CSC236 winter 2020

theory of computation

Colin Morris
colin@cs.toronto.edu
http://www.teach.cs.toronto.edu/~colin/236/W20/

January 6, 2020

Outline

What is this course?

P'(n): Every bipartite graph on n vertices has no more than $n^2/4$ edges if n is even, or $(n^2-1)/4$ edges if n is odd.

base case: An empty bipartite graph has 0 vertices and 0 edges, and $0 \le 0^2/4$, which verifies $P(0)$.

inductive step: Let n be an arbitrary, fixed, natural number. Assume $P'(n)$, that every bipartite graph on n vertices has no more than $n^2/4$ edges if n is even, or $(n^2-1)/4$ edges if n is even. I will show that $P'(n+1)$ follows, that every bipartite graph on $n+1$ edges has no more than $(n+1)^2/4$ edges if $n+1$ is even, or $[(n+1)^2-1]/4$ edges if $n+1$ is odd.

Let G be an arbitrary bipartite graph on $n+1$ vertices. Remove a vertex, together with its edges, from G's larger partition to produce a new bipartite graph G'. There are two possibilities, depending on whether $n+1$ is even or odd:

case $n+1$ is odd: G's smaller partition has, at most, $n/2$ vertices, so we removed at most $n/2$ edges to produce G'. $n+1$ odd means n is even, so by assumption $P(n)$, G' has at most $n^2/4$ edges, so accounting for the edges removed G had, at most:

$$\frac{n^2}{4} + \frac{n}{2} = \frac{n^2+2n}{4} \le \frac{(n+1)^2-1}{4}$$

So $P'(n+1)$ follows in this case.

case $n+1$ is even: G's smaller partition has, at most, $(n+1)/2$ vertices, so we removed at most $(n+1)/2$ edges to produce G'. $n+1$ even means n is odd, so by assumption $P(n)$ G' has at most $(n^2-1)/4$ edges, so accounting for the edges removed G had, at most:

$$\frac{n^2-1}{4} + \frac{n+1}{2} = \frac{n^2+2n+1}{4} \le \frac{(n+1)^2}{4}$$

So $P'(n+1)$ follows in this case.

$P'(n+1)$ follows in both possible cases ■

```
    try:
        f = codecs.open(filename, "r", encoding='UTF-8')
        lines = joinLines(f.readlines())
        f.close()
    except:
        pprint("Cannot read file: %s" % filename, sys.stderr)
        sys.exit(-2)

    return lines


def scan_for_selected_frames(lines):
    """scans for frames that should be rendered exclusively, 
true if such frames have been found"""
    p = re.compile("^!====\s*(.*?)\s*====(.*)", re.VERBOSE)
    for line in lines:
        mo = p.match(line)
        if mo is not None:
            return True
    return False

def line_opens_unselected_frame(line):
    p = re.compile("^====\s*(.*?)\s*====(.*)", re.VERBOSE)
    if p.match(line) is not None:
        return True
    return False

def line_opens_selected_frame(line):
```

More like this... ...than this

Who am I?

Who are you?

Course information sheet

- Info is a subset of what's on the course website
- Let's take a tour now
  - (Sorry, this part is boring.)

About these slides

- Adapted from Danny Heap
- Plain slides posted online in advance
- Annotated slides uploaded after lecture
  - You may want to annotate your own copy during lecture

We behave as though you already know...

- CSC165 material, especially proofs and big-Oh material
  - But you can *relax* the structure a little (more on this later)
- Chapter 0 material from *Introduction to Theory of Computation*
- recursion, efficiency material from CSC148

By end of course you'll know...

1. Several flavours of proof by induction
2. Reasoning about recurrences
3. Proving the correctness of programs
4. Formal languages

Simple induction

Domino fates foretold

$$[\, P(0) \ \wedge \ (\forall n \in \mathbb{N}, P(n) \Rightarrow P(n+1)\,)\,] \Longrightarrow \forall n \in \mathbb{N}, P(n)$$

If the initial case works,
and each case that works implies its successor works,
then all cases work

Simple induction outline

1. Define predicate, $P(n)$
2. Inductive step
   2.1 Let $n \in \mathbb{N}$
   2.2 Assume $P(n)$ (inductive hypothesis)
   2.3 use it to show that $P(n+1)$ holds
3. Verify base case(s) (AKA basis)

Example: triangular numbers

Show that for any n, $\sum_{k=0}^{n} k = \frac{n(n+1)}{2}$.

1. Define predicate

2. Inductive step

3. Base case

Sometimes we need more than one base case

Show that $\forall n \in \mathbb{N}, 3^n \geq n^3$

Sometimes we need more than one base case

Show that $\forall n \in \mathbb{N}, 3^n \geq n^3$

Bases other than zero

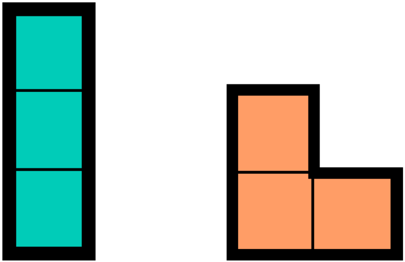
Prove that $n! \geq n^2$ for $n >$???

Bases other than zero

Prove that $n! \geq n^2$ for $n >$???

Trominoes

See https://en.wikipedia.org/wiki/Tromino



Can a $2^n \times 2^n$ square grid, **with one subsquare removed**, be tiled (covered without overlapping) by "chair" trominoes?

Trominoes

$P(n)$: a $2^n \times 2^n$ square grid with a subsquare removed can be tiled with chair trominoes.

The units digit of any power of 7 is one of 1, 3, 7, or 9

Scratch work

The units digit of any power of 7 is one of 1, 3, 7, or 9

Use the simple induction outline

The units digit of any power of 7 is one of 1, 3, 7, or 9

Use the simple induction outline

The units digit of any power of 7 is one of 1, **2**, 3, 7, or 9

Is the claim still true? What happens if you add this other case to the inductive step?