NAME
nissy — a Rubik's cube solver and FMC assistant

SYNOPSIS
nissy [**-b**]
nissy *command* [options...]

DESCRIPTION
nissy is a Rubik's Cube solver. It uses techniques from Herbert Kociemba's Cube Explorer and Tomas Rokicki's nxopt. With 4 cores at 2.5GHz and using less than 3Gb of RAM, Nissy can find the optimal solution for a random Rubik's cube position in about a minute on average. Nissy can also solve different substeps of the Thistlethwaite's algorithm and more.

When run without any argument an interactive shell is launched, otherwise the provided *command* is executed and nissy terminates. If the option **-b** is given, every argument after it is ignored and the shell is launched without any prompt or welcome message. This can be used to run nissy in batch mode, for example writing a list of commands in a *file* (one per line) and running *nissy -b < file*

The commands that can be run in the interactive shell are the same that can be run non-interactively and are provided below.

COMMANDS
The available *commands* are the following:

commands
> List all available commands.

gen [**-t** *N*]
> Generate all tables used by nissy. Run this to complete your installation. If *N* is specified, *N* CPU threads will be used (defaults to 1).

help [*command*]
> Display help. If no *command* is given, a generic help message is printed, otherwise a specific help relative to *command* is returned.

invert *scramble*
> Invert the given scramble.

print *scramble*
> Display a text-only description of the cube obtained after applying *scramble*.

quit Quit nissy.

scramble [**-n** *N*][*type*]
> Print a randomly-generated (random position) scramble If *N* is given, it produces *N* scrambles. *type* can be specified to be one of the following:
>
> *eo*   Scramble with solved EO on F/B axis.
>
> *corners*
> > Scramble with solved edges (only cornes are scrambled).
>
> *edges* Scramble with solved corners (only edges are scrambled).

solve *step* [*options*]*scramble.*
> Solve the given *step* on the given *scramble.* By default it finds only one (shortest) solution, without using niss, and it displays the number of moves at the end of the line. The options for the *solve* command are the following:

-a Print all solutions: some solutions are filtered out by default for some steps, for examples EOs that finish with F', with this options they are not.

-c Display only the number of solutions found, not the solutions themselves.

-m *min*
  Only look for solution that are at least *min* moves long.

-M *MAX*
  Only look for solution that are at most *MAX* moves long.

-n *N* Try to find *N* solutions. By default and unless the **-M** or **-o** options are used, at most one solution is returned. If at least one of **-M** and **-o** is used, all the solutions found within the given bounds are returned. The option **-s** overwrites these default behaviors and at most *N* solutions are returned, still satisfiyng the other constraints.

-N Allow use of NISS.

-o Only find solutions that require the minimum number of moves.

-O *N* Only find solutions that require at most *N* moves more than the optimal solution. If *N* is 0, this is equivalent to **-o**

-p Plain style: do not print the number of moves.

-t *N* Use *N* CPU threads. By default nissy uses only 1 thread. Using more than one thread will improve performance, but the optimal number depends on your machine and operating system. Generally, using one less than the number of threads of your CPU works quite well.

-v Verbose mode: print some information during the search and print each solution as it is found instead of only printing them all together at the end.

steps List all available *steps* for the *solve* command.

twophase *scramble*
  Find a solution using a two-phase method. This does not guarantee to return an optimal solution (and in fact most often it does not), but it is very fast.

unniss *scramble*
  Rewrite the scramble without using NISS.

version
  Display version information.

SCRAMBLES
All the commands above that accept a scramble also accept a **-i** option with no arguments. If this option is given, multiple scrambles are read from standard input (one per line) until and EOF is found, at which point stdin is cleared.

ENVIRONMENT
Data is stored in the folder pointed to by **$NISSYDATA.** If that variable is unset the folder **$XDG_DATA_HOME/nissy** or **$HOME/.nissy** is used instead. If none of this environment variables is defined (e.g. in a non-UNIX system), the current folder is used.

EXAMPLES
The command:
```
nissy solve -v -O 1 "R'U'FD2L2FR2U2R2BD2LB2D'B2L'R'BD2BU2LU2R'U'F"
```
Returns:
```
Searching depth 0
```

```
Searching depth 1
(some more lines)
Searching depth 16
D2 F' U2 D2 F' L2 D R2 D F B2 R' L2 F' U' D
Searching depth 17
D2 F' U2 D2 F' L2 D R2 D F B2 R' L2 F' U' D (16)
```
Notice that the solution is printed twice: the first time it is printed as soon as it is found as requested by the -v option.

The command:
```
nissy solve eofb -m 4 -M 5 -N -n 6 "R'U'FD2L2 FR2 U2R2BD2 L B2 D' B2
L' R'"
```
Returns:
```
U  B U' B (4)
U  (B R' B) (4)
(U B R' B) (4)
U2 F R2 F (4)
U2 B U2 B (4)
(U2 B R' B) (4)
```

On a UNIX shell, the composite command
```
nissy scramble -n 2 | nissy solve -i > file.txt
```
Generates two random scrambles, solves them and saves the result to file.txt. The file will look something like this:
```
>>> Line: D U2 F D B' F L2 D' F2 R2 L B2 L' U2 B2 R F2 L' D2
U2 R2 F2 L B2 D' R2 D' F U L2 B' U' R2 D2 R2 U (17)
>>> Line: D B R U' B' L2 U L U D2 R L B2 U2 L2 U2 R U2 B2 L F2
D' F R' D B L2 B R2 L U L U2 B D' U R U F2 (18)
```

AUTHORS
Sebastiano Tronto <sebastiano.tronto@gmail.com>

SOURCE CODE
Source code is available at **https://github.com/sebastianotronto/nissy**