across programming languages, processor types, etc.
- Why not simplify further?
  - Capture just the growth rate of $T(n)$ as a function of n
  - Ignore constant factors
    - No need to count operations in a loop (their number should be bounded by a constant)
  - Ignore exceptions from the formula for small values of n

Asymptotic Complexity: Big-O notation

Definition

Given functions $f, g : \mathbb{R} \longrightarrow \mathbb{R}$, we say $f = O(g)$, i.e., "f grows no faster than g," *iff*

$$\lim_{x \to \infty} f(x)/g(x) < c \text{ for some constant } c$$



Big-O notation: Examples

- $10n = O(n)$
- $0.0001n^3 + n = O(n^3)$
- $2^n + 10^n + n^2 + 2 = O(10^n)$
- $0.0001n \log n + 10000n = O(n \log n)$

Solving Divide-and-Conquer Recurrences: *Master Theorem*

If $T(n) = aT\left(\frac{n}{b}\right) + O(n^d)$ for constants $a > 0$, $b > 1$, and $d \geq 0$, then

$$T(n) = \begin{cases} O(n^d), & \text{if } d > \log_b a \\ O(n^d \log n) & \text{if } d = \log_b a \\ O(n^{\log_b a}) & \text{if } d < \log_b a \end{cases}$$

Solving Recurrences: Examples Using Master Theorem

$$T(n) = 2T(n/2) + n$$

$$T(n) = aT\left(\frac{n}{b}\right) + O(n^d)$$

$$T(n) = \begin{cases} O(n^d), & \text{if } d > \log_b a \\ O(n^d \log n) & \text{if } d = \log_b a \\ O(n^{\log_b a}) & \text{if } d < \log_b a \end{cases}$$

Solving Recurrences: Examples Using Master Theorem

$$T(n) = 4T(n/2) + n^3$$

$$T(n) = aT\left(\frac{n}{b}\right) + O(n^d)$$

$$T(n) = \begin{cases} O(n^d), & \text{if } d > \log_b a \\ O(n^d \log n) & \text{if } d = \log_b a \\ O(n^{\log_b a}) & \text{if } d < \log_b a \end{cases}$$

Solving Recurrences: Examples Using Master Theorem

$$T(n) = 3T(n/2) + n$$

$$T(n) = aT\left(\frac{n}{b}\right) + O(n^d)$$

$$T(n) = \begin{cases} O(n^d), & \text{if } d > \log_b a \\ O(n^d \log n) & \text{if } d = \log_b a \\ O(n^{\log_b a}) & \text{if } d < \log_b a \end{cases}$$

Types of complexity analyses

- **Worst case:** Bound that applies to any possible input.
  - *Guaranteed performance* — avoids assumptions about input.

Types of complexity analyses

- **Worst case:** Bound that applies to any possible input.
  - *Guaranteed performance* — avoids assumptions about input.
- **Average case:** Average performance across expected inputs.
  - Useful sometimes, especially when typical performance much better than worst-case
  - *Use with caution:*
    - Requires details of input distributions that is rarely available
    - Often ends up making unrealistic assumptions or simplifications

Types of complexity analyses

- **Worst case:** Bound that applies to any possible input.
  - *Guaranteed performance* — avoids assumptions about input.
- **Average case:** Average performance across expected inputs.
  - Useful sometimes, especially when typical performance much better than worst-case
  - *Use with caution:*
    - Requires details of input distributions that is rarely available
    - Often ends up making unrealistic assumptions or simplifications
- **Best case:** Not useful — unless you are trying to show than an algorithm is bad even in the best of circumstances!

Big-O versus Ω and Θ

$10n = O(n^2)$ — Note that O stands for upper bound

Big-O versus Ω and Θ

$10n = O(n^2)$ — Note that O stands for upper bound

Ω — Lower Bounds

$$f = O(g) \Leftrightarrow g = \Omega(f)$$

Examples:

$n^2 = \Omega(n)$ $\quad$ $n^2 = \Omega(n \log n)$ $\quad$ $2^n \neq \Omega(10^n)$

Big-O versus Ω and Θ

$10n = O(n^2)$ — Note that O stands for upper bound

Ω — Lower Bounds

$$f = O(g) \Leftrightarrow g = \Omega(f)$$

Examples:

$n^2 = \Omega(n)$ $\quad$ $n^2 = \Omega(n \log n)$ $\quad$ $2^n \neq \Omega(10^n)$

Θ — Tight Bounds

$$f = \Theta(g) \Leftrightarrow f = O(g) \land f = \Omega(g)$$

Example: $0.2n^2 + 2n + 7 = \Theta(n^2)$

Proofs

- Proofs arise in many contexts in this course
  - Correctness of algorithm
  - Optimality (or lack there of)
  - Time or space complexity

Proofs

- Proofs arise in many contexts in this course
  - Correctness of algorithm
  - Optimality (or lack there of)
  - Time or space complexity
- What is a proof?

Proofs

- Proofs arise in many contexts in this course
  - Correctness of algorithm
  - Optimality (or lack there of)
  - Time or space complexity
- What is a proof?
  - Establishes (or disproves) a predicate $P(x)$
    - May be universally ($\forall x\ Q(x)$) or existentially quantified ($\exists x\ Q(x)$)

Proofs

- Proofs arise in many contexts in this course
  - Correctness of algorithm
  - Optimality (or lack there of)
  - Time or space complexity
- What is a proof?
  - Establishes (or disproves) a predicate $P(x)$
    - May be universally ($\forall x\ Q(x)$) or existentially quantified ($\exists x\ Q(x)$)
  - *Explains* why $P(x)$ holds (or why not)

Proofs

- Proofs arise in many contexts in this course
  - Correctness of algorithm
  - Optimality (or lack there of)
  - Time or space complexity
- What is a proof?
  - Establishes (or disproves) a predicate $P(x)$
    - May be universally ($\forall x\ Q(x)$) or existentially quantified ($\exists x\ Q(x)$)
  - *Explains* why $P(x)$ holds (or why not)

Proofs

- Proofs arise in many contexts in this course
  - Correctness of algorithm
  - Optimality (or lack there of)
  - Time or space complexity
- What is a proof?
  - Establishes (or disproves) a predicate $P(x)$
    - May be universally ($\forall x \; Q(x)$) or existentially quantified ($\exists x \; Q(x)$)
  - *Explains* why $P(x)$ holds (or why not)
- Often, proofs are difficult to construct but can be fun to figure out!

Proving complexity of *InsSort*

Requires us to show

$$\sum_{j=1}^{n} j = \frac{n(n+1)}{2}$$

Proving complexity of *InsSort*

Requires us to show

$$\sum_{j=1}^{n} j = \frac{n(n+1)}{2}$$

OK, how about

$$\sum_{j=1}^{n} j^2 = \frac{n(n+1)(2n+1)}{6}$$

or

$$\sum_{j=1}^{n} j^3$$

Summation of i^k

- A single method is applicable for all $\sum i^k$. Let us take $\sum i^2$ as an example:

$$
\begin{array}{rcrcrcrcccccc}
\sum_{i=1}^n & i^3 & = & & & 1^3 & + & 2^3 & + & \cdots & + & (n-1)^3 & + & n^3 \\
\sum_{i=1}^n & (i-1)^3 & = & 0^3 & + & 1^3 & + & 2^3 & + & \cdots & + & (n-1)^3 & & \\
\hline
\sum_{i=1}^n & (i^3-(i-1)^3) & = & & & & & & & \cdots & & & & n^3
\end{array}
$$

Summation of i^k

- A single method is applicable for all $\sum i^k$. Let us take $\sum i^2$ as an example:

$$
\begin{array}{lccccccccccccc}
\sum_{i=1}^{n} & i^3 & = & & & 1^3 & + & 2^3 & + & \cdots & + & (n-1)^3 & + & n^3 \\
\sum_{i=1}^{n} & (i-1)^3 & = & 0^3 & + & 1^3 & + & 2^3 & + & \cdots & + & (n-1)^3 & & \\
\hline
\sum_{i=1}^{n} & (i^3-(i-1)^3) & = & & & & & & & \cdots & & & & n^3
\end{array}
$$

- Simplifying lhs using the identity $a^3 - b^3 = (a-b)(a^2+b^2+ab)$, we get

Summation of i^k

- A single method is applicable for all $\sum i^k$. Let us take $\sum i^2$ as an example:

$$
\begin{array}{rcccccccccccc}
\sum_{i=1}^n & i^3 & = & & & 1^3 & + & 2^3 & + & \cdots & + & (n-1)^3 & + & n^3 \\
\sum_{i=1}^n & (i-1)^3 & = & 0^3 & + & 1^3 & + & 2^3 & + & \cdots & + & (n-1)^3 & & \\
\hline
\sum_{i=1}^n & (i^3-(i-1)^3) & = & & & & & & & \cdots & & & & n^3
\end{array}
$$

- Simplifying lhs using the identity $a^3 - b^3 = (a-b)(a^2+b^2+ab)$, we get

$$
\begin{aligned}
\sum_{i=1}^n i^3 - (i-1)^3 &= \sum_{i=1}^n (i-(i-1))(i^2+(i-1)^2+i(i-1)) \\
&= \sum_{i=1}^n 3i^2 - 3i + 1 \\
&= 3\sum_{i=1}^n i^2 - 3\sum_{i=1}^n i + \sum_{i=1}^n 1 \qquad = n^3
\end{aligned}
$$

- Further simplifying, $\sum_{i=1}^n i^2 = (n^3 - n + 3\sum_{i=1}^n i)/3$
- Substituting for $\sum_{i=1}^n i$ from previous slide into rhs and simplifying:

$$\sum_{i=1}^n i^2 = \frac{n(n+1)(2n+1)}{6}$$

Another Example

Property

$\forall n\ n^2 + n + 41$ *is a prime number*

Yet Another Example

Property

$\forall n > 2\ a^n + b^n = c^n$ *has no integer solutions*

Yet Another Example

Property

$\forall n > 2\ a^n + b^n = c^n$ *has no integer solutions*

- "Fermat's Last Theorem" — stated by Fermat in 1637 in the margin of a copy of Arithmetica
- Fermat said he had a proof, but it won't fit in the margin!

Yet Another Example

Property

$\forall n > 2\ a^n + b^n = c^n$ *has no integer solutions*

- "Fermat's Last Theorem" — stated by Fermat in 1637 in the margin of a copy of Arithmetica
- Fermat said he had a proof, but it won't fit in the margin!
- Remained one of the most famous open mathematical problems for over 350 years! Finally proved in 1995 by Wiles.

Proof by Induction

Requires us to show

$$\sum_{j=1}^{k} j = \frac{k(k+1)}{2}$$

Base: For $j = 1$, easy to check that $1 = 1 * (1 + 1)/2$

Induction hypothesis: Assume that the equality holds for $k < n$

Induction Step:

Proof by Induction

Theorem

All horses have the same color

Base: Trivial, as there is a single horse.

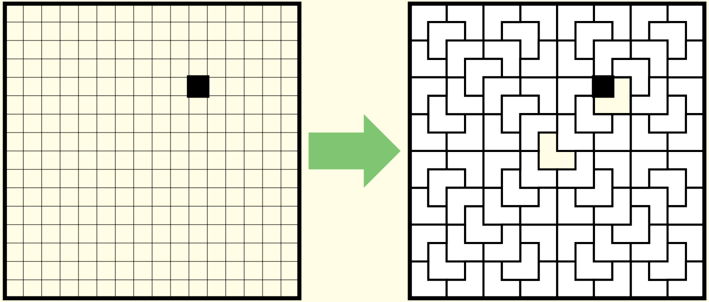
Induction hypothesis: All sets of horses with n or fewer horses have the same color.

Induction Step: Consider a set of $h_1, h_2, \ldots, h_{n+1}$. By induction hypothesis:

$$\underbrace{h_1, h_2, \ldots, h_n}_{\text{same color}}, h_{n+1} \qquad h_1, \underbrace{h_2, \ldots, h_n, h_{n+1}}_{\text{same color}}$$

This obviously means that all $n + 1$ horses have the same color!

Tiling a $2^n \times 2^n$ board with Triominos



Theorem

Any $2^n \times 2^n$ checkerboard with any single square removed can be tiled using L-shaped triominos.

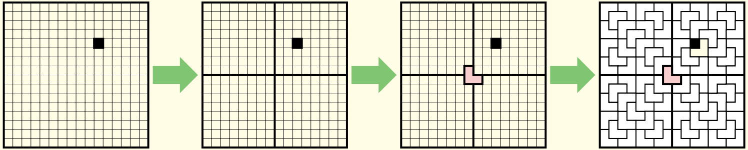
Figures/text from Jeff Erickson's "Algorithms"

Tiling a $2^n \times 2^n$ board with Triominos

Proof by top-down induction: Let n be an arbitrary non-negative integer. Assume that for any non-negative integer $k < n$, the $2^k \times 2^k$ grid with any square removed can be tiled using triominos. There are two cases to consider: Either $n = 0$ or $n \geq 1$.

- The $2^0 \times 2^0$ grid has a single square, so removing one square leaves nothing, which we can tile with zero triominos.
- Suppose $n \geq 1$. In this case, the $2^n \times 2^n$ grid can be divided into four smaller $2^{n-1} \times 2^{n-1}$ grids. Without loss of generality, suppose the deleted square is in the upper right quarter. With a single L-shaped triomino at the center of the board, we can cover one square in each of the other three quadrants. The induction hypothesis implies that we can tile each of the quadrants, minus one square.

In both cases, we conclude that the $2^n \times 2^n$ grid with any square removed can be tiled with triominos. $\square$



Recursion and Iteration Vs Induction

- Inductive proofs are often used in the context of recursive (and sometimes iterative) programs
- The recursive case closely resembles the inductive step in what we may call as top-down induction.
- Iterative programs are typically more closely related to bottom-up inductive proofs.

Relevant Mathematical Background (from CSE 150)

- Topics you should know well enough not to need a refresher:
  - Sets
  - Boolean algebra
  - Quantifiers
  - Functions and Relations
  - Graphs
- Topics you should look up in the next week or two:
  - Proof techniques
  - Summations
  - Recurrences and their solution
- Topics you should look up later in this course:
  - Counting
  - Dicrete probability

Proof techniques for professors

- *by obviousness:* “It is too obvious to waste our time with the details ...”

Proof techniques for professors

- *by obviousness:* “It is too obvious to waste our time with the details ...”
- *by omission:* “Proof is easy and left as an after-class exercise...”

  “The other 100 cases are similar”

Proof techniques for professors

- *by obviousness:* “It is too obvious to waste our time with the details ...”
- *by omission:* “Proof is easy and left as an after-class exercise...”

  “The other 100 cases are similar”
- *by lack of interest:* “Does anyone really want to see this?”

Proof techniques for professors

- *by obviousness:* “It is too obvious to waste our time with the details ...”
- *by omission:* “Proof is easy and left as an after-class exercise...”

  “The other 100 cases are similar”
- *by lack of interest:* “Does anyone really want to see this?”
- *by lack of time:* “We don’t have time to do this in class today ...”

Proof techniques for professors

- *by obviousness:* "It is too obvious to waste our time with the details ..."
- *by omission:* "Proof is easy and left as an after-class exercise..."

  "The other 100 cases are similar"
- *by lack of interest:* "Does anyone really want to see this?"
- *by lack of time:* "We don't have time to do this in class today ..."
- *by exhaustion:* Spend most of a lecture on background supposedly needed for a proof ...

Proof techniques for professors

- *by obviousness:* "It is too obvious to waste our time with the details ..."
- *by omission:* "Proof is easy and left as an after-class exercise..."

  "The other 100 cases are similar"
- *by lack of interest:* "Does anyone really want to see this?"
- *by lack of time:* "We don't have time to do this in class today ..."
- *by exhaustion:* Spend most of a lecture on background supposedly needed for a proof ...
- *by accumulated evidence:* "Long and diligent search has not revealed a counterexample."

Proof techniques for professors

- *by obviousness:* “It is too obvious to waste our time with the details ...”
- *by omission:* “Proof is easy and left as an after-class exercise...”

  “The other 100 cases are similar”
- *by lack of interest:* “Does anyone really want to see this?”
- *by lack of time:* “We don’t have time to do this in class today ...”
- *by exhaustion:* Spend most of a lecture on background supposedly needed for a proof ...
- *by accumulated evidence:* “Long and diligent search has not revealed a counterexample.”
- *by funding:* How could three different government agencies be wrong?

Proof techniques for professors

- *by obviousness:* “It is too obvious to waste our time with the details ...”
- *by omission:* “Proof is easy and left as an after-class exercise...”

  “The other 100 cases are similar”
- *by lack of interest:* “Does anyone really want to see this?”
- *by lack of time:* “We don’t have time to do this in class today ...”
- *by exhaustion:* Spend most of a lecture on background supposedly needed for a proof ...
- *by accumulated evidence:* “Long and diligent search has not revealed a counterexample.”
- *by funding:* How could three different government agencies be wrong?
- *by asserting intellectual superiority:* “You don’t have the background for the proof ...”

Proof techniques for professors

- *by obviousness:* "It is too obvious to waste our time with the details ..."
- *by omission:* "Proof is easy and left as an after-class exercise..."

  "The other 100 cases are similar"
- *by lack of interest:* "Does anyone really want to see this?"
- *by lack of time:* "We don't have time to do this in class today ..."
- *by exhaustion:* Spend most of a lecture on background supposedly needed for a proof ...
- *by accumulated evidence:* "Long and diligent search has not revealed a counterexample."
- *by funding:* How could three different government agencies be wrong?
- *by asserting intellectual superiority:* "You don't have the background for the proof ..."
- *by authority:* "I saw Fermat in the elevator and he said he had a proof ..."

Proof techniques for professors

- *by obviousness:* "It is too obvious to waste our time with the details ..."
- *by omission:* "Proof is easy and left as an after-class exercise..."

  "The other 100 cases are similar"
- *by lack of interest:* "Does anyone really want to see this?"
- *by lack of time:* "We don't have time to do this in class today ..."
- *by exhaustion:* Spend most of a lecture on background supposedly needed for a proof ...
- *by accumulated evidence:* "Long and diligent search has not revealed a counterexample."
- *by funding:* How could three different government agencies be wrong?
- *by asserting intellectual superiority:* "You don't have the background for the proof ..."
- *by authority:* "I saw Fermat in the elevator and he said he had a proof ..."
- *by intimidation:* "Don't be stupid; of course it's true!"

Proof techniques for professors

- *by obviousness:* "It is too obvious to waste our time with the details ..."
- *by omission:* "Proof is easy and left as an after-class exercise..."

  "The other 100 cases are similar"
- *by lack of interest:* "Does anyone really want to see this?"
- *by lack of time:* "We don't have time to do this in class today ..."
- *by exhaustion:* Spend most of a lecture on background supposedly needed for a proof ...
- *by accumulated evidence:* "Long and diligent search has not revealed a counterexample."
- *by funding:* How could three different government agencies be wrong?
- *by asserting intellectual superiority:* "You don't have the background for the proof ..."
- *by authority:* "I saw Fermat in the elevator and he said he had a proof ..."
- *by intimidation:* "Don't be stupid; of course it's true!"
- *by terror:* When intimidation fails...

Proof techniques for students

- *by example:* Provide one example, and claim that the ideas hold for all cases. Frequently used for partial credit in exams.

Proof techniques for students

- *by example:* Provide one example, and claim that the ideas hold for all cases. Frequently used for partial credit in exams.
- *by picture:* A more convincing form of proof-by-example.

Proof techniques for students

- *by example:* Provide one example, and claim that the ideas hold for all cases. Frequently used for partial credit in exams.
- *by picture:* A more convincing form of proof-by-example.
- *by profusion of adjectives and adverbs:* “As is quite clear, the elementary aforementioned statement is obviously valid.”

Proof techniques for students

- *by example:* Provide one example, and claim that the ideas hold for all cases. Frequently used for partial credit in exams.
- *by picture:* A more convincing form of proof-by-example.
- *by profusion of adjectives and adverbs:* "As is quite clear, the elementary aforementioned statement is obviously valid."
- *by vigorous handwaving:* For seminar settings, esp. if the presenter exudes supreme confidence ...

Proof techniques for students

- *by example:* Provide one example, and claim that the ideas hold for all cases. Frequently used for partial credit in exams.
- *by picture:* A more convincing form of proof-by-example.
- *by profusion of adjectives and adverbs:* "As is quite clear, the elementary aforementioned statement is obviously valid."
- *by vigorous handwaving:* For seminar settings, esp. if the presenter exudes supreme confidence ...
- *by cumbersome notation:* $\forall \varpi \in \mathfrak{P}\ \exists \nu \in \aleph\ \mathcal{P}(\varpi) \looparrowright \overset{\curvearrowleft}{\mathcal{Q}}(\nu)$ — works well in journal papers

Proof techniques for students

- *by example:* Provide one example, and claim that the ideas hold for all cases. Frequently used for partial credit in exams.
- *by picture:* A more convincing form of proof-by-example.
- *by profusion of adjectives and adverbs:* “As is quite clear, the elementary aforementioned statement is obviously valid.”
- *by vigorous handwaving:* For seminar settings, esp. if the presenter exudes supreme confidence ...
- *by cumbersome notation:* $\forall \varpi \in \mathfrak{P}\ \exists \nu \in \aleph\ \mathcal{P}(\varpi) \looparrowright \mathcal{Q}(\overset{\curvearrowleft}{\nu})$ — works well in journal papers
- *by throwing in the kitchen sink:* Write down all proofs vaguely related to the problem. A form of proof-by-exhaustion that is popular in exams.

Proof techniques for students

- *by example:* Provide one example, and claim that the ideas hold for all cases. Frequently used for partial credit in exams.
- *by picture:* A more convincing form of proof-by-example.
- *by profusion of adjectives and adverbs:* “As is quite clear, the elementary aforementioned statement is obviously valid.”
- *by vigorous handwaving:* For seminar settings, esp. if the presenter exudes supreme confidence ...
- *by cumbersome notation:* $\forall \varpi \in \mathfrak{P}\ \exists \nu \in \aleph\ \mathcal{P}(\varpi) \looparrowright \overset{\curvearrowleft}{\mathcal{Q}(\nu)}$ — works well in journal papers
- *by throwing in the kitchen sink:* Write down all proofs vaguely related to the problem. A form of proof-by-exhaustion that is popular in exams.
- *by illegibility:* Combines well with many other techniques in exams

Proof techniques for students

- *by example:* Provide one example, and claim that the ideas hold for all cases. Frequently used for partial credit in exams.
- *by picture:* A more convincing form of proof-by-example.
- *by profusion of adjectives and adverbs:* “As is quite clear, the elementary aforementioned statement is obviously valid.”
- *by vigorous handwaving:* For seminar settings, esp. if the presenter exudes supreme confidence …
- *by cumbersome notation:* $\forall\varpi \in \mathfrak{P}\ \exists\nu \in \aleph\ \mathcal{P}(\varpi) \looparrowright \overset{\curvearrowleft}{\mathcal{Q}}(\nu)$ — works well in journal papers
- *by throwing in the kitchen sink:* Write down all proofs vaguely related to the problem. A form of proof-by-exhaustion that is popular in exams.
- *by illegibility:* Combines well with many other techniques in exams
- *by mutual reference:* In reference A, Theorem 5 is said to follow from Theorem 3 in reference B, which is shown to follow from Corollary 6.2 in reference C, which is an easy consequence of Theorem 5 in reference A.