Moopec

A tool for creating programming problems

Rui Mendes

Motivation

- Pandemic
- Large classes
- Need to provide feedback
- Automatic evaluation
- Many solutions available online
- Need to create a large number of problems

Interface

Point & Click

- More intuitive
- Creating problems involves filling fields and pressing buttons
- Most fields are self explanatory
- Tests must be created elsewhere

Batch

- Users need to learn a small language
- Problems are created in a textfile using a DSL
- Tests may be created elsewhere or generated programatically
- Can create many problems by editing a single text file

Automatic evaluation systems

- Most opt for a Point & Click interface
- Intuitive
- Problems are created one at a time
- Creating several problems is **slow**

Mooshak

- Developped by this session's chairman
- Used in ICPC and IOI contests
- Robust
- Used in classes in several Universities
- Allows both binary and partial evaluation
- File system based
- Contests and problems may be created by adding files

How to tackle different problem types?

- Compiler
- Static corrector
- Dynamic corrector

Problem types

Problem	inputs and outputs
Basecode	Compiler or Static corrector
Algorithm	Time or space complexity
Buggy code	Compiler or Static corrector

Creating problems in Mooshak

- Name, letter and description
- Limits (e.g., CPU, memory)
- Each test
  - input and output
  - Arguments
  - Context
  - How many points?
  - Is it shown?
  - Feedback

DSL

NAME	Problem name; problem ends with END
LETTER	Folder name
TESTS	One or more tests; ends with END
IMPORT	The name of a Python module to import
CODE	Python code, ends with END
SOLVER	Lambda or function name
DESCRIPTION	Uses markdown, ends with END
POINTS	Evenly distributed among all tests

DSL — Tests

INPUT	One line of input
LONGINPUT	One or more lines of input; ends with END
INPUTGEN	*n* followed by lambda or function name
FEEDBACK	feedback message for these tests
POINTS	Points per test
SHOW	Are they shown?

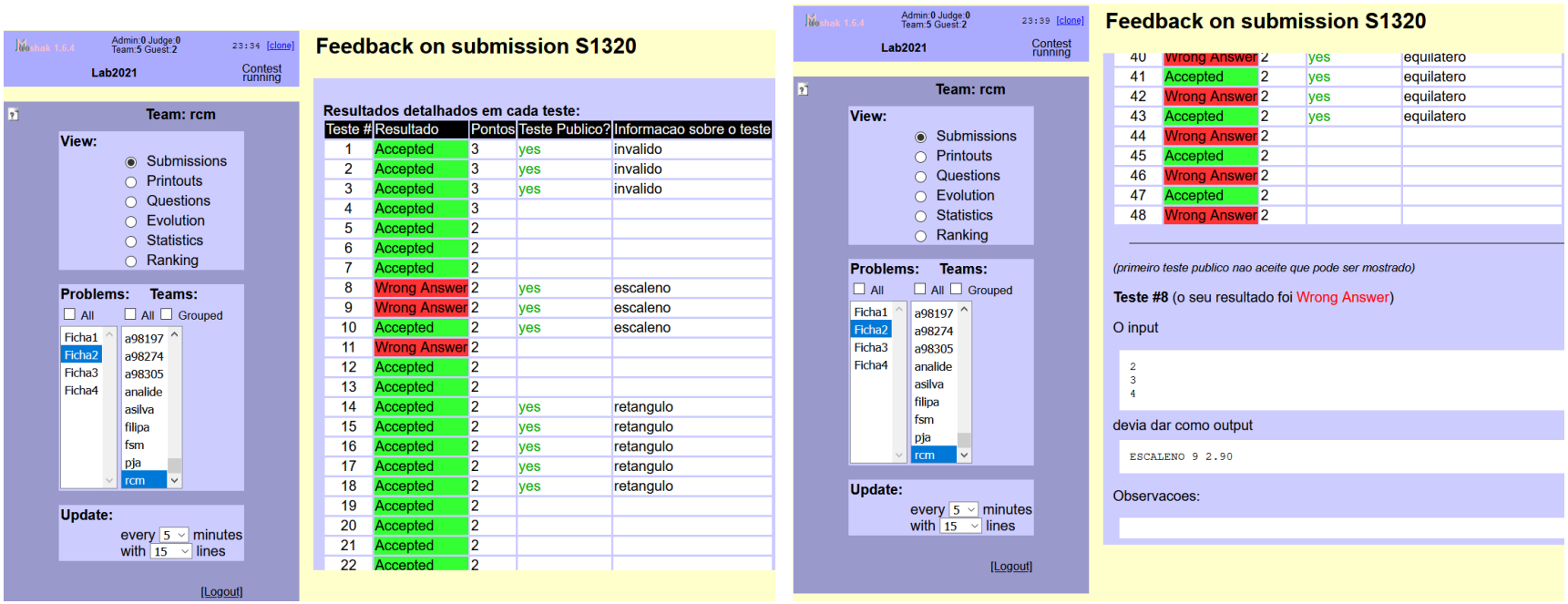
Example

```
NAME      Summing a bunch of  numbers
LETTER    A
SOLVER  lambda s: sum(int(x) ** 2 for x in s.split())
TESTS
    INPUT    10 20 30
    FEEDBACK     Example given in the problem description
    SHOW
END
DESCRIPTION
# Sum a list of numbers
Create a program that:
-  reads several numbers and
-  prints the sum of their squares
END
POINTS 100
END
```

Example

```
NAME     Summing a bunch of  numbers
LETTER   A
IMPORT sum_problem
SOLVER  solution
TESTS
    INPUTGEN 10 pequeno
    FEEDBACK small tests
END
DESCRIPTION
# Sum a list of numbers
Create a program that:
-  reads several numbers and
-  prints the sum of their squares
END
POINTS 100
END
```

Example of feedback



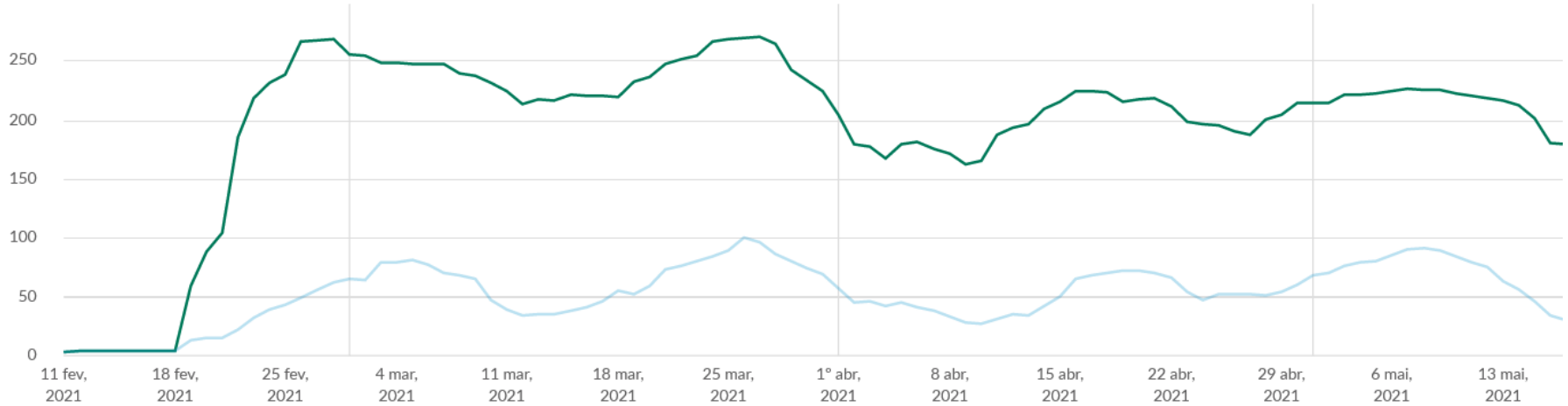
Anatomy of a project

- 5 deliverables (all tests are shown)
  1. 50 tests
  2. 50 tests
  3. 52 tests
  4. 100 tests
  5. 40 tests
- Final tests: 40 blind tests
- Created using Moopec
- Implemented as a contest with 6 problems
- Each problem was worth 100 points
- Each problem had its own description and several examples

Anatomy of a project: feedback

- Students could see the first test that failed
- They could ask for help solving the problems with their code in Slack
- Slack had 8435 messages
- Tutoring was also available by videoconference

Anatomy of a project: feedback on Slack



Anatomy of a project

Result	Problems						Total
	Final	Guiao1	Guiao2	Guiao3	Guiao4	Guiao5	
Accepted		635	656	661	174	74	**2200**
		6.1%	*6.3%*	*6.4%*	*1.7%*	*0.7%*	***21.2%***
Presentation Error		9		1			**10**
		0.1%		*0.0%*			***0.1%***
Wrong Answer	35	157	473	727	274	69	**1735**
	0.3%	*1.5%*	*4.6%*	*7.0%*	*2.6%*	*0.7%*	***16.7%***
Output Limit Exceeded							
Memory Limit Exceeded	36				11	4	**51**
	0.3%				*0.1%*	*0.0%*	***0.5%***
Time Limit Exceeded	1	1	15		17	53	**87**
	0.0%	*0.0%*	*0.1%*		*0.2%*	*0.5%*	***0.8%***
Invalid Function					5	1	**6**
					0.0%	*0.0%*	***0.1%***
Runtime Error	535	314	480	287	1905	1002	**4523**
	5.1%	*3.0%*	*4.6%*	*2.8%*	*18.3%*	*9.6%*	***43.5%***
Compile Time Error	76	414	434	236	331	105	**1596**
	0.7%	*4.0%*	*4.2%*	*2.3%*	*3.2%*	*1.0%*	***15.4%***
Invalid Submission							
Program Size Exceeded	17	41	26	41	29	28	**182**
	0.2%	*0.4%*	*0.3%*	*0.4%*	*0.3%*	*0.3%*	***1.8%***
Requires Reevaluation							
Evaluating					1		**1**
					0.0%		***0.0%***
Total	700	1571	2084	1953	2747	1336	**10391**
	6.7%	*15.1%*	*20.1%*	*18.8%*	*26.4%*	*12.9%*	***100.0%***

Anatomy of a project

Problems					
Final	Guiao1	Guiao2	Guiao3	Guiao4	Guiao5
98 (1)	100 (2)	100 (1)	100 (2)	100 (1)	100 (2)
98 (16)	100 (31)	100 (58)	100 (29)	100 (38)	100 (51)
96 (29)	100 (3)	100 (9)	100 (1)	100 (6)	100 (20)
96 (13)	100 (4)	100 (6)	100 (2)	100 (17)	100 (39)
95 (2)	100 (25)	100 (29)	100 (19)	100 (28)	100 (12)
94 (45)	100 (43)	100 (72)	100 (77)	100 (43)	100 (78)
93 (11)	100 (8)	100 (6)	100 (19)	100 (23)	100 (35)
92 (12)	100 (37)	100 (24)	100 (14)	100 (44)	100 (67)
92 (16)	100 (47)	100 (45)	100 (60)	100 (79)	100 (87)
92 (1)	100 (4)	100 (2)	100 (2)	99 (3)	100 (5)
91 (56)	100 (10)	100 (16)	100 (12)	100 (39)	100 (59)
90 (1)	100 (4)	100 (4)	100 (4)	100 (4)	100 (9)
90 (1)	100 (46)	100 (44)	100 (21)	100 (45)	100 (53)
88 (18)	100 (15)	100 (35)	100 (32)	100 (161)	100 (68)
86 (8)	100 (6)	100 (26)	100 (55)	100 (81)	98 (44)

Conclusions

- Moopec allows the batch creation of exercises
- It allows the usage of Python for creating both tests' input and output
- It is possible to use arbitrary programs as well due to `system` and `subprocess`
- Problem descriptions can be created with markdown
- Author tested it by creating more than 50 problems

Avaliability

https://github.com/rcm/mooshak_problem_creator