CSC165 fall 2014

Mathematical expression

Danny Heap
heap@cs.toronto.edu
BA4270 (behind elevators)
http://www.cdf.toronto.edu/~heap/165/F14/
416-978-5899

Course notes, chapter 3, 4

Outline

inference rules

asymptotics

notes

annotated slides

get wrong right

Be careful proving a claim false. Consider the claim, for some suitably defined X, Y and P, Q:

$$S : \qquad \forall x \in X, \forall y \in Y, P(x,y) \Rightarrow Q(x,y)$$

To **disprove** S, should you prove:

$$\forall x \in X, \forall y \in Y, P(x,y) \Rightarrow \neg Q(x,y)$$

What about

$$\forall x \in X, \forall y \in Y, \neg(P(x,y) \Rightarrow Q(x,y))$$

Explain why, or why not.

Define $T(n)$ by:

$$\forall n \in \mathbb{N} \qquad T(n) \Leftrightarrow \exists i \in \mathbb{N}, n = 7i + 1.$$

Take some scrap paper, **don't** write your name on it, and fill in as much of the proof of the following claim as possible:

$$\forall n \in \mathbb{N}, T(n) \Rightarrow T(n^2)$$

Now fill in as much of the **disproof** of the following claim as possible:

$$\forall n \in \mathbb{N}, T(n^2) \Rightarrow T(n)$$

allowed inference

At this point you've been introduced to some rules of inference, that allow you to reach conclusions in certain situations. You may use these (see pages 44–46 of the course notes) to guide your thinking, or as marginal notes to justify certain steps:

conjunction elimination: If you know $A \wedge B$, you can conclude A separately (or B separately).

existential instantiation: If you know that $\exists k \in X, P(k)$, then you can certainly pick an element with that property, let $k' \in X, P(k')$.

disjunction elimination: If you know $A \vee B$, the additional information $\neg A$ allows you to conclude B.

implication elimination: If you know $A \Rightarrow B$, the additional information A allows you to conclude B. On the other hand, the additional information $\neg B$ allows you to conclude $\neg A$.

universal elimination: If you know $\forall x \in X, P(x)$, the additional information $a \in X$ allows you to conclude $P(a)$.

more inferences

Here are some rules that allow you to introduce new logical structures

implication introduction: If you assume A and, under that assumption, B follows, than you can conclude $A \Rightarrow B$.

universal introduction: If you assume that a is a generic element of D and, under that assumption, derive $P(a)$, then you can conclude $\forall a \in D, P(a)$.

existential introduction: If you show $x \in X$ and you show $P(x)$, then you can conclude $\exists x \in X, P(x)$.

conjunction introduction: If you know A and you know B, then you can conclude $A \wedge B$.

disjunction introduction: If you know A you can conclude $A \vee B$.

sorting strategies

Which algorithm do you use to sort a 5-card euchre hand?

- insertion sort
- selection sort
- some other sort?

If you use one of the first two, the number of "steps" you execute will more than quadruple if you graduate from euchre to a 13-card bridge hand.

If you were enough of a virtuoso to use mergesort or quicksort on your cards, the change from euchre to bridge would roughly double your work.

We are most interested in how quickly running-time grows with the size of the problem, since these quickly swamp constant-factor differences between algorithms that are of the same "order."

different, but the same?

Suppose you could count the "steps" required by an algorithm in some sort of platform-independent way. You would find that the steps required for insertion sort and selection sort on lists of size n were no more than some quadratic functions of n

To a computer scientists, even though they may vary by substantial constant factors, all quadratic functions are the "same" — they are in $\mathcal{O}(n^2)$.

$$g(n) = n^2 \qquad f(n) = 3n^2 + 50 \qquad h(n) = 15n^2 + n$$

counting costs

want a coarse comparison of algorithms "speed" that ignores hardware, programmer virtuosity

which speed do we care about: best, worst, average? why?

define idealized "step" that doesn't depend on particular hardware and idealized "time" that counts the number of steps for a given input.

linear search

```
def LS(A,x) :
  """ Return index i such that x == L[i].  Otherwise, ret
    i = 0
    while i < len(A) :
        if A[i] == x :
            return i
        i = i + 1
    return -1
```

Trace LS([2,4,6,8],4), and count the time complexity $t_{\mathrm{LS}}([2,4,6,8],4)$

What is $t_{\mathrm{LS}}(A,x)$, if the first index where x is found is j?

What is $t_{\mathrm{LS}}(A,x)$ is x isn't in A at all?

worst case

denote the worst-case complexity for program P with input $x \in I$, where the input size of x is n as $W_P(n) = \max\{t_P(x) \mid x \in I \wedge \text{size}(x) = n\}$

The upper bound $W_P \in \mathcal{O}(U)$ means

$$\begin{aligned}
&\exists c \in \mathbb{R}^+, \exists B \in \mathbb{N}, \forall n \in \mathbb{N}, n \geq B \\
&\Rightarrow \max\{t_P(x) \mid x \in I \wedge \text{size}(x) = n\} \leq cU(n) \\
&\text{That is:} \quad \exists c \in \mathbb{R}^+, \exists B \in \mathbb{N}, \forall x \in I, \text{size}(x) \geq B \\
&\qquad \Rightarrow t_P(x) \leq cU(\text{size}(x))
\end{aligned}$$

The lower bound $W_P \in \Omega(L)$ means

$$\begin{aligned}
&\exists c \in \mathbb{R}^+, \exists B \in \mathbb{N}, \forall n \in \mathbb{N}, n \geq B \\
&\qquad \Rightarrow \max\{t_P(x) \mid x \in I \wedge \text{size}(x) = n\} \geq cL(n) \\
&\text{That is:} \quad \exists c \in \mathbb{R}^+, \exists B \in \mathbb{N}, \forall n \in \mathbb{N}, n \geq B \\
&\qquad \Rightarrow \exists x \in I, \text{size}(x) = n \wedge t_P(x) \geq cL(n)
\end{aligned}$$

bounding a sort

```
def IS(A) :
    """ IS(A) sorts the elements of A in non-decreasing or
    i = 1
    while i < len(A) :
        t = A[i]
        j = i
        while j > 0 and A[j-1] > t :
            A[j] = A[j-1] # shift up
            j = j-1
        A[j] = t
        i = i+1
```

I want to prove that $W_{\text{IS}} \in \mathcal{O}(n^2)$.

Notes

annotated slides

- monday's annotated slides
- wednesday's annotated slides