How team «Snakes vs Lambdas» was ~~writing an AI to conquer the galaxy~~ programming all the weekend and got to top30 at icfpc-2011

Foreword

At long last I found time to write this report (actually it become slightly bigger that I expected :), but nevertheless I had a lot of fun writing it). I decided to write it in English because of the two reasons:

- More people will be able to read it (so that will probably popularize the greatest contest ever: icfpc)
- It is a good opportunity to improve my written English, which in fact is not good as I wanted it to be (so if you see mistake – feel free to write me about it)

Introduction

In this report I'll try describe the strategy of our team and share some of my experience and thoughts about ICFPC–2011. The previous year report (in Russian) can be found here.

For those, who don't know, ICFPC = [International Conference][Functional Programming][Contest]. Captain Obvious already guessed that it is a contest dedicated to the annual conference about functional programming. So every year a university, which holds a conference, prepares hard and challenging programming task. I've heard that two years ago it was calculating complex trajectory of spacecraft. Last year we were solving great puzzle about cars, fuels, factories, prefix ternary decoding, finite automata, nonlinear equations and lots of other stuff. This year it was an AI development for card game "Lambda: the Gathering" ("L:tG" for short) – a crazy and complicated mix of MtG and functional programming.

"I always knew Magic: the Gathering is a combo of Lisp and Brainfuck"

@dallaylaen

If you want to read more about previous ICFPC tasks, I would recommend to have a look at adept's reports (he writes very interesting and thrilling). (For example: 2006, 2007, 2010.)

Well, I think it's enough for introduction.

The Team

Last year I was in a team "SnakeTeam" (the name was influenced by the power of programming language Python), which showed very good performance (15th place). So we decided to gather the same team and use exactly the same programming environment this year (and maybe invite somebody else, because we felt that 3 is not the optimal team size number) .

Some facts

Team size:	3 (mainly) + 2 (partly)
Team member location:	gathered at one room with lots of food
Preliminary Rank:	SnakeZombie #41 – #61 Lambdas vs Zombies #10 – #23
Final Rank:	Snakes vs Lambdas – top30 (out of 199 teams) (other information will be revealed at ICFP 2011 in September 19–21, Tokyo)

:
I put our source codes at github for those who are interested in.

While writing these lines I realized that this report will definitely contain over 9000 of words, so feel free to skip some sections if you are not interested in them :).

Now I'll try to describe the sequence of events in a chronological order.

Day -2 (June 15, Wednesday) Hint on the main page

SkidanovAlex mentioned on codeforces.com that there is some secret about image on the main page of the contest website.

Googled the image I found out that it's a "pngjar" (whatever it means) image. So after renaming it to *.jar I was able to open it with archive manager. Decompiled Q.class I made sure that it just write some bytes to stdout. Then I executed the command

```
$ java -jar new_books.png > new_books2.png
```

and got the following image:

That gave a hint that the contest task will be somehow connected with card game "~~Magic~~ Lambda: the Gathering".

The amazing thing is that new_books2.png is also a "pngjar", that generates the original image.

```
$ java -jar new_books2.png > new_books3.png
$ md5sum new_books.png new_books3.png
     ef1feaa66e31d6765cf47cce93413613  new_books.png
     ef1feaa66e31d6765cf47cce93413613  new_books3.png
```

Impressed by that and rejoiced by the connection with MtG (which I used to play a lot some years ago), I was waiting for the beginning of the contest twice more.

Day -1 (June 16, Thursday)
Fight with Debian

In the evening I took my backpack with clothes (I was leaving Moscow right after the contest), 2 laptops, 2 sleeping bags and get to Alexey's (aka taras) apartments by midnight [from here and till the end all times are given in Moscow Time (GMT+4)] .

Contest was going to begin at 04:00, but we decided that to sleep is better than to wait for the beginning; as according to pieces of information from organizers, scores will be given only for final submission after contest ends, so submission time is not as important as it was previous year.

Judges also recommended to have some machine (physical or virtual) running Debian Squeeze. I wanted to install Debian Squeeze as OS on my old laptop, but burning of 2 CDs, several attempts to load from usb stick and googling howto-s didn't help much. So I gave up and easily installed Ubuntu Lucid Lynx. (Now I'm not surprised that Ubuntu is much more popular on desktops than Debian ^_^)

Day 0 (June 17, Friday)
The Contest Starts

I woke up and the first thing I did was reading problem description. The task was quite clear - to write an AI for some kind of turn-by-turn deterministic game with full information (named Lambda the Gathering).

Comparing to the previous year (in which you almost have to hack using the machete through the jungle of words and meanings just to understand what is the destination of your journey) this year's task was a straight highway with signs where to go all over the way. The main problem was to build a car that could drive that road better than others cars.

The design of the cards was awesome. It followed all the traditions of CCG: cool art, exciting flavour text and even small details (like printing number of arguments of function in the right upper corner) – everything was done with love and carefully. I especially like «copy» card (though we didn't get how to use it).

Full spoiler of the first edition is available at
http://www.thenewsh.com/~newsham/icfp/2011/cards.html

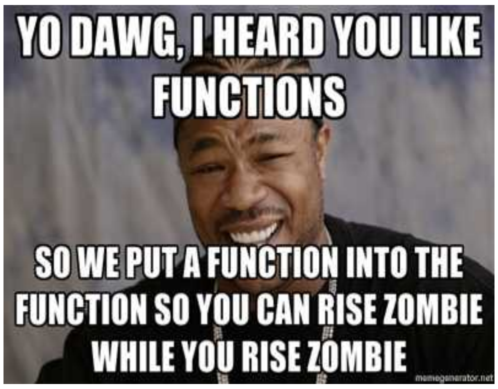
Just a few words about Lambda the Gathering (for those who didn't read problem description). It is a game for two players. Each player has 256 slots. Each slot has vitality points and contains a function. There are several types of cards (each of them also represent some function). You have an unlimited access to all of the cards and during your turn can play one of them (to play means to apply function in slot to card's function or vice versa). Your goal is to kill (decrease the vitality to zero) all of the opponents slots.

While I was waiting for other team members to come I understood example sessions and played a couple of games in a interactive mode. These sessions showed how composition of functions S(K(f, g)) (further I will write f o g for it) and simple recursion could be implemented. It was great that judges provided binaries with the interactive mode in it, because that gives you possibility to test you understanding of the game rules.

Oleg (aka Dr.Korbin) arrived at about 14. We discussed the task and came to decision that any strategy that wants to get to at least top30, should have a full game state in memory. However, it is forbidden to use judges' binaries as a part of the submission and the only information your strategy can get from the outside is your opponent turns. So it was a good idea to implement an emulator of the game based on the given specification. Oleg started to write ltg_emulator.py, while I was thinking of any strategy that will kill opponent before the turn 100 000.

There was a leaderboard on the «unofficial» duel server: teams were sorted by the number of wins among last 30. We found several teams known by last year contest:

- shinh3 – probably the owner of the golf competition site golf.shinh.org
- caml riders – just cool name (reminds me famous song «riders of the storm»)
- THIRTEEN – guys from kharkiv
- 6/11 – from Kaspersky lab


YO DAWG, I HEARD YOU LIKE FUNCTIONS
SO WE PUT A FUNCTION INTO THE FUNCTION SO YOU CAN RISE ZOMBIE WHILE YOU RISE ZOMBIE

and others. But there was a team with the name I really liked: «I heard you like functions».

Alexey returned at about 15, made some assumptions about possible game strategies and went to sleep pleading illness (he had temperature at about 38).

Here are some details of our implementation of the game emulator.
We represent functions as python tuples. (It was more convenient to consider multi-variable functions in a usual way, but not as a function "..which (when applied) will take another argument and return yet another function, which (when applied) ... and so on") So the first element of tuple is the type of function ("get", "help", ...), whereas other elements are arguments of that function. In LTG lazy evaluations are used, therefore in our implementation we don't compute the arguments of function until it is fully defined (given all the arguments).

"I think the design of this contest was influenced by the desire to stop having C++ programmers win" #icfpc2011

sully

By the evening I found simple recursion (based on the example sessions) that helps 0 slot until it reaches maximum vitality. It looks something like

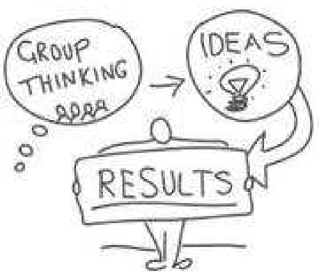
S(get o K(0), help(0)(0) o K(N))

where N – is the large number greater than 10. But unfortunately I didn't know how to translate this formula to ltg turns, because I could use composition (SK combo) only for the first argument.

While discussing the task we were also thinking about possible name of our team. Last year name "SnakeTeam" was too simple and not cool enough to use :). New name of course must contain the word "snake" (in the name of the great python ^_^). I proposed something like "Snake Strikes Back", but we agreed on "SnakeZombie". (Alexey being ill was like a zombie in our team :), and besides "zombie" card in ltg seems to be very powerful. As if you combine it with help – it will damage two of your opponent cells for free, whereas using only attack costs you more vitality than is decreased from your opponent).

My friend Veniamin joined our team at the late evening. He hadn't participated in any ACM–topcoder–like stuff, until two weeks ago – when we take him in our team "MSU Chapelnik" and he showed great performance in solving problems at ISPC. So expecting this ICFPC to have more of a thinking than of a programming I had asked him to come. Besides he is a roommate of a ACM ICPC world finals 2011 bronze medallist, so probably it would help too :).

We discussed possible strategy with him and some details about game rules and then he started to think about SKI combos.



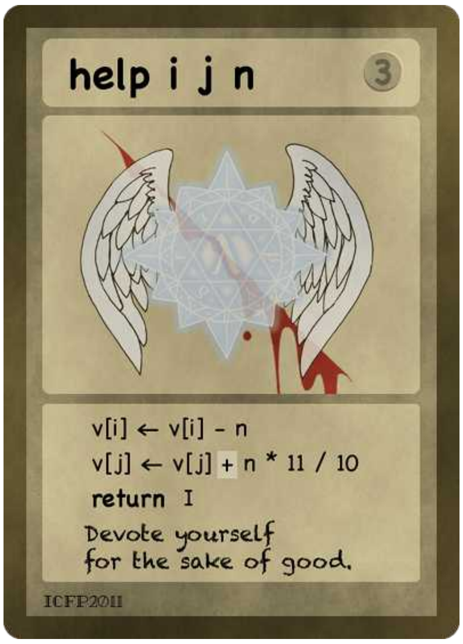
I was experimenting with emulator. But after some time I was fed up with making moves by hands (one wrong move and you have to start the whole combo from the very beginning) so I created simple text_bot.py, which performs a sequence of turns from a file "cmd.txt". Soon I felt necessity of some macros, like create number N at slot i.
So I wrote a function gen_num that generated such sequence of turns, put it into botlib.py, and added option to evaluate python functions in text_bot.py. Later I wanted to make more complex tasks and we started to think about future architecture of such bots. Oleg proposed a genuine and at the same time simple idea: the bot should be a python program that generates and executes moves.

So more useful functions were added to botlib.py (and it almost reached its final version).
Also for future testing we wrote zero_bot.py – a bot that do nothing.
At about 4, I felt exhausted and went to sleep, but Oleg and Venya continued working on task.

Day 1 (June 18, Saturday) Breakthrough

I woke up at 10 (all other guys were sleeping) and after having breakfast and spending some time of thinking about infinite help I got an insight.

Help combo



help i j n
3
v[i] ← v[i] - n
v[j] ← v[j] + n * 11 / 10
return I
Devote yourself for the sake of good.
ICFP2011

I found that formula:
g = S(S(f, get), I), where

f = help(0)(0) o get o succ

It could be easily seen that

f(0) =
= help(0)(0) (get(1)) =
= I + (effect of help)

So put any big number N at slot 1 and function g at slot 0 and consider the following:

g (0) =
= (S(f, get) (0)) (I (0)) =
= (f(0) (get(0))) (0) =
= (I(g)) (0) = g (0) = ...

On each iteration we increase the life of slot 0 by 0.1 * N. According to ltg rules infinite loops are stopped after 1000 calls of functions — so function help will be launched about 80 times. And if N is at least 8000 then slot 0 will have maximum vitality (65535) after such combo.

Another obvious observation is that we don't need to build this combo every time. We can store function g in slot i and copy it by using card "get". So the time needed to one call of this help combo could be reduced to 2 + Compl(i) turns, where Compl(i) — is the complexity of slot i (number of turns needed to get i at some slot).

That combo revealed a bug in ltg_emulator.py. Doing this combo with N = 16 was supposed to gave us 83 vitality points, whereas our emulator showed increase only by 70 points. That meant that the way we implemented a restriction of 1000 function calls was wrong. Oleg as a creator of that code started investigating this bug.

The finding of help combo also opened a way to creation of bots which are able to kill opponent. I didn't found good attack function right away. That's why I used straight forward method:

```
f = attack (0)
g = (f o get o succ o succ) (0)
h = (g o get o succ)
```

So

```
h(0) = attack(0) (get(2)) (get(1))
```

The only thing that should be done before doing all this stuff -- is to put target of our attack at slot 2.

I wanted to attack slots starting from 255 to 0. But Alexey (main strategy expert in our team) advised to make first attack at slot 0, because if they do something similar to our strategy – then slot 0 is critical for them, so shot at it will ruin their plans and we can start unhurry destruction of their slots.

Then I wrote prototype "example_attack.py", tested it with zero_bot.py (it wins after 6695 turn) and Oleg submitted it to the server (that happened at about 15). Our bot used simple way of making moves: at first it generated over 100k turns in a list and then simple function "execute(list)" mixed writing to stdout generated sequence with reading opponent turns.

"someone should write the code (most part of it), someone should construct cool SKI terms, someone should develop cool gaming strategy, coevolution etc"

ulidtko

At about this time one more Alexey (Alexey-II) came. We gave him task to improve infrastructure and make a testing framework. While doing that he found some critical bugs in ltg_emulator that caused it to fail during execution.

After submitting our first bot we moved to the next task: adding brains (emulator) to our bot. But at first we discussed refactoring of the botlib.py, because it was impossible to write interactive bot, using that framework.

We agreed to have to following hierarchy of classes:

```
    Bot
      | -> MacroStrategy
            |-> MicroStrategy
                    |-> Game
```

- Game contains state of the game and changes it by taking the moves from MicroStrategy.
- MicroStrategy is responsible for short-term goals: create help function, rebuild combo, and so on.
- MacroStrategy is responsible for long-term plans and can uses functions from MicroStrategy as building blocks.

(To be honest, Oleg later admitted that such structure wasn't very convenient and almost everything was contained in MicroStrategy)

Also we decided to rewrite most of the functions from botlib.py using iterators to avoid memory problems.

By 8 o'clock Veniamin and Alexey-II had left us. They didn't have enough time to fully understand the problem, but discussing problem with them was quite useful.

By 10 I wrote TopCoder Round 1: solved 2 problems with not very good time, got +13 to rating and advanced to the next round. (Last year I simply forgot about this round - I was busy with ICFPC-2010 related stuff, but this time I was smarter: I set up an alarm on the time of contest)

Программист работает.
Программист празднует день программиста.
http://iampop.livejournal.com
Copyright 2009 Ivan Ershov

The rest of the evening I was struggling with attack function, because the one we used was too slow. I was trying to find function a: such that

$$f(0) = attack(0)\ (get(i))\ (get(j))$$

And here comes the second insight.

Attack combo

Let

$f = attack(0)$

$g = f \circ get \circ succ$

$Att = S(g, get) \circ succ$

then it's easy to see, that

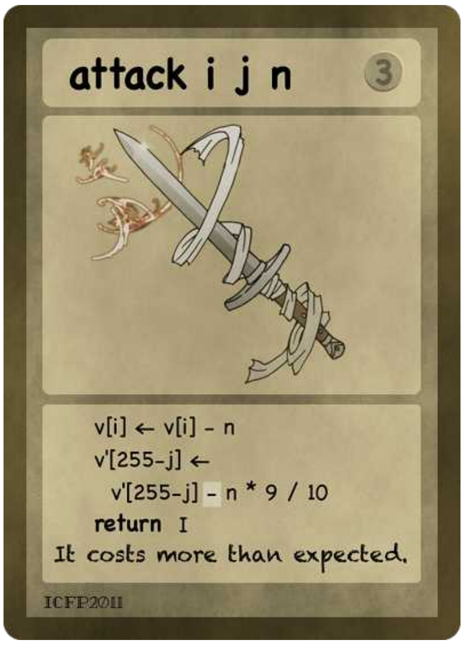
$Att\ (0) =$

$= S(g, get)\ (1) =$

$= g\ (1)\ (get(1)) =$

$= f\ (get(2))\ (get(1)) =$

$= attack(0)\ (get(2))\ (get(1))$



So again, we can store function h at some slot, target of our attack at slot 2, and the amount of attack at slot 1 (it is already there – we used it in help combo). The complexity of one attack is the same as of one help (6 turns if h is stored at slot 4).

By that time Oleg refactored the source and almost finished our production branch: "example_attack.py with brains". It had very good improvement (thanks to Alexey who was constantly thinking about possible strategies) bot was using the following heuristic, when choosing the best target to attack:

1. filter alive slots
2. filter slots with function not equal to "I" in it
3. filter slots that we are able to kill (vitality < 16000)
4. choose the slot without heal function in it (first we want to kill attack combo, then heal)

If at any step we got an empty set of slots, then bot attacked at any slot from previous step.
This bot was submitted and Oleg started to add help-recovering feature. (If our slot where we store help function was destroyed - than we should build it in another slot)

At the same time I was doing usual development process (beta -> rc -> release) and replaced slow attack function with the fast one. At about 1 a.m. we had bot "fast_attack.py" ready to be submitted. We decided to send it under another name "Lambdas vs Zombies" in order not to stop the test of our production.

Alexey had found several useful optimizations that could be done. I was too tired to code it, so I switched to voice programming (I was telling Alexey what to write), which was actually very efficient, because when you don't care about what to type - you have more time to think about algorithm.

О, я знаю как решать эту задачу! Тут все очевидно. В общем, пиши Укконена.
Сам пиши.
ACMGeeks001
A.Ripatti©2010

Alexey believed that duels in ltg are similar to duels of cowboys in Westerns. Winner is the cowboy, who shots earlier than other. So the key to victory in ltg is to make the first attack as early as possible. We did it pretty soon (at about 70th turn), but according to duel results attack at slot 0 seems to be not lethal, so while we were killing the opponent from 255 slot to 0 - he rises from the knees and kills us. Thereby we made the following optimizations:

1. changed the pattern of our attack strategy [0, 255, 255, ..] to [3, 4, 4, 8, 8, 16, 32, 64, 128, 255, 255, ...] (list of numbers represents the first opponents slot, which slot we started our attack from; e.g. first series of shots is from 3 to 0, second: from 4 to 0, and so on). This increased the time needed to fully kill opponent, but also increased stability of the win.

2. changed the main strategy from [help, attack] to [help, attack, attack] (we had 65535 vitality in slot 0 and shot by 16K, so we have enough vitality to make 2 shots instead of one). This increased the attack rate.

3. optimized the creation of numbers 252 and N (N > 8000). Before, we build 8192 and 252 separately. Now, we build number 252, then copy it to another slot and by doubling it several times we create some number N (N > 8000). This postponed the turn of first help, but brought closer the turn of first attack.

This very fast optimized bot beats our bot with brains. But we believed that not-bad-written bots with game emulators in it are better than any bot without it.

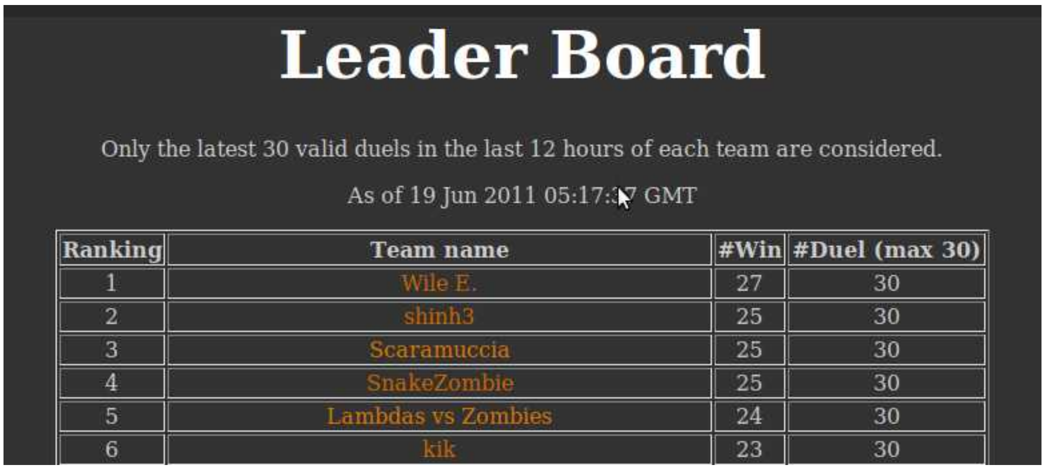
Before going to sleep I realized that we should use the name "Snakes vs Lambdas". Actually it means what we did during contest: struggle with lambda calculus, using python. :)

Day 2 (June 19, Sunday)
Rise of the zombies

"Today any army of zombies will rise, driven by a desire to reduce SK combinator terms to unnatural forms.." #icfpc2011

@TotallyToRA

Waking up in the morning I was very happy and surprised to found that both our bots get to top5.



Leader Board

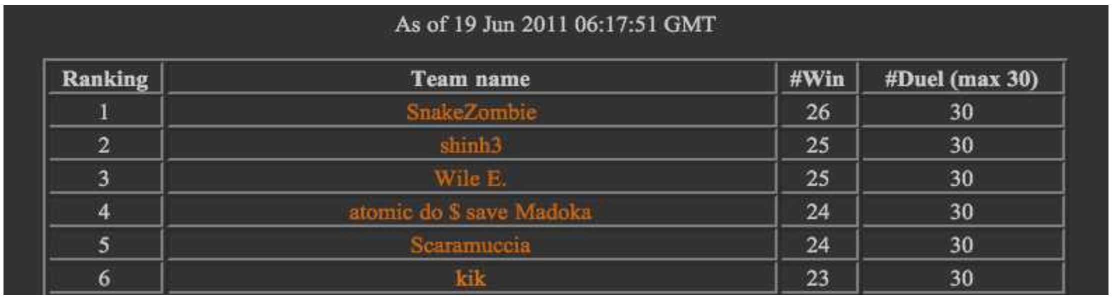
Only the latest 30 valid duels in the last 12 hours of each team are considered.

As of 19 Jun 2011 05:17:37 GMT

Ranking	Team name	#Win	#Duel (max 30)
1	Wile E.	27	30
2	shinh3	25	30
3	Scaramuccia	25	30
4	SnakeZombie	25	30
5	Lambdas vs Zombies	24	30
6	kik	23	30

Also I found message from Oleg that before going to sleep he implemented several recovery features (help function, numbers) and now bot with brains and slow attack beats our very fast and optimized bot. That was good news.

One hour late there was even a moment of glory when our main bot was at the first place.



As of 19 Jun 2011 06:17:51 GMT

Ranking	Team name	#Win	#Duel (max 30)
1	SnakeZombie	26	30
2	shinh3	25	30
3	Wile E.	25	30
4	atomic do $ save Madoka	24	30
5	Scaramuccia	24	30
6	kik	23	30

Oleg had lots of things to do: merge development branch (fast_attack.py) and production(bot with brains).

As for me – I wanted to find more cool SKI combos. (Some of the duel results showed very fast win (in about 500 turns) – perhaps somebody had found machinegun combo)

Suddenly :) we realized that function in the slot will remain if the slot dies. That's why revive happened to be very powerful card, as it is very cheap to revive some of the low slots and when you did so – you'll get your function back. One of the way to deal with it is to replace function in that slot by another, using zombie card.

"Lambda: The Gathering clearly doesn't follow the #mtg color pie: you counter spells by combining direct damage and zombies." #icfpc2011

@xlerb

Zombie combo

zombie i x
2
f'[255-i] ← x
v'[255-i] ← -1
return I
The dream went to nightmare.
ICFP2011

Some time later I found zombie combo:

```
f = S(zombie, put)
Zmb = f o get o succ o succ
```

So

```
Zmb (0)
= f (get (2))
= zombie (get(2)) (I)
```

will put function "I" at slot get(2).

Also I found a way to combine attack combo and zombie combo at one shot. Let Zmb be at slot 3 (actually we put it in slot 5), consider

```
f = S(Att) o get o succ o succ o succ
AttZmb = S(f, I)
```

then

```
AttZmb (0) =
= (f(0)) (0) =
= S (Att, get(3)) (0) =
= S (Att, Zmb) (0) =
= Att(0) (Zmb(0))
```

will attack slot and then put zombie in it. We replaced Att function with AttZmb. (But we don't want to loose speed, so first series of shots (from slot 3 to 0) is done by usual attack combo)

By the evening with tremendous efforts Oleg using all the power of vim pushed all this combos and a lot of other cool heuristics in production (I don't know the details of it, because for me code of our bot-with-brains seems write-only though it was implemented in python).

We found out that our clever bot lost some duels due to exceeding memory or time limit. (We believed that our algorithm is fast enough and the problem was with memory limit). So we reduced internal memory limit to 300 mb (in case OS kills our process earlier than we get MemoryError exception) by command s

```
import resource
LIM_MB = 300
MEGS = 2**20
resource.setrlimit(resource.RLIMIT_AS, (MEGS * LIM_MB, -1L))
```

but actually this didn't help (even now I don't know why).
Also we tried using the following hack with sizeof (we decided not to remember too big function):

```
if sys.getsizeof(new_slot) > 100000:
     new_slot = ("I",)
```

At 0 o'clock (when only 4 hours of contest were left) I had to leave (had tickets to train to Nizhniy Novgorod). So we discussed which of the two bots is better to submit (development or production branch):

1. blind, fast, dirty, stupid, that was extremely stable and seems to contain no bugs.
2. get very high place but also might not get to top30.

We decided that if we have any chance to get to top2 – we should risk and try to use it (because only top2 gets valuable prizes).

Conclusion

I really enjoyed the contest, especially the competitive part of it. When you trying to guess what is the strategy of your opponents and then you are thinking about counter-strategy.

I tried to remember the things that I didn't like about the contest but I couldn't remember any. Guys from Tokyo did a great job to prepare ICFPC!

The task itself was also really cool. It was quite simple but at the same it gives you

unlimited freedom in developing and creating solutions because there was no "right solution".
I also like that time wasn't a great factor here (comparing to the previous year), you could get the first working bot almost at the end of contest but you still have a chance to win.

I regret that I didn't have enough time to create more cool SKI terms. (Some teams have created combo that kills really fast). Maybe we would achieve better place if we found more combos. Nevertheless getting to top30 is a good achievement.

Great respect and thanks to:

- my team for participating and being patient with me
- organizers for preparing and holding such a cool contest
- Guido van Rossum for python (which is really awesome to code for fun)

Participate in ICFPC and have fun! see you next year!