LMHanoi - Tower of Hanoi with Lego Mindstorms

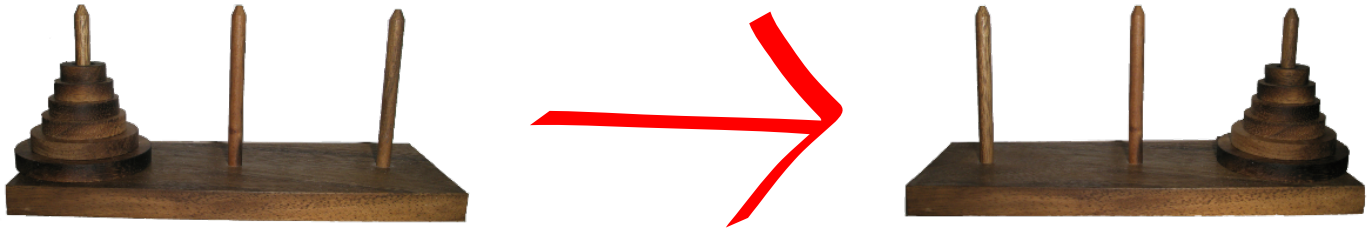
Markus Flatz, Peter Palfrader, Andreas Rottmann

25.01.2011

Overview

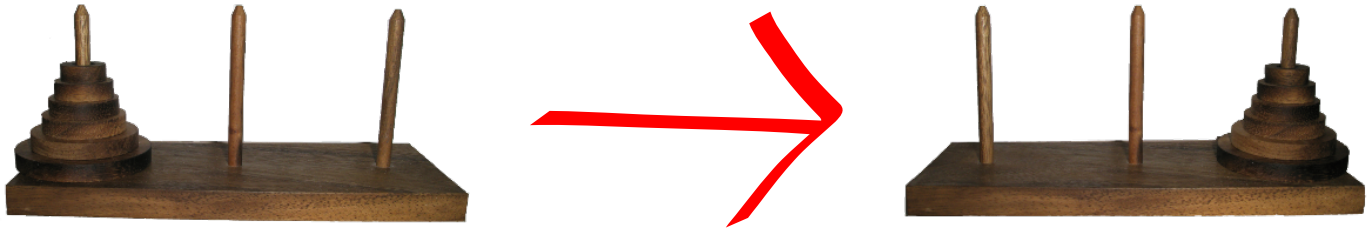
The puzzle

- Move a set of disks from one rod to another



The puzzle

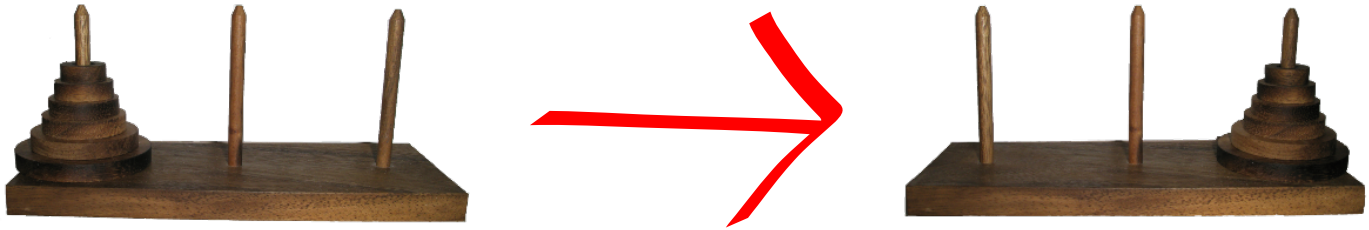
- Move a set of disks from one rod to another



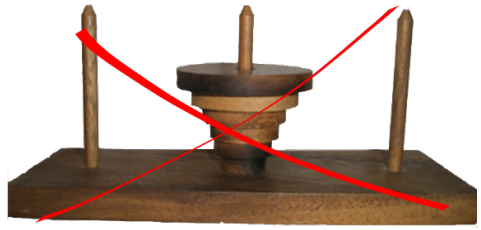
- With moving only ever one disk

The puzzle

- Move a set of disks from one rod to another



- With moving only ever one disk
- No larger disk may rest on a smaller one



Lego Mindstorms

- Extends Lego with new building blocks:
  - Sensors
  - Motors
  - Programmable computer
- NXT 2.0: 32-bit ARM7 microprocessor, 64 kB RAM and 256 kB Flash memory
- 4 input, 3 output ports
- Bluetooth, USB
- Sound

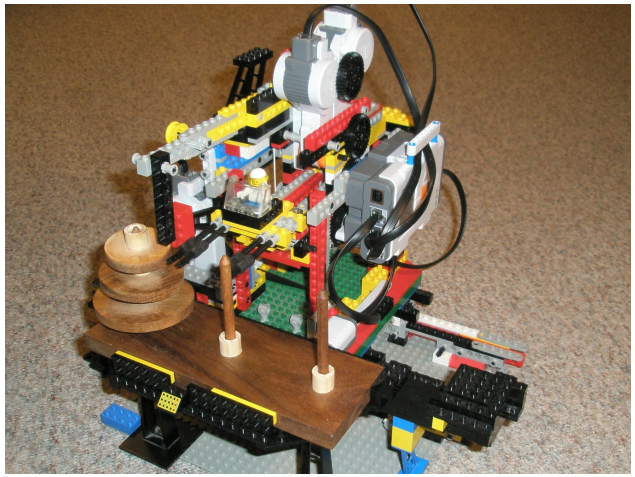
~~ABuNQEUC~~[1] NXC
Not eXactly C

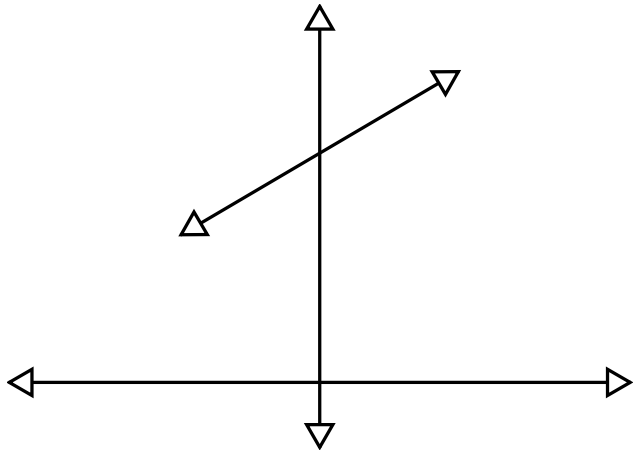
- Free compiler to build Lego Mindstorms executables from something similar to C
- Somewhat documented
- Several limitations, e.g., no recursion, no passing of mutexes as parameters
- Workarounds require abusing the preprocessor

[1] almost, but not quite, entirely unlike C

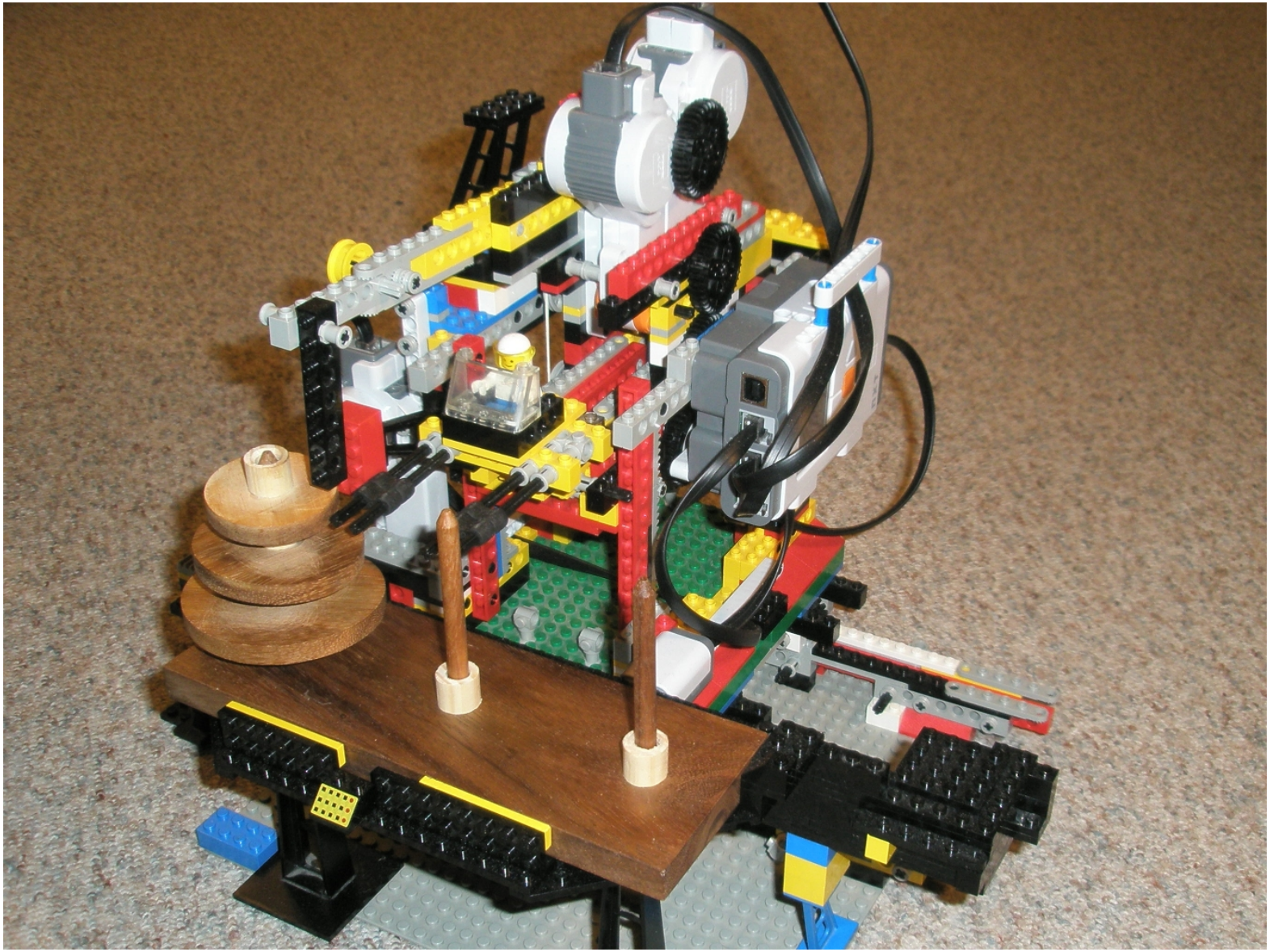
The robot: Klaus

- Forklift
- Can move along three axes

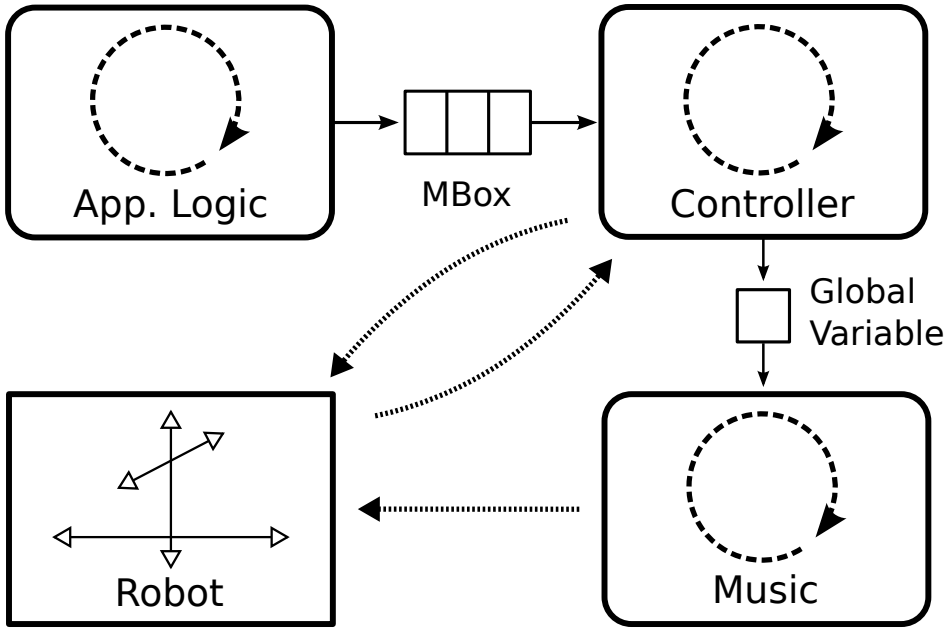




- Movement limited by sensors for two of them
- Using touch and color sensors



Software



Three threads

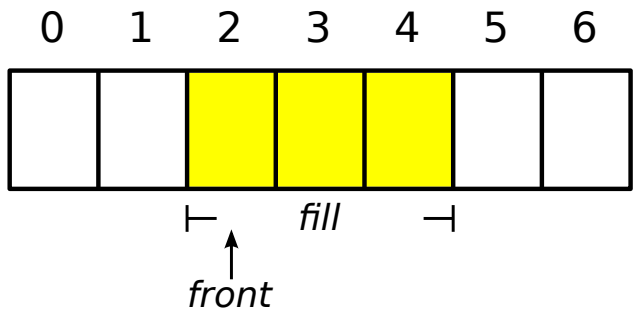
- Application logic (Tower of Hanoi algorithm)
- Robot controller
- Music playback

Inter-thread communication

NXC Primitives

- Mutex

Mailbox



- Circular buffer
- Mutex (protects fill count and front pointer)
- Two counting semaphores (full, empty)[1]

Application logic

- Determine movement sequence

Recursive algorithm

```
void hanoi(int n, rod source, rod help, rod dest) {
    if (n > 0) {
      hanoi(n - 1, source, dest, help);
      move_disk(source, dest);
      hanoi(n - 1, help, source, dest);
    }
}
```

Recursive algorithm

```
void hanoi(int n, rod source, rod help, rod dest) {
    if (n > 0) {
      hanoi(n - 1, source, dest, help);
      move_disk(source, dest);
      hanoi(n - 1, help, source, dest);
    }
}
```

- $2^n - 1$ moves

Recursive algorithm

```
void hanoi(int n, rod source, rod help, rod dest) {
    if (n > 0) {
      hanoi(n - 1, source, dest, help);
      move_disk(source, dest);
      hanoi(n - 1, help, source, dest);
    }
}
```

- $2^n - 1$ moves
- Optimal movement sequence

Iterative algorithm [2],[3]

- Rods source, help and destination on a circle
- Even number of disks: ordered clockwise
- Odd number of disks: ordered counterclockwise

Iterative algorithm [2],[3]

- Rods source, help and destination on a circle
- Even number of disks: ordered clockwise
- Odd number of disks: ordered counterclockwise

```
while (not all disks on destination rod) {
    move the smallest disk one rod clockwise;
    if (a disk other than the smallest can be moved) {
        move this disk;
    }
}
```

Iterative algorithm [2],[3]

- Rods source, help and destination on a circle
- Even number of disks: ordered clockwise
- Odd number of disks: ordered counterclockwise

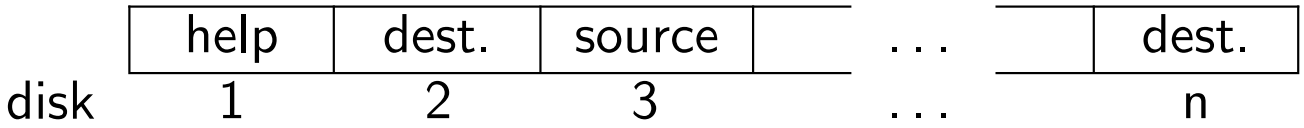
```
while (not all disks on destination rod) {
    move the smallest disk one rod clockwise;
    if (a disk other than the smallest can be moved) {
        move this disk;
    }
}
```

- Same movement sequence

Implementation

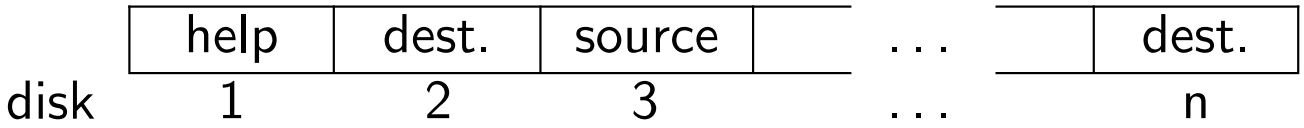
- No support for recursion
- Choose iterative algorithm

Naïve implementation



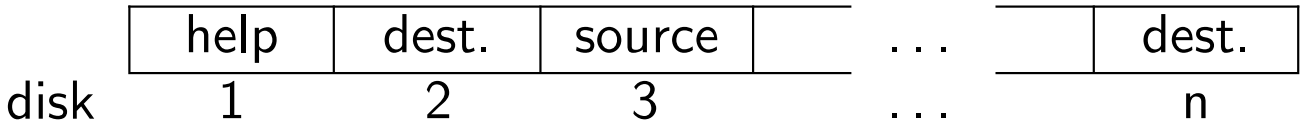
- Store the current location of each disk

Naïve implementation



- Store the current location of each disk
- Space complexity: $\mathcal{O}(n)$

Naïve implementation



- Store the current location of each disk
- Space complexity: $\mathcal{O}(n)$
- Time complexity per move: $\mathcal{O}(n)$

Refined implementation

source	3				
help	4	1			
destination	5	2			

height:	1	2	2
	source	help	dest.

- Store the disks on each rod

Refined implementation

source	3				
help	4	1			
destination	5	2			

height:	1	2	2
	source	help	dest.

- Store the disks on each rod
- Store the height of each rod's stack

Refined implementation

source	3				
help	4	1			
destination	5	2			

height:	1	2	2
	source	help	dest.

- Store the disks on each rod
- Store the height of each rod's stack
- Approximately three times as much memory

Refined implementation

source	3				
help	4	1			
destination	5	2			

height:	1	2	2
	source	help	dest.

- Store the disks on each rod
- Store the height of each rod's stack
- Approximately three times as much memory
- Space complexity: $\mathcal{O}(n)$

Refined implementation

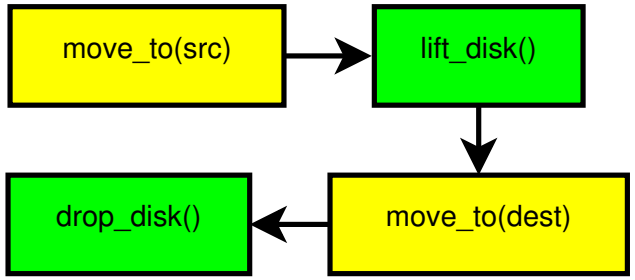
source	3				
help	4	1			
destination	5	2			

height:	1	2	2
	source	help	dest.

- Store the disks on each rod
- Store the height of each rod's stack
- Approximately three times as much memory
- Space complexity: $\mathcal{O}(n)$
- Time complexity per move: $\mathcal{O}(1)$

Robot controller (1)

General structure

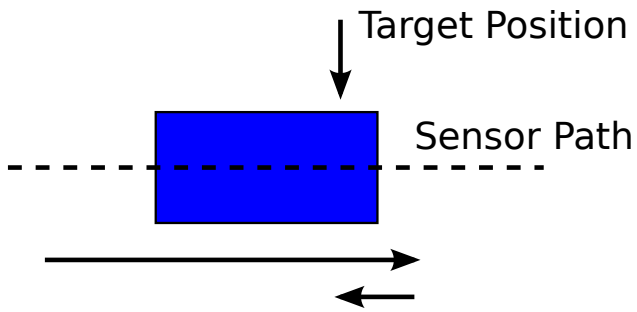


- Get movements from mailbox and execute them
- Maintains current position
- Wagon position stops on color sensor readings
- Fork vertical movement stops on touch sensor
- Fork moves horizontally by fixed amount

Robot controller (2)

Specialties

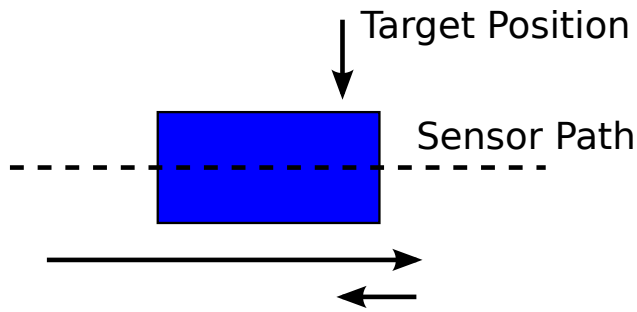
- Overshoot



Robot controller (2)

Specialties

- Overshoot



- Use raw mode of color sensor to detect blue

Music

- Additional concurrency

Music

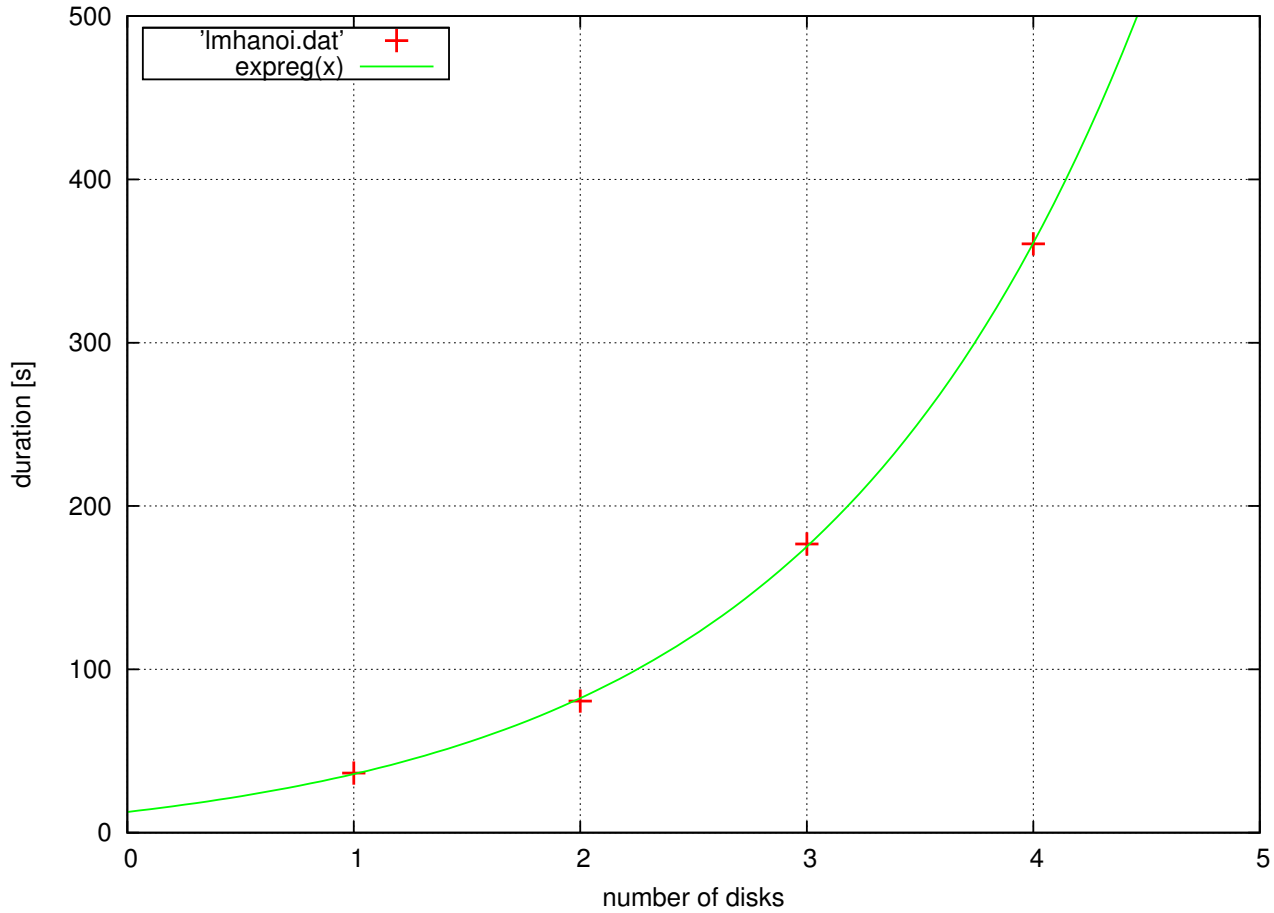
- Additional concurrency
- Situational

Experiment

$$duration(n) = (2^n - 1) \cdot d + c$$

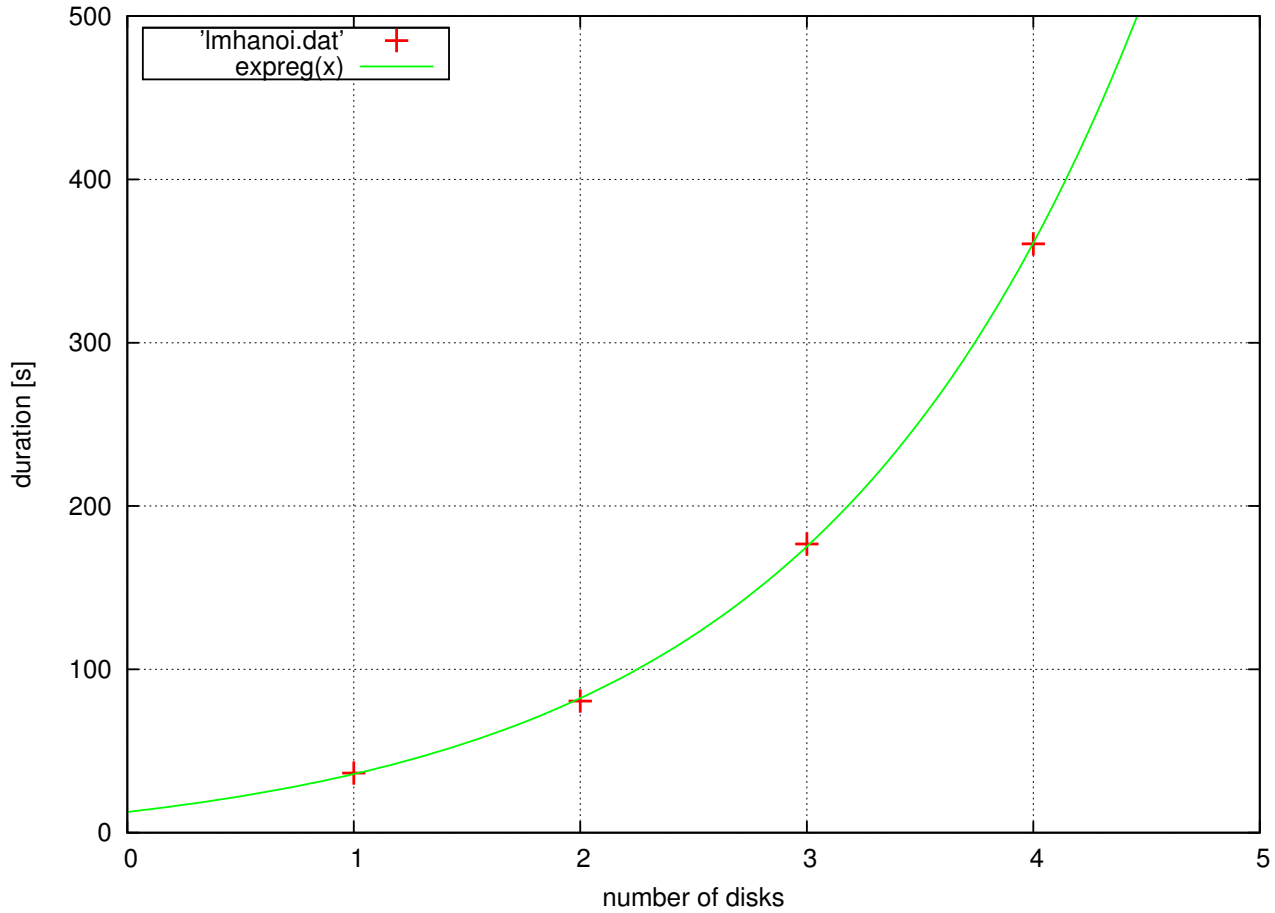
Experiment

$$duration(n) = (2^n - 1) \cdot d + c$$



Experiment

$$duration(n) = (2^n - 1) \cdot d + c$$



$$expreg(x) = (e^{0.693 \cdot x} - 1) \cdot 23.2 + 12.7$$

References


John A. Trono and William E. Taylor.
Further comments on "a correct and unrestrictive implementation of general semaphores".
SIGOPS Oper. Syst. Rev., 34:5–10, July 2000.


Wikipedia.
Tower of Hanoi – Wikipedia, The Free Encyclopedia, 2011.
https://secure.wikimedia.org/wikipedia/en/w/index.php?title=Tower_of_Hanoi&oldid=407310252.


Wikipedia.
Türme von Hanoi – Wikipedia, Die freie Enzyklopädie, 2011.
https://secure.wikimedia.org/wikipedia/de/w/index.php?title=T%C3%BCrme_von_Hanoi&oldid=83701385.

Thank you for your attention.