The Gotoh package

Takuto ASAKURA (wtsnjp)

v1.1 [2017/07/07]

Abstract

This package is an implementation in TEX of the Gotoh algorithm, which calculates biological sequence alignments. The package provides only two user commands: `\Gotoh` for executing the algorithm and `\GotohConfig` for setting various parameters with a key-value list.

1 System Requirements

System requirements of Gotoh are shown bellow.

- TEX engine: any engine
- TEX format: LATEX 2_ε
- Document class: any class
- Required package: xkeyval

2 Loading the Gotoh Package

To use the Gotoh package, load `gotoh.sty` with `\usepackage` command in preamble. No package option is available.

```
\usepackage{gotoh}
```

3 Calculating the Alignment

`\Gotoh` The syntax of `\Gotoh` is shown below.

`\Gotoh[`⟨*key-value list*⟩`]{`⟨*sequence A*⟩`}{`⟨*sequence B*⟩`}`

The command puts the optimal score of the alignment to specified control sequence (default: `\GotohScore`) after executing the Gotoh algorithm, and returns the alignment results to other control sequences (default: `\GotohResultA` and `\GotohResultB`). Note that these assignments are done globally. Using the optional argument, you can change the configuration temporary with the same keys in `\GotohConfig` (see the next section).

4 Configuration

\GotohConfig

You can change various settings and parameters related to this package with `\GotohConfig` command. The command takes a key-value list of the settings as its argument and changes the values locally.

`\GotohConfig{`⟨*key-value list*⟩`}`

4.1 Control Sequences to Store Results

\GotohScore
\GotohResultA
\GotohResultB

Control sequences which `\Gotoh` command return results can be specified with following keys:

`score =` ⟨*control sequence*⟩ `\GotohScore`
: sets the control sequence to store optimal score of the last alignment which calculated by `\Gotoh` command.

`result A =` ⟨*control sequence*⟩ `\GotohResultA`

`result B =` ⟨*control sequence*⟩ `\GotohResultB`
: specify the control sequences to store alignment results respectively corresponding to ⟨*sequence A*⟩ and ⟨*sequence B*⟩ of the arguments of `\Gotoh`.

4.2 Algorithm Parameters

The default value of algorithm parameters which define the scoring are

$$\text{match} = 1,\ \text{mismatch} = -1,\ d = 7,\ e = 1. \tag{1}$$

Parameters match, mismatch define the penalties of a *match* and *mismatch* respectively, and d, e are used in a gap penalty (see section 5.6). The parameters appeared in the above equations are able to be changed with following keys:

`match =` ⟨*number*⟩ `1`

`mismatch =` ⟨*number*⟩ `-1`

`d =` ⟨*number*⟩ `7`

`e =` ⟨*number*⟩ `1`
: set the parameters appear in the above equations.

Though the `\Gotoh` command calculates sequence alignment with standard dynamic programming as default, you can use memoization functions instead. Note that both of the methods produce exactly the same results.

`memoization =` ⟨`true`|`false`⟩ or `memoization` `false`
: If true, use memoization functions to execute the Gotoh algorithm.

4.3 Others

`gap char` = ⟨*character*⟩ . (period)
: is inserted to the alignment results where gaps appear. Note that you have to be careful if using `-` (hyphen) as gap characters because successive hyphens automatically converted to dashes by TEX. In this case, you can specify `\mbox{-}` instead.

`uppercase` = ⟨`true`|`false`⟩ or `uppercase` `false`
: If true, uppercase both ⟨*sequence A*⟩ and ⟨*sequence B*⟩ before executing the algorithm.

5 Algorithm and Implementation

5.1 Required package

Package xkeyval is required to process key-value lists.

```
\RequirePackage{xkeyval}
```

5.2 Messages

`\gth@warn` `\gth@error`

Commands for warning and error messages.

```
\def\gth@pkgname{gotoh}
\def\gth@warn{\PackageWarningNoLine\gth@pkgname}
\def\gth@error{\PackageError\gth@pkgname}
```

5.3 Switches, Registers and Constants

All boolean switches, count registers and dimen registers used by this package are declared here.

`\if@gth@first@` `\if@gth@renain@` `\if@gth@gap@` `\if@gth@gapx@` `\if@gth@gapy@`

A switch `\if@gth@first@` is for a calculation command `\gth@max`, and the other switches are for trace back (in macro `\gth@gotoh`).

```
\newif\if@gth@first@
\newif\if@gth@remain@
\newif\if@gth@gap@
\newif\if@gth@gapx@
\newif\if@gth@gapy@
```

`\gth@tempcnta` `\gth@tempcntb` `\gth@tempcntc` `\gth@tempcntd` `\gth@calc`

Scratch four count registers for general use and one for some calculation routines.

```
\newcount\gth@tempcnta
\newcount\gth@tempcntb
\newcount\gth@tempcntc
\newcount\gth@tempcntd
\newcount\gth@calc
```

\gth@sn This is *not* a counter register but a macro to store ⟨*number*⟩ which is used as 'serial number'.

```
\edef\gth@sn{\number\z@}
```

\gth@min This macro stores a large negative ⟨*number*⟩ which is used instead of $-\infty$. Here \gth@M is a largest constant which can be defined by \mathchardef primitive (in traditional TEX engine).

```
\mathchardef\gth@M="7FFF
\edef\gth@min{-\number\gth@M}
```

\gth@tempdima Scratch a dimen register for general use.

```
\newdimen\gth@tempdima
```

5.4 Defining and Setting Keys

\gth@match \gth@mismatch \gth@d \gth@e These are keys for setting the parameters of the algorithm. The default values which are shown in Equations (1) are set here, and stored in the macros (*not* count registers) respectively.

```
\define@cmdkeys[gth]{config}[gth@]{match, mismatch, d, e}
\setkeys[gth]{config}{match=1, mismatch=-1, d=7, e=1}
```

\gth@score \gth@resulta \gth@resultb The macros to store the result of the algorithms are also stored in internal macros.

```
\define@key[gth]{config}{score}{\def\gth@score{#1}}
\define@key[gth]{config}{result A}{\def\gth@resulta{#1}}
\define@key[gth]{config}{result B}{\def\gth@resultb{#1}}
\setkeys[gth]{config}{
  score=\GotohScore, result A=\GotohResultA, result B=\GotohResultB}
```

\if@gth@memoization \if@gth@uppercase \gth@gap@char Both boolean keys \if@gth@memoization and \if@gth@uppercase are set false as default. The gap char is stored in \gth@gap@char.

```
\define@boolkeys[gth]{config}[@gth@]{memoization, uppercase}[true]
\define@key[gth]{config}{gap char}{\def\gth@gap@char{#1}}
\setkeys[gth]{config}{memoization=false, uppercase=false, gap char={.}}
```

5.5 Utility Commands

\gth@nameuse \gth@name@edef \gth@name@xdef \gth@glet Some primitives to use, define or copy control sequences are wrapped.

```
\def\gth@nameuse#1{\csname gth@#1\endcsname}
\def\gth@name@edef#1{\expandafter\edef\csname gth@#1\endcsname}
\def\gth@name@xdef#1{\expandafter\xdef\csname gth@#1\endcsname}
\def\gth@glet{\global\let}
```

\gth@advance \gth@increment \gth@decrement Commands to treate macros which store ⟨*number*⟩ like count registers.

```
\def\gth@advance#1#2{%
  \gth@calc#1\advance\gth@calc#2\edef#1{\the\gth@calc}}
\def\gth@increment#1{\gth@advance#1\@ne}
\def\gth@decrement#1{\gth@advance#1\m@ne}
```

5.6 Calculation Routines

Since these calculation routines (macros) are *not* full expandable, all commands defined here return the results by storing the value in the counter register `\gth@calc`. This assignments are done locally.

`\gth@max` This command take comma-separated list of ⟨*number*⟩ and returns the max value of them.

```
\def\gth@max#1{%
  \@gth@first@true
  \@for\gth@member:=#1\do{%
    \if@gth@first@
      \gth@calc\gth@member
      \@gth@first@false
    \else
      \ifnum\gth@member>\gth@calc
        \gth@calc\gth@member
      \fi
    \fi}}
```

`\gth@gap@penalty` This command takes l, the length of a *gap*, as its argument and calculates its penalty which defined by following function[1]:

$$g(l) = -d - (l-1)e. \tag{2}$$

In this equation, d and e are parameters of the algorithm, which have default values shown in Equations (1).

```
\def\gth@gap@penalty#1{%
  \gth@calc#1\relax
  \advance\gth@calc\m@ne
  \multiply\gth@calc-\gth@e
  \advance\gth@calc-\gth@d}
```

5.7 Printing Matrices

`\gth@print@matrix` This command shows the matrices M, I^x or I^y (See section 5.8). The argument of this macro specifies which matrix to show, so you can detect `m`, `ix`, or `iy`. However, this macro is currently only for the author of this package to debug.

```
\def\gth@tab#1{%
  \bgroup
    \sbox\z@ 0%
    \expandafter\gth@tempdima\ht\z@
    \multiply\gth@tempdima 8%
    \mbox{\hbox to\gth@tempdima{\hss #1}}%
  \egroup}
\def\gth@print@matrix#1{%
  \bgroup\ttfamily
```

[1] This form of gap penalty is called 'Affine gaps'.

```
    #1:\\
    \gth@tempcnta\z@
    \@whilenum\gth@tempcnta<\gth@m\do{%
      \gth@tempcntb\z@
      \@whilenum\gth@tempcntb<\gth@n\do{%
        \gth@tab{%
          \gth@nameuse{\gth@sn @#1@\the\gth@tempcnta
                       @\the\gth@tempcntb}}%
      \advance\gth@tempcntb\@ne}\\
    \advance\gth@tempcnta\@ne}%
  \egroup}
```

5.8 Executing the Gotoh Algorithm

`\Gotoh` This is the user interface to execute the Gotoh algorithm. First, the 'serial number' is incremented. The whole other process executed by this macro contained in a group because a number of temporary macros are defined during processing.

```
\newcommand{\Gotoh}{%
  \gth@increment\gth@sn
  \bgroup  % \egroup is in \gth@gotoh@pre
  \@ifnextchar[{\gth@gotoh@setkeys}{\gth@gotoh@pre}}
```

`\gth@gotoh@setkeys` `\gth@gotoh@pre` These inner macros are for preparing: `\gth@gotoh@setkeys` process temporary configuration specified with the optional argument of `\Gotoh`. In `\gth@gotoh@pre`, if the boolean switch `\if@gth@uppercase` is true, the arguments are processed by `\uppercase` preamble before passing them to `\gth@gotoh` macro.

```
\def\gth@gotoh@setkeys[#1]{\setkeys[gth]{config}{#1}\gth@gotoh@pre}
\def\gth@gotoh@pre#1#2{%
  \edef\gth@tmpa{\noexpand\gth@gotoh{#1}{#2}}%
  \if@gth@uppercase
    \uppercase\expandafter{\gth@tmpa}%
  \else
    \gth@tmpa
  \fi\egroup}
```

`\gth@gotoh` This macro actually executes the Gotoh algorithm [2]. The input of the algorithm is two biological sequences

$$A \equiv a_1 a_2 a_3 \dots a_m,\ B \equiv b_1 b_2 b_3 \dots b_n$$

where a_i and b_j are chosen from a finite alphabet, e.g. $\{\mathrm{A}, \mathrm{T}, \mathrm{G}, \mathrm{C}\}$. The command takes these sequences as its arguments, and calculate following recurrence formula[2] to get optimal score of the algorithm:

$$M_{i+1,j+1} = \max\left\{M_{ij}, I^x_{ij}, I^y_{ij}\right\} + c_{ij} \tag{3}$$

[2]This calculation is done in $O(mn)$ time.

where

$$I^x_{i+1,j} = \max\left\{M_{ij} - d, I^x_{ij} - e, I^y_{ij} - d\right\}, \tag{4}$$
$$I^y_{i,j+1} = \max\left\{M_{ij} - d, I^y_{ij} - e\right\}, \tag{5}$$

and c_{ij} is a score for a pair (a_i, b_j), namely

$$c_{ij} = \begin{cases} 1 & \text{if } a_i = b_j \quad \text{(match)} \\ -1 & \text{otherwise} \quad \text{(mismatch)}. \end{cases}$$

Note that the Equations (4) and (5) have asymmetric form because the order of inserting *gaps* when alternating A and B does not affect the optimal score.

5.8.1 Getting sequences

`\gth@m` `\gth@n` Here each calacters a_i, b_j in the input sequences are stored in `\gth@seqa@`a_i and `\gth@seqa@`b_j, and the lengthes of the sequences m, n are stored in `\gth@m` and `\gth@n` respectively.

```
\def\gth@gotoh#1#2{%
  \gth@tempcnta\z@
  \@tfor\gth@member:=#1\do{%
    \gth@name@edef{seqa@\the\gth@tempcnta}{\gth@member}%
    \advance\gth@tempcnta\@ne}%
  \advance\gth@tempcnta\@ne
  \edef\gth@m{\the\gth@tempcnta}%
  \gth@tempcntb\z@
  \@tfor\gth@member:=#2\do{%
    \gth@name@edef{seqb@\the\gth@tempcntb}{\gth@member}%
    \advance\gth@tempcntb\@ne}%
  \advance\gth@tempcntb\@ne
  \edef\gth@n{\the\gth@tempcntb}%
```

5.8.2 Initialization

We have to be careful with initialization because many implementations of the Gotoh algorithm have problems here [1]. Specifically, the package initialzes the matrxes with following equations:

$$M_{i0} = -\infty, M_{0j} = -\infty, M_{00} = 0,$$
$$I^x_{i0} = -g(i), I^x_{0j} = -\infty,$$
$$I^y_{i0} = -\infty, I^y_{0j} = -g(j).$$

Note that function $g(l)$ is a gap penalty (shown in Equation (2)) and can be calculated with the macro `\gth@gap@penalty`.

```
  \gth@tempcnta\z@
  \@whilenum\gth@tempcnta<\gth@m\do{%
    \gth@gap@penalty{\gth@tempcnta}%
```

```
    \gth@name@xdef{\gth@sn @m@\the\gth@tempcnta @0}{\gth@min}%
    \gth@name@xdef{\gth@sn @ix@\the\gth@tempcnta @0}{\the\gth@calc}%
    \gth@name@xdef{\gth@sn @iy@\the\gth@tempcnta @0}{\gth@min}%
    \advance\gth@tempcnta\@ne}%
  \gth@tempcntb\z@
  \@whilenum\gth@tempcntb<\gth@n\do{%
    \gth@gap@penalty{\gth@tempcntb}%
    \gth@name@xdef{\gth@sn @m@0@\the\gth@tempcntb}{\gth@min}%
    \gth@name@xdef{\gth@sn @ix@0@\the\gth@tempcntb}{\gth@min}%
    \gth@name@xdef{\gth@sn @iy@0@\the\gth@tempcntb}{\the\gth@calc}%
    \advance\gth@tempcntb\@ne}%
  \gth@name@xdef{\gth@sn @m@0@0}{\number\z@}%
```

5.8.3 Memoization

If the switch `\if@gth@memoization` is true, memoization functions (See section 5.9) are called recursively.

```
  \if@gth@memoization
    \gth@decrement\gth@m \gth@decrement\gth@n
      \gth@memo@ix{\gth@m}{\gth@n}%
      \gth@memo@iy{\gth@m}{\gth@n}%
      \gth@memo@m{\gth@m}{\gth@n}%
    \gth@increment\gth@m \gth@increment\gth@n
```

5.8.4 Dynamic Programming

To fill matrxes M, I^x, and I^y the package use the recurrence formula

```
  \else
    \gth@tempcnta\@ne
    \@whilenum\gth@tempcnta<\gth@m\do{%
      \gth@tempcntb\@ne
      \@whilenum\gth@tempcntb<\gth@n\do{%
```

First, I^x is calculated with Equation (4).

```
        \advance\gth@tempcnta\m@ne
          \gth@max{%
            \gth@nameuse{\gth@sn @m@\the\gth@tempcnta
                         @\the\gth@tempcntb},%
            \gth@nameuse{\gth@sn @iy@\the\gth@tempcnta
                         @\the\gth@tempcntb}}%
          \gth@tempcntc\gth@calc
          \gth@tempcntd
            \gth@nameuse{\gth@sn @ix@\the\gth@tempcnta
                         @\the\gth@tempcntb}%
          \advance\gth@tempcntc-\gth@d\advance\gth@tempcntd-\gth@e
        \advance\gth@tempcnta\@ne
        \gth@max{\gth@tempcntc,\gth@tempcntd}%
        \gth@name@xdef{\gth@sn @ix@\the\gth@tempcnta
                       @\the\gth@tempcntb}{%
```

```
139           \the\gth@calc}%
```

Secondly, I^y is calculated with Equation (5).

```
140         \advance\gth@tempcntb\m@ne
141           \gth@tempcntc
142             \gth@nameuse{\gth@sn @m@\the\gth@tempcnta
143                          @\the\gth@tempcntb}%
144           \gth@tempcntd
145             \gth@nameuse{\gth@sn @iy@\the\gth@tempcnta
146                          @\the\gth@tempcntb}%
147           \advance\gth@tempcntc-\gth@d\advance\gth@tempcntd-\gth@e
148         \advance\gth@tempcntb\@ne
149         \gth@max{\gth@tempcntc,\gth@tempcntd}%
150         \gth@name@xdef{\gth@sn @iy@\the\gth@tempcnta
151                        @\the\gth@tempcntb}{%
152           \the\gth@calc}%
```

Finally, M is calculated with Equation (3) and a loop is completed.

```
153         \advance\gth@tempcnta\m@ne\advance\gth@tempcntb\m@ne
154           \gth@max{%
155             \gth@nameuse{\gth@sn @m@\the\gth@tempcnta
156                          @\the\gth@tempcntb},%
157             \gth@nameuse{\gth@sn @ix@\the\gth@tempcnta
158                          @\the\gth@tempcntb},%
159             \gth@nameuse{\gth@sn @iy@\the\gth@tempcnta
160                          @\the\gth@tempcntb}}%
161           \edef\gth@tmpa{\gth@nameuse{seqa@\the\gth@tempcnta}}%
162           \edef\gth@tmpb{\gth@nameuse{seqb@\the\gth@tempcntb}}%
163         \advance\gth@tempcnta\@ne\advance\gth@tempcntb\@ne
164         \ifx\gth@tmpa\gth@tmpb
165           \advance\gth@calc\gth@match
166         \else
167           \advance\gth@calc\gth@mismatch
168         \fi
169         \gth@name@xdef{\gth@sn @m@\the\gth@tempcnta
170                        @\the\gth@tempcntb}{%
171           \the\gth@calc}%
172         \advance\gth@tempcntb\@ne}%
173       \advance\gth@tempcnta\@ne}%
174   \fi
```

5.8.5 Printing Matrices

This is piece of code for debugging the package (so usually commented out).

```
175   %\gth@print@matrix{m}%
176   %\gth@print@matrix{ix}%
177   %\gth@print@matrix{iy}%
```

5.8.6 Returning the Optimal Score

The calculated optimal score of the alignment stored here to the control sequences which stored in `\gth@score`.

```
\bgroup
  \gth@decrement\gth@m \gth@decrement\gth@n
  \expandafter\xdef\gth@score{%
    \gth@nameuse{\gth@sn @m@\gth@m @\gth@n}}%
\egroup
```

5.8.7 Trace Back

After processing dynamic programming (or memoization functions), matrices M, I^x, and I^y are all filled. Using these matrices, we can determine the 'trace' from the optimal score $M_{m,n}$ to the start $M_{0,0}$ and get the result of alignment (x, y).

Considering the form of Equation (3)–(5), the calculation formulae of the former values depend on which matrix the 'current position' exists. In order to take into account these differences, trace back process is calculated while switching three modes: default, gap x, and gap y.

First, make sure to empty `\gth@rseq@x` and `\gth@rseq@y` which is going to store result alignment sequences x and y respectively. Note that this is the trace *back* process, so new characters are prepended to the existing sequences in each loop. In the next line, the switches `\if@gth@gapx@` and `\if@gth@gapy@` are both turned off (which means processing starts from the default mode).

```
\let\gth@rseq@x\@empty\let\gth@rseq@y\@empty
\@gth@remain@true\@gth@gapx@false\@gth@gapy@false
```

In the main loop of trace back, process the condition either x or y is already completed first. In this situation, the other sequence should be filled all the remaining with *gaps*.

```
\@whilesw\if@gth@remain@\fi{%
  \ifnum\gth@m=\z@
    \gth@decrement\gth@n
    \expandafter\expandafter\expandafter
    \ifx\gth@nameuse{seqb@\gth@n}\relax\else
      \edef\gth@rseq@x{\gth@gap@char\gth@rseq@x}%
      \edef\gth@rseq@y{\gth@nameuse{seqb@\gth@n}%
                       \gth@rseq@y}%
    \fi
  \else\ifnum\gth@n=\z@
    \gth@decrement\gth@m
    \expandafter\expandafter\expandafter
    \ifx\gth@nameuse{seqa@\gth@m}\relax\else
      \edef\gth@rseq@x{\gth@nameuse{seqa@\gth@m}%
                       \gth@rseq@x}%
      \edef\gth@rseq@y{\gth@gap@char\gth@rseq@y}%
    \fi
  \else
```

mode: gap x Prepend former base to x and a *gap* to y. If $M_{m-1,n} - d > I^x_{m-1,n} - e$ back to default mode. Else if $I^x_{m-1,n} - e < I^y_{m-1,n} - d$ switch to mode gap y and otherwise stay in mode gap x.

```
\if@gth@gapx@
  \gth@decrement\gth@m
  \expandafter\expandafter\expandafter
  \ifx\gth@nameuse{seqa@\gth@m}\relax\else
    \edef\gth@rseq@x{\gth@nameuse{seqa@\gth@m}%
                    \gth@rseq@x}%
    \edef\gth@rseq@y{\gth@gap@char\gth@rseq@y}%
  \fi
  \gth@tempcnta\gth@nameuse{\gth@sn @m@\gth@m @\gth@n}%
  \gth@tempcntb\gth@nameuse{\gth@sn @ix@\gth@m @\gth@n}%
  \gth@tempcntc\gth@nameuse{\gth@sn @iy@\gth@m @\gth@n}%
  \advance\gth@tempcnta-\gth@d
  \advance\gth@tempcntb-\gth@e
  \advance\gth@tempcntc-\gth@d
  \ifnum\gth@tempcnta>\gth@tempcntb
    \@gth@gapx@false
  \else\ifnum\gth@tempcntb<\gth@tempcntc
    \@gth@gapx@false\@gth@gapy@true
  \fi\fi
```

mode: gap y Prepend a *gap* to x and former base to y. If $M_{m,n-1} - d > I^y_{m,n-1} - e$ back to default mode and otherwise stay in mode gap y.

```
\else\if@gth@gapy@
  \gth@decrement\gth@n
  \expandafter\expandafter\expandafter
  \ifx\gth@nameuse{seqb@\gth@n}\relax\else
    \edef\gth@rseq@x{\gth@gap@char\gth@rseq@x}%
    \edef\gth@rseq@y{\gth@nameuse{seqb@\gth@n}%
                    \gth@rseq@y}%
  \fi
  \gth@tempcnta\gth@nameuse{\gth@sn @m@\gth@m @\gth@n}%
  \gth@tempcntb\gth@nameuse{\gth@sn @iy@\gth@m @\gth@n}%
  \advance\gth@tempcnta-\gth@d\advance\gth@tempcntb-\gth@e
  \ifnum\gth@tempcnta>\gth@tempcntb
    \@gth@gapy@false
  \fi
```

mode: default Prepend former base to x and y respectively. Only if $M_{m-1,n-1} > I^x_{m-1,n-1}$ and $M_{m-1,n-1} > I^y_{m-1,n-1}$ stay on default mode, and else if $I^x_{m-1,n-1} > I^y_{n-1,m-1}$ go to gap x otherwise go to gap y.

```
\else
  \gth@decrement\gth@m
  \gth@decrement\gth@n
  \expandafter\expandafter\expandafter
  \ifx\gth@nameuse{seqa@\gth@m}\relax\else
```

```
          \expandafter\expandafter\expandafter
          \ifx\gth@nameuse{seqb@\gth@n}\relax\else
            \edef\gth@rseq@x{\gth@nameuse{seqa@\gth@m}%
                            \gth@rseq@x}%
            \edef\gth@rseq@y{\gth@nameuse{seqb@\gth@n}%
                            \gth@rseq@y}%
          \fi
        \fi
        \gth@tempcnta\gth@nameuse{\gth@sn @m@\gth@m @\gth@n}%
        \gth@tempcntb\gth@nameuse{\gth@sn @ix@\gth@m @\gth@n}%
        \gth@tempcntc\gth@nameuse{\gth@sn @iy@\gth@m @\gth@n}%
        \@gth@gap@false
        \ifnum\gth@tempcnta<\gth@tempcntb\@gth@gap@true\fi
        \ifnum\gth@tempcnta<\gth@tempcntc\@gth@gap@true\fi
        \if@gth@gap@
          \ifnum\gth@tempcntb>\gth@tempcntc
            \@gth@gapx@true
          \else
            \@gth@gapy@true
          \fi
        \fi
      \fi\fi
    \fi\fi
```

Finally, if we achieve to $M_{0,0}$, exit from the main loop.

```
    \ifnum\gth@m<\@ne\ifnum\gth@n<\@ne
      \@gth@remain@false
    \fi\fi}%
```

5.8.8 Returning Results

Finally, the results of alignments which stored in `\gth@resq@x` and `\gth@resq@y` are copied globally to the control sequences which stored in `\gth@resulta` and `\gth@resultb`.

```
  \expandafter\gth@glet\gth@resulta\gth@rseq@x
  \expandafter\gth@glet\gth@resultb\gth@rseq@y}
```

5.9 Memoization Functions

`\gth@memo@ix` `\gth@memo@iy` `\gth@memo@m`

These are memoization functions for calculating I^x, I^y, M respectively. Since these functions change many 'temporary' values, all of the processes are wrapped in a group and only the return values and components of matrices are assigned globally.

```
\def\gth@memo@ix#1#2{%
  \bgroup
    \gth@tempcnta#1\gth@tempcntb#2\relax
    \expandafter\expandafter\expandafter
    \ifx\gth@nameuse{\gth@sn @ix@\the\gth@tempcnta
                     @\the\gth@tempcntb}\relax
      \advance\gth@tempcnta\m@ne
```

```
        \gth@memo@ix{\gth@tempcnta}{\gth@tempcntb}%
        \edef\gth@tmp@ix@return{\gth@ix@return}%
        \gth@memo@iy{\gth@tempcnta}{\gth@tempcntb}%
        \edef\gth@tmp@iy@return{\gth@iy@return}%
        \gth@memo@m{\gth@tempcnta}{\gth@tempcntb}%
      \advance\gth@tempcnta\@ne
      \gth@advance{\gth@m@return}{-\gth@d}%
      \gth@advance{\gth@tmp@ix@return}{-\gth@e}%
      \gth@advance{\gth@tmp@iy@return}{-\gth@d}%
      \gth@max{\gth@m@return,\gth@tmp@ix@return,\gth@tmp@iy@return}%
      \gth@name@xdef{\gth@sn @ix@\the\gth@tempcnta
                     @\the\gth@tempcntb}{%
        \the\gth@calc}%
      \xdef\gth@ix@return{\the\gth@calc}
    \else
      \xdef\gth@ix@return{%
        \gth@nameuse{\gth@sn @ix@\the\gth@tempcnta @\the\gth@tempcntb}}
    \fi
  \egroup}
\def\gth@memo@iy#1#2{%
  \bgroup
    \gth@tempcnta#1\gth@tempcntb#2\relax
    \expandafter\expandafter\expandafter
    \ifx\gth@nameuse{\gth@sn @iy@\the\gth@tempcnta
                     @\the\gth@tempcntb}\relax
      \advance\gth@tempcntb\m@ne
        \gth@memo@iy{\gth@tempcnta}{\gth@tempcntb}%
        \edef\gth@tmp@iy@return{\gth@iy@return}%
        \gth@memo@m{\gth@tempcnta}{\gth@tempcntb}%
      \advance\gth@tempcntb\@ne
      \gth@advance{\gth@m@return}{-\gth@d}%
      \gth@advance{\gth@tmp@iy@return}{-\gth@e}%
      \gth@max{\gth@m@return,\gth@tmp@iy@return}%
      \gth@name@xdef{\gth@sn @iy@\the\gth@tempcnta
                     @\the\gth@tempcntb}{%
        \the\gth@calc}%
      \xdef\gth@iy@return{\the\gth@calc}%
    \else
      \xdef\gth@iy@return{%
        \gth@nameuse{\gth@sn @iy@\the\gth@tempcnta @\the\gth@tempcntb}}
    \fi
  \egroup}
\def\gth@memo@m#1#2{%
  \bgroup
    \gth@tempcnta#1\gth@tempcntb#2\relax
    \expandafter\expandafter\expandafter
    \ifx\gth@nameuse{\gth@sn @m@\the\gth@tempcnta
                     @\the\gth@tempcntb}\relax
      \advance\gth@tempcnta\m@ne\advance\gth@tempcntb\m@ne
        \gth@memo@ix{\gth@tempcnta}{\gth@tempcntb}%
```

```
        \edef\gth@tmp@ix@return{\gth@ix@return}%
        \gth@memo@iy{\gth@tempcnta}{\gth@tempcntb}%
        \edef\gth@tmp@iy@return{\gth@iy@return}%
        \gth@memo@m{\gth@tempcnta}{\gth@tempcntb}%
        \edef\gth@tmpa{\gth@nameuse{seqa@\the\gth@tempcnta}}%
        \edef\gth@tmpb{\gth@nameuse{seqb@\the\gth@tempcntb}}%
      \advance\gth@tempcnta\@ne\advance\gth@tempcntb\@ne
      \gth@max{\gth@m@return,\gth@tmp@ix@return,\gth@tmp@iy@return}%
      \ifx\gth@tmpa\gth@tmpb
        \advance\gth@calc\gth@match
      \else
        \advance\gth@calc\gth@mismatch
      \fi
      \gth@name@xdef{\gth@sn @m@\the\gth@tempcnta
                     @\the\gth@tempcntb}{%
        \the\gth@calc}%
      \xdef\gth@m@return{\the\gth@calc}%
    \else
      \xdef\gth@m@return{%
        \gth@nameuse{\gth@sn @m@\the\gth@tempcnta @\the\gth@tempcntb}}%
    \fi
  \egroup}
```

5.10 Configuration Command

`\GotohConfig` This is just a wrap command of `\setkeys`.

```
\newcommand{\GotohConfig}[1]{\setkeys[gth]{config}{#1}}
```

References

[1] Flouri, Tomáš *et al.*, "Are all global alignment algorithms and implementations correct?". *bioRxiv*, 031500, 2015.

[2] Gotoh, Osamu, "An improved algorithm for matching biological sequences". *Journal of Molecular Biology* **162**(3), 705-708, 1982.

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

D

G

Change History