Question about the 'intelligent algorithm' of membership testing (for permutation group) in demo 2.

6 messages

Jae Hyun Lim <jae.hyun.lim@umontreal.ca> Sat, Oct 14, 2017 at 5:29 PM
To: Maëlle Zimmermann <maelle.zimmermann@gmail.com>

Dear Maëlle Zimmermann,

Hi, my name is Jae Hyun Lim, and I'm a graduate student taking the algorithm class of Professor Pierre McKenzie's in this semester.

First of all, I apologize for sending an email to you on the weekend.

I have a question regarding the 'intelligent algorithm' for permutation groups in the demo 2. While it is described (in page 37 of the first presentation file) to run insert for all combinations of (q, r) in table T, the python implementation only considers the (q, r)s, combinations of the newly added permutations and the input permutations. While I tried many times to understand the equivalence between them, I couldn't figure it out.

Regarding this, I would really appreciate if you could help me to understand why the python implementation can be equivalent to the pseudocode in the presentation file.

Thank you again for your helps!!

Sincerely,

Jae Hyun Lim

--

Jae Hyun Lim

PhD student
MILA, University of Montreal

+1-438-979-5819 (mobile)
jae.hyun.lim@umontreal.ca

Jae Hyun Lim <jae.hyun.lim@umontreal.ca> Sat, Oct 14, 2017 at 9:27 PM
To: Pierre McKenzie <mckenzie@iro.umontreal.ca>
Cc: Maëlle Zimmermann <maelle.zimmermann@gmail.com>

Dear Professor Pierre McKenzie,

Hi, my name is Jae Hyun Lim, and I'm a graduate student taking your algorithm class in this semester.

First of all, I apologize for sending an email to you on the weekend.

I have a question regarding the 'intelligent algorithm' for permutation groups in the demo 2. While it is described (in page 37 of the first presentation file) to run insert for all combinations of (q, r) in table T, the python implementation only considers the (q, r)s, combinations of the newly added permutations and the input permutations. While I tried many times to understand the equivalence between them, I couldn't figure it out.

Regarding this, I would really appreciate if you could help me to understand why the python implementation can be equivalent to the pseudocode in the presentation file.

Thank you again for your helps!!

Sincerely,

Jae Hyun Lim

--

Jae Hyun Lim

PhD student
MILA, University of Montreal

+1-438-979-5819 (mobile)
jae.hyun.lim@umontreal.ca

Pierre McKenzie <mckenzie@iro.umontreal.ca> Sun, Oct 15, 2017 at 2:28 PM
To: Jae Hyun Lim <jae.hyun.lim@umontreal.ca>
Cc: Maëlle Zimmermann <maelle.zimmermann@gmail.com>

hi, Jae Hyun,

Great question! The story is that the algo in my slides is the original polynomial time algorithm developed by Furst, Hopcroft and Luks. The Python algo that Maëlle gave results from an observation made subsequently by Don Knuth.

Your question goes beyond the scope of the course.

I think the proof that the Python solution works goes by downward induction on k. A sufficient predicate is something like this: at the end of the algorithm, any product of the form t_n * t_{n-1}*...*t_k * t'_n * t'_{n-1}*...*t'_k where the t_i and t'_i come from row i can be written in the same form.

The proof can be plowed out of Knuth's 1991 note entitled "Efficient Representation of Perm Groups"...but his notation is kind of reverse to ours which makes it delicate.

best,
Pierre

[Quoted text hidden]

Jae Hyun Lim <jae.hyun.lim@umontreal.ca> Sun, Oct 15, 2017 at 2:43 PM
To: Pierre McKenzie <mckenzie@iro.umontreal.ca>
Cc: Maëlle Zimmermann <maelle.zimmermann@gmail.com>

Dear Professor Pierre McKenzie,

Thank you so much for your thoughtful comments! It helped me a lot!

If both are correct, I would also appreciate it if you could help me to understand the uniqueness proof of the permutation algorithm at the question 4 of the demo 3.

I ran the programs for both cases, and they resulted in different tables, where some of elements have different permutations (while the positions of the filled elements in tables are the same). I wonder what it means by 'notation is unique' if different shapes of tables are correct.

Thank you again for your time and assistance.

Sincerely,

Jae Hyun Lim

[Quoted text hidden]

Pierre McKenzie <mckenzie@iro.umontreal.ca> Sun, Oct 15, 2017 at 5:27 PM
To: Jae Hyun Lim <jae.hyun.lim@umontreal.ca>
Cc: Maëlle Zimmermann <maelle.zimmermann@gmail.com>

hi,

Consider the group G = < g1, g2, ..., gk >.
Run either algo on g1,...,gk.
Consider the final resulting table T.

Let "canonical expression" mean a product of the form t_n*t_{n-1}*...*t_1, where t_i is either the identity or a nontrivial element from line i in T.

Then the following holds:

1) any g \in G has a canonical expression
and
2) different canonical expressions multiply out to different elements of G.

(2) is all that is meant by "this notation is unique'.

best,
Pierre

[Quoted text hidden]

Jae Hyun Lim <jae.hyun.lim@umontreal.ca> Sun, Oct 15, 2017 at 5:35 PM
To: Pierre McKenzie <mckenzie@iro.umontreal.ca>
Cc: Maëlle Zimmermann <maelle.zimmermann@gmail.com>

Dear Professor Pierre McKenzie,

Thank you so much for your kind responses! I really appreciate it!

Thank you again for your time and have a great day!

Sincerely,

Jae Hyun Lim

[Quoted text hidden]