**Time Complexity**

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

(a):

(b):

(c):

(d):

(e):

**Fibonacci 1**

Prove: .

By trial and error: It appears that for all

To prove: For all positive integers

By induction on . Base case:

Step, assume: Indeed, true that for all

To prove:

LHS:

Suffices to prove:

(by dividing the above by )

It is indeed true that

**Multiplication**

Graphical user interface, text, application

Description automatically generated

Suppose instead of both and being n-bit, is n-bit and is m-bit. What is the worst-case time efficiency of ?

Proposed:

Time Efficiency:

So, final answer:

**Fibonacci 2**

Let be the nth Fibonacci number, Prove .

* Somewhere, we have shown:
* But here, seek to show: There exists positive real , for all in
* Natural proof strategy for “there exists” – construction (i.e., propose some concrete , and show that it works)
* Try some small values for , and see what would work
* Appears that works. Adopt it and check if proof goes through. Now, proof by induction with
* Base case,
* Step: Seek to show given that for all
* by induction assumption

**Fibonacci 3**

Let be the nth Fibonacci number, Prove .

* Recall from logic: not (there exists an egg-laying mammal) for all mammals , is not egg-laying
* Here, : There exists positive real , for all natural ,
* So here, need to prove: Given any positive real , it is true that there exists such that
* By contradiction: Suppose that there exists positive real , such that, for all natural ,
* Then:
* This is true only if is “large” compared to
* What is large? We need
* Try :
* Try :
* Try :
* Try :
* Try :
* Try :
* Prove by induction: for all natural
* Base case : See above
* Step:
* So far: We have shown that indeed, for ,

**Selection Sort**

What is a meaningful characterization of the time efficiency of ?

* Suppose we invoke . In . Suppose now, min is at index in . This index of a min in is at index
* Suppose on input: . Then A evolves in as follows:
* For time efficiency: Need to make meaningful assumption(s)
* Customary Assumptions: (1) is unbounded, (ii) each is bounded
* What should we count? Suppose we all agree that counting # swaps is a meaningful measure for time efficiency
* Then:
* Now, let’s say we want to get a bit more fine-grained. Incorporate (worst case) time for each swap # swaps
* So now, time efficiency:

**Modular Simplification**

1. Is ?

So:

So:

Now:

So:

So:

1. Is divisible by ?

Trick: Keep exponentiating until numbers start to repeat.

Suppose we repeatedly exponentiate :

So: . And . So

Now check whether is divisible by . Indeed:

Repeat with . Repeated exponentiation of :

So:

Now: .

.

1. Is a multiple of ?

is prime. And .

Compare with :

.

Show that if has a multiplicative inverse modulo , then this inverse is unique (modulo ).

Let’s assume .

Suppose are both multiplicative inverses of . Then:

: Substitution Rule:

Then:

(2): Commutativity

Suppose . Show that is an integer.

So: , which is divisible by .

We say that is a square root of modulo a prime if . Show that if and has a square root modulo , then is such a square root.

Let be the square root of . Then: .

Write . Then,

Keep in mind: .

Try plugging in in the last expression:

Is ?

So, we’re asking: Is ?

So at least one of: or must be .