and we have $M, M_i, (i \le k)$, s.t.
$g(\vec{n}) \le A_M(\max(\vec{n}))$
and $h_i(\vec{x}) < A_{M_i}(\max(\vec{x}))$ $\forall i \le k$

- Let $N = \max(M, M_1, \ldots, M_k)$
- then $A_N(x) \ge A_M(x), A_{M_1}(x), \ldots, A_{M_k}(x)$ (An incr. fn)
- so: $F(\vec{x}) = g(\vec{h}(\vec{x}))$
$\le A_N(\max(\vec{h}(\vec{x})))$
$\le A_N(A_N(\max(\vec{x})))$
$\le A_{N+1}(\max(\vec{x}) + 1)$

why: $A_{N+1}(y+1) = A_N(A_{N+1}(y))$
$\ge A_N(A_N(y))$

$\le A_{N+2}(\max(\vec{x}))$ using (*)

so: $N+2$ works as our $M$ for $F$ ✓

(ii) (Primitive recursion)
- Sps $F$ is defined by prim recursion:
$F(0, \vec{x}) = g(\vec{x})$
$F(n+1, \vec{x}) = h(F(n, \vec{x}), n, \vec{x})$
and moreover we have $N, M$ s.t.
$g(\vec{x}) < A_M(\max(\vec{x}))$ and $h(i, n, \vec{x}) < A_N(\max(i, n, \vec{x}))$
- Let $k = \max(N, M) + 1$

- then $F(0,\vec{x}) = g(\vec{x}) \leq A_{k-1}(\max(\vec{x}))$ at most M

- else: $F(1,\vec{x}) = h(F(0,\vec{x}), 0, \vec{x})$
$\leq A_{k-1}(\max(F(0,\vec{x}), 0, \vec{x}))$
$\leq A_{k-1}(\max(A_{k-1}(\max(\vec{x})), 0, \vec{x}))$
since $A_{k-1}$ incr. in argument → $= A_{k-1}(A_{k-1}(\max(\vec{x})))$
$< A_{k-1}(A_k(\max \vec{x}))$
from def'n of h here → $= A_k(\max(\vec{x}) + 1)$

- repeating this: can prove inductively.
$F(h,\vec{x}) < A_k(\max(\vec{x}) + h)$
which is $\leq A_k(2\max(\vec{x}, h))$
$< A_{k+2}(\max(\vec{x}, h))$ by lemma 2

So $k+2$ works as our M for F ✓✓

↳ Theorem says: seq of p.r. functions $A_0 < A_1 < A_2 < \cdots$ eventually dominates every p.r. function

- actual calculation is pencil, but result not so surprising:
pr f's are built from initial functions (only one that grows is $+1$) using comp'n and prim rec'n
⇒ suggests should be possible to compute a (pr) bound on growth rate of a given pr function

based on its depth of def'n (how many composition + prim recursion applied in constructing f)

↳ A(n,x) does this in a precise way.

—

- So: we've shown A(n,x) = $A_n(x)$ is not pr (though each $A_n$ is pr)
- next goal: show A(n,x) is recursive.

- Here's an alternate (recursive-y) def'n of A(n,x):

$$A(0,x) = x+1$$
$$A(n+1,0) = A(n,1)$$
$$A(n+1,x+1) = A(n, A(n+1,x))$$

↳ we derived the last identity from orig. def'n (rewritten: $A_{n+1}(x+1) = A_n(A_{n+1}(x))$)
↳ can also go backwards (from these to orig def'n (you try)

**Def'n:** (lexicographical ordering)
For $n,x,m,y \in \mathbb{N}$, we write
$(m,y) <_{lex} (n,x)$ iff
$m<n$ or $(m=n \wedge y<x)$

- Def'n of $A(n,x)$ above is recursive "in the lex ordering" i.e. value of $A(n,x)$ depends on values of $A(m,y)$ for $(m,y) <_{lex} (n,x)$

---

↳ we're going to show $A$ is recursive by showing we can "do a $\mu$-search for a computation of $A(n,x)$"

↳ need to code computations of $A(n,x)$ as numbers.

↳ before that: what is a computation?

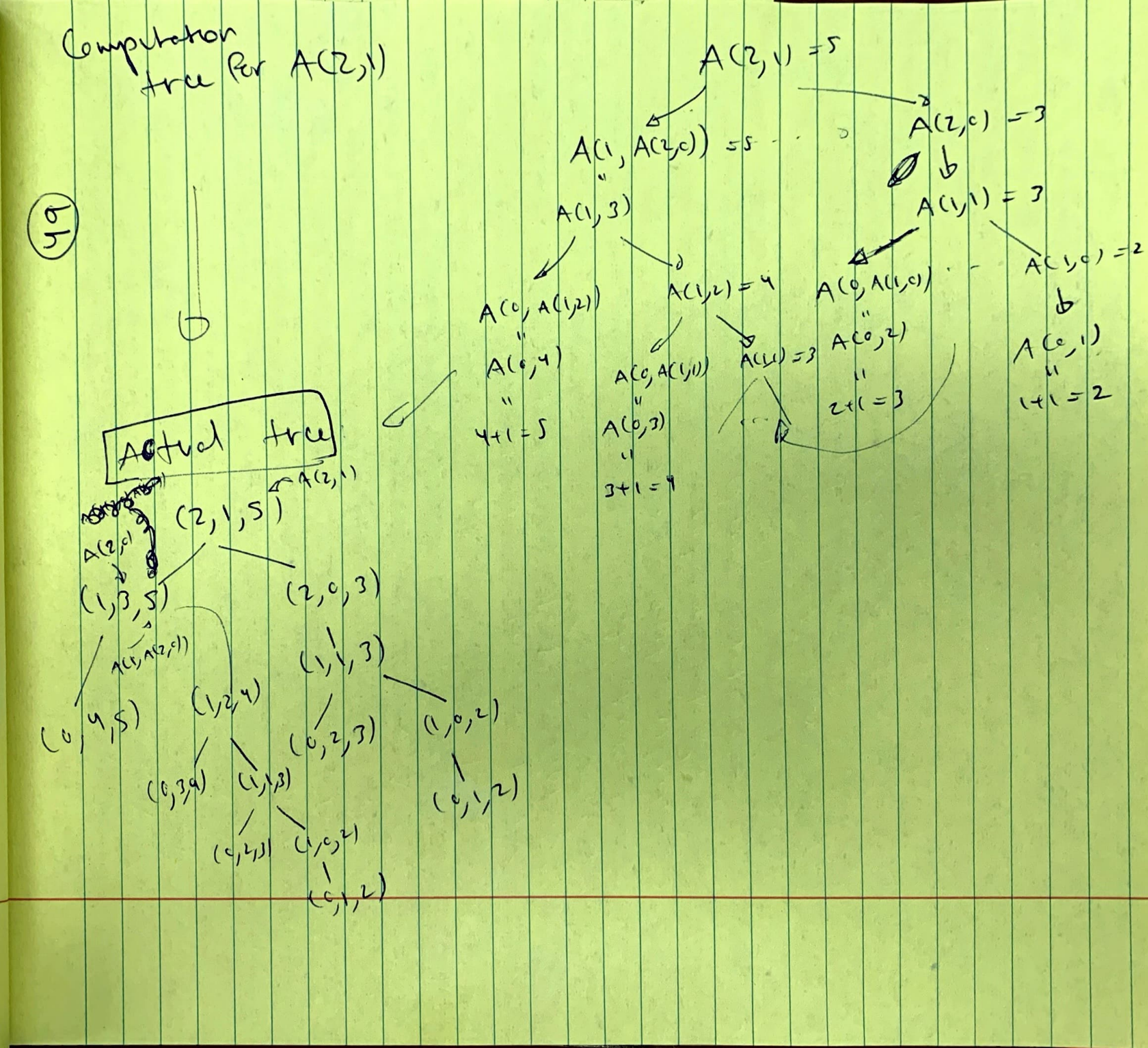
↳ we introduce a concept of a computation tree for $A(n,x)$
- → has nodes $(n, x, A(n,x))$
- → three kinds of nodes:

① $(0, x, x+1)$ (terminal node)

② $(n, 0, A(n,0))$ w/ $n > 0$
↳ has one child node: $(n-1, 1, A(n-1,1))$

③ $(n, x, A(n,x))$ w/ $n, x > 0$
↳ has two child nodes:
- ① $(n-1, A(n,x-1), A(n-1, A(n,x-1)))$
- ② $(n, x-1, A(n,x-1))$

Computation tree for A(2,1)

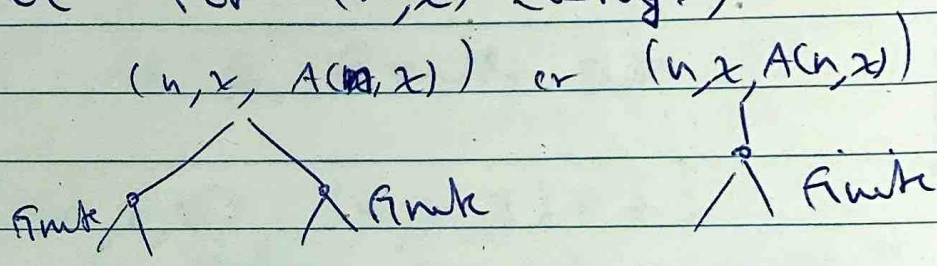


**Lemma** The computation tree of $A(n,x)$ is finite $\forall$ pairs $(n,x) \in \mathbb{N}^2$.

**Pf.** Clear.

Actual pf: by "lexicographic induction"

- true (ie "termnd pairs" $(0,x)$
- Sps true $\forall$ pairs $(m,y)$ s.t. $(m,y) <_{lex} (n,x)$
- then true for $(n,x)$ (why?)




(Q4u: why is this a legitimate form of induction!)

**Def'n** For $(n,x) \in \mathbb{N}^2$ let $S_{(n,x)}$ be the set of all triples $(n', x', A(n',x'))$ that appear in the computation tree of $A(n,x)$.

Some properties of $S_{(n,x)}$:

① $(0,x',z) \in S_{(n,x)} \Rightarrow z = x+1$

② $(n',0,z) \in S_{(n,x)} \Rightarrow (n'-1, 1, z) \in S_{(n,x)}$

③ $(n'+1, x'+1, z) \in S_{(n,x)} \Rightarrow$
$\exists w [(n'+1, x', w) \in S_{(n,x)} \wedge (n', w, z) \in S_{(n,x)}]$

Def'n if $S$ is a set of triples ($S \subseteq \mathbb{N}^3$) satisfying these properties we say $S$ is <u>correct</u>.

<u>Claim</u> $A(n,x) = z$ iff there is a correct set $S$ with $(n,x,z) \in S$.

Pf ($\Rightarrow$) Let $S = S_{(n,x)}$

($\Leftarrow$) by lexicographic induction on $(n,x)$ (you try!) ✓

---

- Sps $S \subseteq \mathbb{N}^3$ is a finite set of triples.
- Let $S' := \{\langle n,x,z\rangle : (n,x,z) \in S\}$
  = set of codes $\langle n,x,z\rangle$ for $(n,x,z) \in S$
  $2^{n+1} 3^{x+1} 5^{z+1}$
- Viewing $S'$ as a set of natural numbers, write $S'$ in order:
  $S' = \{u_0 < u_1 < \cdots < u_{k-1}\}$
- Let $s = \langle u_0, u_1, \ldots, u_{k-1}\rangle$ = "$\langle S\rangle$" be the code for this sequence.
- Define a unary relation $C(s)$ on $\mathbb{N}$ by:
  $C(s)$ iff there is a correct set of triples $S$ s.t. $s = \langle S\rangle$

- Define a relation $P(n,x,s)$ by

$P(n,x,s)$ iff

$$(s) \wedge \exists z \ (n,x,z) \in S$$

(where $S$ denotes set of triples coded by $s$)

"$s$ is a sequence of codes for a correct set $S$ and $(n,x,A(n,x)) \in S$"

**Claim** $C$ and $P$ are p.r. relations

**Pf**: Given $s \in \mathbb{N}$ we have

$C(s)$ iff $\mathrm{Seq}(s) \wedge \mathrm{lh}(s) > 0$

("$s$ is a code for a nonempty sequence")

$\wedge$ $\forall i \leq \mathrm{lh}(s) \ (\mathrm{Seq}((s)_i) \wedge \mathrm{lh}((s)_i) = 3)$

("every entry in $s$ is a code for a triple")

$\wedge$ "if $(s)_i = \langle 0, x', z \rangle$, then $z = x' + 1$"

$\wedge$ "if $(s)_i = \langle n', 0, z \rangle$, then $\exists j \leq \mathrm{lh}(s)$ s.t. $(s)_j = \langle n'-1, 1, z \rangle$"

$\wedge$ "... describe rule ③ in p.r. way..."

Can you write more precisely?

everything p.r.

$\Rightarrow$ up to writing the informal clauses above more formally, shows $C(s)$ is p.r. ✓

# Stack algorithm for computing A(n,x)

- Let's write Anm for A(n,m)
- Can compute

A21 = A1A20 = A1A11 = A1A0A10
= A1A0A01 = A1A02 = A13 = A0A12
= A0A0A11 = ... = A0A03 = A04 = 5

compute

- Or even drop A's (remembering to work right to left)

21 = 120 = 111 = 1010 = 1001 = 102 = 13
= 012 = 0011 = ... = 003 = 04 = 5

- Can capture this calculation as a "stack algorithm"
- alg works on strings $s = n_1 \cdots n_k$ of natural numbers
- begins on input $s = nx$

**Algorithm:**

Given $s$:

If $s = n$, return $n+1$

If $s = s'nx$, then:

(possibly empty)

if $n = 0$, replace $s'nx$ by $s'(x+1)$

if $n > 0$, $x = 0$, replace $s'nx$ by $s'(n-1)1$

if $n>0$, $x>0$ replace
$s'nx$ by $s(n-1)n(x-1)$

Claim on a given input $s=nx$ this algorithm terminates w/ $A(n,x)$

pf: (HW)