

AI match maker for Prologin finals

Thibault Allançon
December 2018

Background

- Non-profit organization since 1991



- Non-profit organization since 1991
- National computer science contest

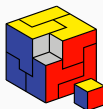


Prologin

- Non-profit organization since 1991
- National computer science contest
- Girls Can Code!



Prologuin
2019



- Free computer science contest
- Open to French-speaking students under 21
- Introduces the world of programming and algorithms

■ Qualification



The screenshot shows the Prologin 2019 website interface. The top navigation bar is dark blue with the Prologin 2019 logo on the left and links for 'Participer', 'Découvrir', 'S'entraîner' (highlighted), 'Archives', 'Forums', 'Connexion', and 'S'inscrire' on the right. Below the navigation bar, a breadcrumb trail reads 'Prologin > Entraînement > Qualification 2019 > Statuettes'. The main content area is titled 'Statuettes – Qualification 2019' with a sub-header 'Niveau 5'. The 'Énoncé' (Statement) section describes a problem where Haruhi is looking for a gift for her group of friends, represented by figurines of different heights. The input consists of two lines: the first line contains two integers n and m , and the second line contains a permutation of $1, \dots, n$. The output is the number of figurines that are taller than all their friends. The 'Entrée' (Input) section specifies the format of the input. On the right side of the page, there are links for 'Énoncé', 'Contraintes d'exécution', 'Exemples d'entrée/sortie', and 'Proposez votre solution'.

Prologin 2019

Participer Découvrir **S'entraîner** Archives Forums Connexion S'inscrire

Prologin > Entraînement > Qualification 2019 > Statuettes

Statuettes – Qualification 2019

Niveau 5

Énoncé

C'est maintenant la fin des vacances pour Haruhi et Joseph. Haruhi est à l'île de Pâques et cherche un cadeau pour son groupe d'amis. Elle découvre une guirlande composée de statuettes de l'île, de différentes tailles. Son groupe d'amis a une photo emblématique et pour y faire un clin d'œil elle aimerait avoir un bout de cette guirlande où chacune des statues représente un des amis du groupe. Les amis sont reconnaissables par leurs tailles relatives respectives.

Entrée

L'entrée comprendra :

- Deux entiers, n le nombre d'amis sur la photo, et m la taille totale de la guirlande (le nombre de statuettes sur la guirlande).
- Sur la ligne suivante, n entiers $1 \leq p_i \leq n$ donnent la position du i ème ami le plus petit. Ces tailles forment une permutation de $1, \dots, n$.
- Si $p_i = k$ alors l'ami en position k est le i ème plus petit des n amis sur la photo.
- Sur la ligne suivante m entiers h_j , la taille de la statuette en position j .

[Énoncé](#)

[Contraintes d'exécution](#)

[Exemples d'entrée/sortie](#)

[Proposez votre solution](#)

Online exercises and quiz (3 months)

National Computer Science Contest

- Qualification
- Regional events



Theoretical and practical exam (1 day)

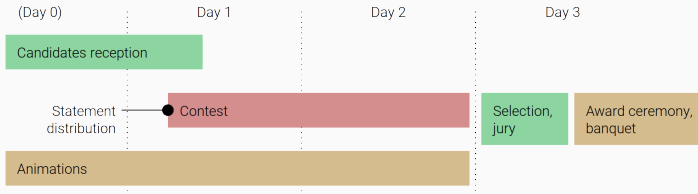
National Computer Science Contest

- Qualification
- Regional events
- Finals



Artificial intelligence battle (3 days)

The Finals



The Game

We create a custom game every year.



Our needs

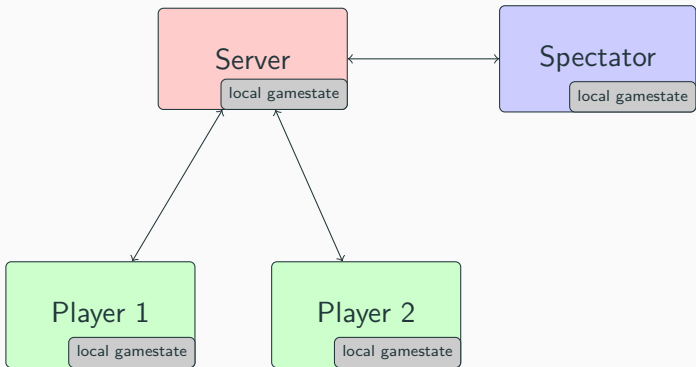
- Flexible
- Polyglot
- Distributed
- Isolated

stechec

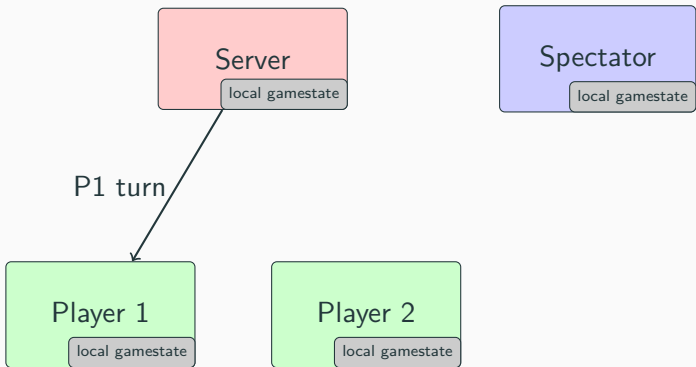
stechec2

Architecture

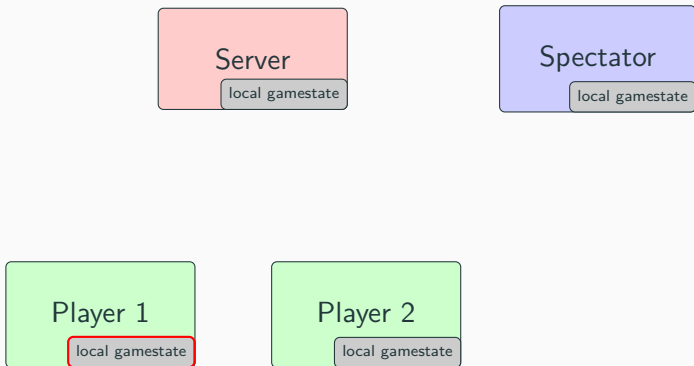
stechec2 architecture



stechec2 architecture

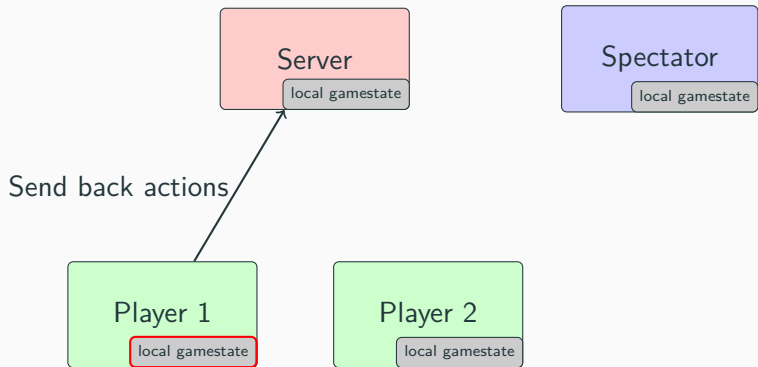


stechec2 architecture



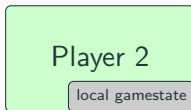
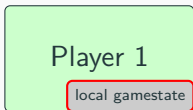
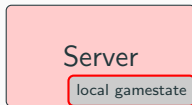
- Call user functions
- Check actions on local gamestate

stechec2 architecture

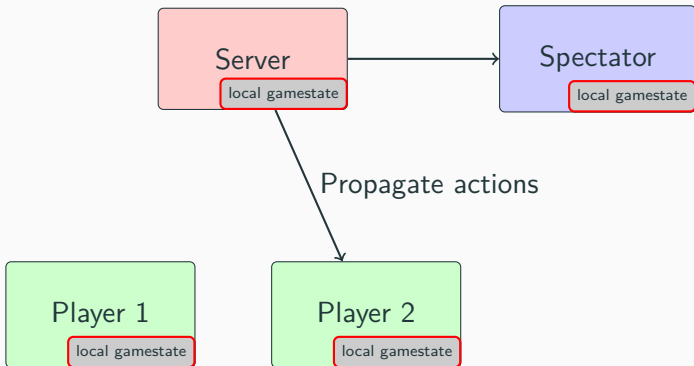


stechec2 architecture

Check actions on
local gamestate



stechec2 architecture



Problem I: Flexibility

We want generic and re-usable code...

We want generic and re-usable code...

...but we need flexibility to have original games

Generic interface:

- `at_start`
- `start_of_player_turn`
- `end_of_round`
- ...

Generic interface:

- `at_start`
- `start_of_player_turn`
- `end_of_round`
- ...

Each type of rules overloads interface functions:

- Synchronous rules
- Turn based rules

Creating a game

Coding a game always starts with a YAML file:

Creating a game

Coding a game always starts with a YAML file:

- Name
- Rules type

Coding a game always starts with a YAML file:

- Name
- Rules type
- Constants
- Enumerations
- Structures

Coding a game always starts with a YAML file:

- Name
- Rules type
- Constants
- Enumerations
- Structures
- Actions functions
- Observers functions

Creating a game

Coding a game always starts with a YAML file:

- Name
- Rules type
- Constants
- Enumerations
- Structures
- Actions functions
- Observers functions

```
stechec2-generator rules prologin2019.yml prologin2019
```

Problem II: Polyglot

The API user interface

Actions functions:

- `deplacer`
- `glisser`
- `pousser`

Observers functions:

- `position_agent`
- `liste.aliens`
- `info_alien`
- ...

The API user interface

Actions functions:

- `deplacer`
- `glisser`
- `pousser`

Observers functions:

- `position_agent`
- `liste.aliens`
- `info_alien`
- ...

Problem: This interface is coded in C++

The API user interface

Actions functions:

- `deplacer`
- `glisser`
- `pousser`

Observers functions:

- `position_agent`
- `liste.aliens`
- `info_alien`
- ...

Problem: This interface is coded in C++

Solution: Use foreign function interface (or simply **FFI**)

Contestant environment

```
stechec2-generator player prologin2019 env
```

Contestant environment

stechec2-generator player prologin2019 env

env/

├─ c/

├─ caml/

├─ cs/

├─ cxx/

├─ haskell/

├─ includes/

├─ java/

├─ php/

├─ python/

└─ rust/

Simple case: C++

```
cxx/  
├── Makefile  
├── prologin.cc  
└── prologin.hh
```

```
#include "prologin.hh"
```

```
void partie_init()  
{  
}
```

```
void jouer_tour()  
{  
}
```

```
void partie_fin()  
{  
}
```

```
#ifndef PROLOGIN_HH
#define PROLOGIN_HH

...

// constants
#define NB_TOURS 100
...

// structures/enums
typedef enum direction {
    NORD,
    EST,
    SUD,
    OUEST,
} direction;
...

// Actions functions
erreur deplacer(int id_agent, direction dir);
...

// Observers functions
int tour_actuel();
...

#endif
```


Using another language

So let's try with Python.

```
python/  
├── api.py  
├── interface.cc  
├── interface.hh  
├── Makefile  
└── prologin.py
```

Using another language

So let's try with Python.

```
python/  
├── api.py  
├── interface.cc  
├── interface.hh  
├── Makefile  
└── prologin.py
```

`api.py` contains basics
(constants, enums, structs)

Using another language

So let's try with Python.

```
python/  
├── api.py  
├── interface.cc  
├── interface.hh  
├── Makefile  
└── prologin.py
```

`interface.cc` enables the
Python code to call C++
functions using FFI

Using another language

Few more examples:

Using another language

Few more examples:

```
caml/  
├── api.ml  
├── interface.cc  
├── interface.hh  
├── Makefile  
├── prologin.ml  
└── prologin.mli
```

Using another language

Few more examples:

```
c/  
├── interface.cc  
├── interface.hh  
├── Makefile  
├── prologin.c  
└── prologin.h
```

Using another language

Few more examples:

rust/

- api.rs
- ffi.rs
- interface.cc
- interface.hh
- Makefile
- prologin.h
- prologin.rs

haskell/

- Api.hs
- CApi.hsc
- interface.cc
- interface_c.cc
- interface.hh
- Makefile
- Prologin.hs

Fun fact

Rust and Haskell were added by contestants themselves!

All the languages come with generated code **and** a Makefile

All the languages come with generated code **and** a Makefile

C++

```
$ make  
cxx    prologin.cc -> prologin.o  
lib    champion.so
```

All the languages come with generated code **and** a Makefile

Python

```
$ make  
cxx    interface.cc -> interface.o  
lib    champion.so
```

Interpreted languages still need to compile the interface

All the languages come with generated code **and** a Makefile

Rust

```
$ make
rustc  prologin.rs -> prologin.o
cxx    interface.cc -> interface.o
lib    champion.so
```

All the languages come with generated code **and** a Makefile

Rust

```
$ make
rustc  prologin.rs -> prologin.o
cxx    interface.cc -> interface.o
lib    champion.so
```

What is this `champion.so` file anyway?

.so stands for **shared objects**

This is a dynamically linked library, used in the matchmaking system.

Launching a match

```
$ make  
cxx    prologin.cc -> prologin.o  
lib    champion.so  
$ make tar  
tar champion.tgz
```

Upload this compressed file to the local finals website.

Launching a match



The screenshot shows the PIB website interface. At the top right, there are links for 'prologin', 'Admin', and 'API'. The main header features the 'PIB' logo and the title 'Envoyer un champion'. On the left, a sidebar menu includes 'Accueil', 'État des serveurs', 'Champions', 'Mes champions', 'Envoyer un champion' (highlighted), 'Tous les champions', 'Matches', 'Mes matches', 'Les matches des mes champions', 'Lancer un match', and 'Tous les matches'. The main content area contains three form fields: 'Nom' (a text input), 'Sources' (a file upload button labeled 'Browse...' with the text 'No file selected.' and 'Archive au format .tgz'), and 'Commentaire' (a text area). Below these fields is a dark button labeled 'Envoyer le champion'. A small cartoon character is visible in the bottom left corner of the page.

prologin Admin API

PIB Envoyer un champion

Nom

Sources No file selected.
Archive au format .tgz

Commentaire

Uploading your champion

Launching a match



The screenshot shows the PIB website interface. At the top right, there are links for 'prologin', 'Admin', and 'API'. The main header features the 'PIB' logo and the title 'Lancer un match'. On the left, a sidebar menu lists various navigation options: 'Accueil', 'État des serveurs', 'Champions' (with sub-items 'Mes champions', 'Envoyer un champion', 'Tous les champions'), 'Matches' (with sub-items 'Mes matches', 'Les matches des mes champions', 'Lancer un match', and 'Tous les matches'). The main content area contains two dropdown menus labeled 'Champion 1' and 'Champion 2', each with a placeholder text '.....'. Below these is a dark button labeled 'Lancer le match'. The background of the main content area features a faint, artistic illustration of a glass with a drink and a straw.

prologin Admin API

PIB Lancer un match

Champion 1

Champion 2

Lancer le match

- Accueil
- État des serveurs
- Champions
 - Mes champions
 - Envoyer un champion
 - Tous les champions
- Matches
 - Mes matches
 - Les matches des mes champions
 - Lancer un match
 - Tous les matches

Launching a new match

Launching a match

The screenshot displays the PIB website interface for a match. On the left is a navigation menu with links: Accueil, État des services, Champions, Mes championnes, Revue des championnes, Tous les championnes, Matches, Mes matches, Les matches des mes championnes, Lancer un match, and Tous les matches. The main content area is titled 'Détail du match #10'. It features a 'Match' box with details: ID 5, État 'en cours', Date 'today à 01:01:40', and a 'play' button. To the right is a 'Replay' section with a 'Télécharger' button. Below these is a 'Participants' table with columns: ID player, ID champion, Champion, Score, and Log. The table is currently empty. At the bottom is a 'Replay' section showing a game board with various pieces. A status bar at the very bottom displays match statistics: Score 1, Score 2, Total 0, and a list of pieces: 0 Roi de diables, 0 Chaperon (piece 1), and 0 Chaperon (piece 2).

PIB

Détail du match #10

Match

ID 5
État en cours
Date today à 01:01:40
play

Replay

Télécharger

Participants

ID player	ID champion	Champion	Score	Log
-----------	-------------	----------	-------	-----

+ Afficher le log serveur

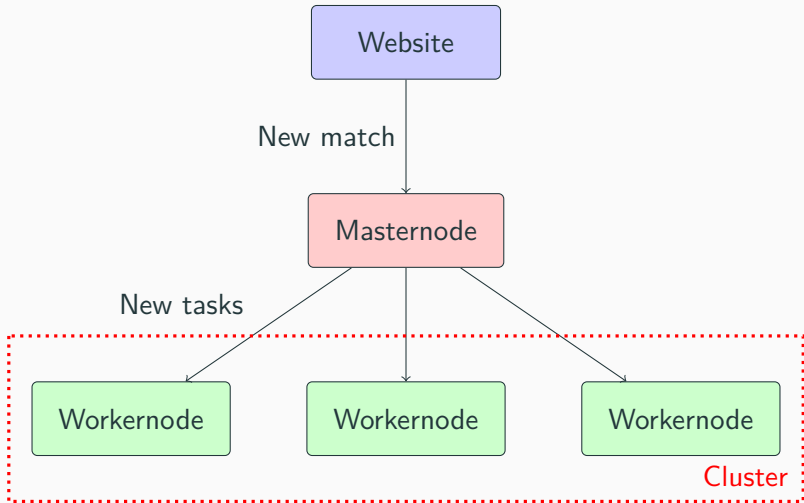
Replay

Score 1 Score 2 Total 0
0 Roi de diables
0 Chaperon (piece 1)
0 Chaperon (piece 2)

Match replay

Problem III: Distributed

Launching a match II



Masternode **divides tasks among workernodes.**

Masternode **divides tasks among workernodes.**

A workernode **completes a task.**

(compiling a champion, or running a match)

Communication between masternode and workernodes uses:

- Remote procedure calls (**RPC**)
- Asynchronous methods

Problem IV: Isolated

Need for isolation

- We **do not want** to execute unknown code on our machines
- Candidats have time and memory limits for their AI

Need for isolation

- We **do not want** to execute unknown code on our machines
- Candidats have time and memory limits for their AI

Workernode's tasks are **always** running in isolated mode.

`https://github.com/ioi/isolate`

*a sandbox built to safely run untrusted executables,
offering them a limited-access environment and
preventing them from affecting the host system.*

`https://github.com/ioi/isolate`

*a sandbox built to safely run untrusted executables,
offering them a limited-access environment and
preventing them from affecting the host system.*

Built for IOI (International Olympiads in Informatics)



camisole is:

- Our own secure online judge
- Running isolate as a backend

<https://github.com/prologin/camisole>

Conclusion

<https://prologin.org>

<https://github.com/prologin/stechech2>

<https://github.com/prologin/sadm>

<https://github.com/prologin/prologin2018>

<https://prologin.org>

<https://github.com/prologin/stechech2>

<https://github.com/prologin/sadm>

<https://github.com/prologin/prologin2018>

info@prologin.org

[#prologin @ irc.freenode.net](https://irc.freenode.net/#prologin)

Questions?

Thanks for listening!

Thibault 'haltode' Allançon
`thibault.allancon@prologin.org`