HITsにおけるTyping練習用採点プログラムの改良2020

松　井　吉　光

要　　旨：情報リテラシー・入門で利用しているe-LearningシステムであるHITsにおいては，Typing練習の問題を多数用意し，自習できる環境を用意している。そのHITsにおけるTyping採点プログラムについて，2020年度の講義に向けて，エラーメッセージ表示機能の根幹をなす差分検出アルゴリズムを、より高速に動作するアルゴリズムへの変更を行った。本稿では、変更に用いた差分検出アルゴリズムについて述べる。

キーワード：e-Learning，Typing，情報リテラシー教育，差分検出アルゴリズム

1．はじめに

近年，スマートフォンの普及によりネットを使ったコミュニケーション，情報収集に一定のスキルを持って大学に入学してくる学生が増えてはいるが，逆にPCの利用に関して言えば，その機会が減ってきていて扱いに不慣れな学生が増えているように感じられる。それを反映してか，キーボードの入力に関して，アルファベットの大文字の入力の仕方といったごく基本的な操作方法についての質問がなされるようになってきている。スマートフォンのフリック入力・予測変換による入力には長けている学生でも，キーボードによる文字入力についても不慣れで入力速度がほとんど上がらない学生が多いようである。

PCにおける入力の主体が今後音声入力等の他の方法に移行していくことは考えられるが，今後何年かはキーボード入力がその地位を譲ることは想定できる現状ではないので，キーボード入力の技術，特にタッチタイピングのスキルは学生にとって必須のものであり続けるであろう。そのため，情報リテラシー・入門ではe-LearningシステムであるHITsにおいて，Typing練習問題を多数用意し，練習できる環境を提供してきた。

そのTyping練習を支えているのが，自動採点プログラムである。採点において，入力の各行の文字列を模範と比較し，一致しているかどうかで正解・不正解を判定するという方法をとっており，その点において全く支障は発生していない。ただ，不正解と判定した場合に，不正解箇所の学生への提示において，最初の間違い箇所を表示するのみで，全体で何カ所間違っているか，２つ目以降の間違っている箇所がどこかを表示するようにはなっていないという，学習者にとって不親切な仕様になっていた。今回は，前年に行った改良[16] 即ち、不正解箇所のメッセージをより具体的に，文書全体でどの部分に，誤字があるか，書込不足か，書込過剰かといった間違いを一目で分かるようなメッセージを作成する

のに必要だった文字列差分検出アルゴリズムを，より高速に動作するアルゴリズムへの変更を行った。

２．差分検出アルゴリズムの基本

このプログラム改良に用いた文字列差分検出アルゴリズムとは，入力された文字列と解答の文字列との差分を検出するためのものである。UNIX等ではファイル等の差分を検出するdiffというプログラムがあり，PHPにも差分を検出するライブラリーが存在するが，今回も，自前で差分検出プログラムをコーディングすることを試みた。

二つの文字列の差分を求めることとは，変更によって一方の文字列からもう一方の文字列を生成するときに，どの文字が追加されたか，どの文字が削除されたか，どの文字列がそのままなのかを調べることに尽きる。その際，コンピュータで求めるべきは

- SES (Shortest Edit Script)
- 編集距離(Edit Distance)
- 最長共通部分列
  (Longest Common Subsequence)

の三つに還元される。その三つの量について以下に述べる。

SES

二つの文字列A，Bがあったとき，文字列Aから文字列Bへ変更するときの手順は，様々なものが考えられる。文字列に対する編集操作としては

- 任意の位置への一文字の挿入
- 任意の一文字の削除

の二つのみが許されるものとする。手順のうち最大の手数は，文字列Ａの全ての文字を削除し，文字列Ｂのすべての文字を挿入する手順であろう。文字列Ａと文字列Ｂに共通の文字がなければもちろん，この手順しかないが，そうでなければ，もっと少ない手数で変更が可能になるはずである。そういった様々な手順のうち，最も手数が少ない手順をSESと呼ぶ。

例えば，二つ「abcde」と「abief」であれば，

1.「a」をそのまま
2.「b」をそのまま
3.「c」を削除
4.「i」を「b」の後に挿入
5.「d」を削除
6.「e」をそのまま
7.「f」を「e」の後に挿入

がSESとなる。ただし，SESは二つの文字列が与えられとしても，一意に決まらない場合がある。例えば，
「abcde」と「abdce」の場合だと，

1.「a」をそのまま
2.「b」をそのまま
3.「c」を削除
4.「d」をそのまま
5.「c」を「d」の後に挿入
6.「e」をそのまま

と

1.「a」をそのまま
2.「b」をそのまま
3.「c」をそのまま
4.「d」を削除
5.「d」を「b」の後に挿入
6.「e」をそのまま

はともにSESとなる。

編集距離

二つの文字列A，Ｂの違いを数値化したもので，文字列Ａから文字列Ｂへ変更する際の編集操作の最小回数を指す。文字列の編集操作はSESと同様，

- 任意の位置への一文字の挿入
- 任意の一文字の削除

の二つのみが許されるものとし，それぞれを「1」とする。編集距離はSESが分かれば求めることができて，SESにおける削除・挿入の

回数に相当する。先の例「abcde」と「abief」の場合であれば，編集距離は「4」となり，「abcde」と「abdce」の場合であれば，編集距離は「2」となる。編集距離は二つの文字列が与えられれば，一意に決まると考えて良い。

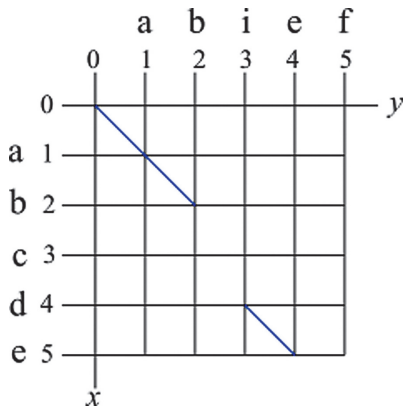
最長共通部分列(LCS)

最長共通部分列とは，二つの文字列に共通して含まれる最も長い部分列のことである。この時，部分列は部分文字列ではなく，連続した文字列である必要はない。ただし，出現する順序は元の文字列の順序通りである必要がある。

先の例「abcde」と「abief」の場合であれば，最長共通部分列は「abe」となる。最長共通部分列はSESと同様，一意に決まらない場合がある。先の例「abcde」と「abdce」だと，最長共通部分列は「abce」と「abde」のどちらとしても良い。最長共通部分列とSESは一対一の関係になっており，SESが決まれば最長共通部分列が決まり，最長共通部分列が決まればSESが決まるという関係にある。

差分検出アルゴリズムでは，上記三つの量，SES，編集距離，最長共通部分列を求めることになるが，この問題を編集グラフ（Edit Graph）上の最短距離取得問題に還元し，解を求めるのが一般的である。編集グラフとは，文字列Ａと文字列Ｂの各要素をＸ軸Ｙ軸上に並べ，それの交点を縦横の辺で結合し，文字列Ａのx番目の文字と文字列Ｂのy番目の文字が等しい場合のみ，$(x-1, y-1)$ から (x, y) へ斜め線で結んだものである。

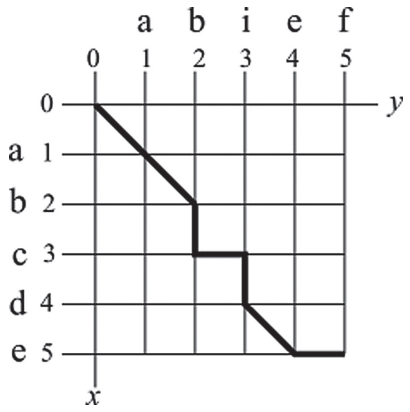
例えば，「abcde」と「abief」の場合であれば，編集グラフは以下のようになる。



（図1　編集グラフの例）

このような編集グラフを考えるとき，SESを求める問題は，原点（0, 0）から座標（M, N）まで向かう場合の，最短距離を求める問題に還元される（ここで，M, Nはそれぞれ文字列A，文字列Ｂの文字数である）。このとき考える距離は単純な距離ではなく，編集グラフにおいて縦横へ一つ分の移動を距離「1」，斜め線での移動を距離「0」とする。この時の最短距離が，編集距離に相当する。また，編集グラフをSESに解釈するときは，斜め線での移動を「そのまま」とし，下方向への移動を「文字の削除」，右方向への移動を「文字の挿入」とすれば良い。

上記の例でSESは例えば，以下のような太線の経路である。



（図2　編集グラフ上でのSESの例）

太線において，縦の線が二つ，横の線が二つ

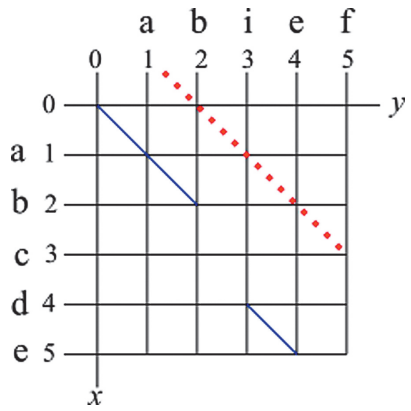
あるので，距離は「4」となり，編集距離が「4」と合致しているのが分かる。

編集グラフ上の最短距離取得問題を解くアルゴリズムとしては，単純な「動的計画法」を用いる方法や，それを文字列の差分を求めるために効率を良くした，O(ND)アルゴリズム[8]やO(NP)アルゴリズム[9]がある。今回の改良ではより高速にSESを求めることができるO(ND)アルゴリズムを採用した。

3．O(ND)アルゴリズム

O(ND)アルゴリズムはMyersによって開発されたアルゴリズムである。このアルゴリズムは「動的計画法」の方法のように各頂点に辿り着くまでの最長共通部分列の長さ，即ち編集距離を求めるのではなく，与えられた編集距離でどこまで辿り着けるかを考える。

まず各頂点(x, y)を通る傾き1の斜めの直線を$k = y - x$とする。このとき値kはy切片の値である。例えば，以下の図における破線は$k = 2$で指定される直線を表す。



（図3　点線は$k = 2$で指定される直線）

直線k上で，編集距離Dで辿り着ける最大のxの値を$V_D(k)$とする。

まずは，編集距離$D = 0$の場合$V_D(k)$を考える。値が存在するのは$V_0(0)$だけで，原点(0, 0)から右下に斜めの辺を辿るだけで求めることができる。このように「直線k上で(x, y)から斜めの辺を行き止まるまで辿った時の最終的なxの値」をsnake(k, x)と置くことにする。これを用いると

$$V_0(0) = \mathrm{snake}(0, 0)$$

と書ける。

次に編集距離$D = 1$の場合即ち$V_1(k)$を考えると，これは二つの場合だけが存在して

- $k = -1$：$V_0(0)$から下に移動して
  $$V_1(-1) = \mathrm{snake}(-1, V_0(0)+1)$$
- $k = 1$：$V_0(0)$：から右に移動して
  $$V_1(1) = \mathrm{snake}(1, V_0(0)+1)$$

のように求めることができる。

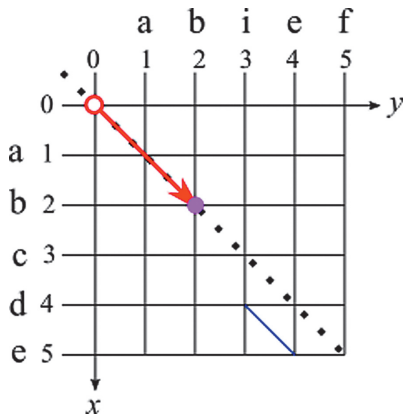
同様に，$V_{D-1}(k)$が全て求められている時，$V_D(k)$は各kに対して，

$$V_D(k) = \mathrm{snake}(k, \max(V_{D-1}(k+1)+1, V_{D-1}(k-1)))$$

で求められる。下端即ち$k = -D$または，右端即ち$k = D$の場合で$V_{D-1}(k+1)$か$V_{D-1}(k-1)$のどちらかしか存在しない場合は，存在する側のみを考える。

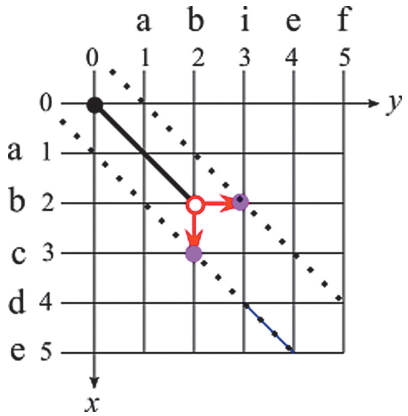
このようにして，編集距離Dを増やしながら$V_D(k)$の値を計算していき，$k = N - M$上で$x = M$に辿り着いたら，その時のDの値が二つの文字列の編集距離になっている。

上記の例の場合，編集距離$D = 0$だと，$V_0(0) = 2$となる。



（図4　点線は$k = 0$で指定される直線。赤線は$D = 1$で移動できる経路）

次に編集距離$D = 1$だと

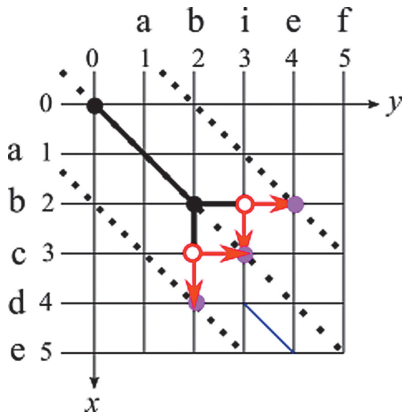


（図5　点線は$k = \pm 1$で指定される直線。
赤線は$D = 0$で移動できる経路）

となり，

- $k = -1$：$V_1(-1) = 3$
- $k = 1$：$V_1(1) = 2$

となる。以下，編集距離$D = 2$の場合は，

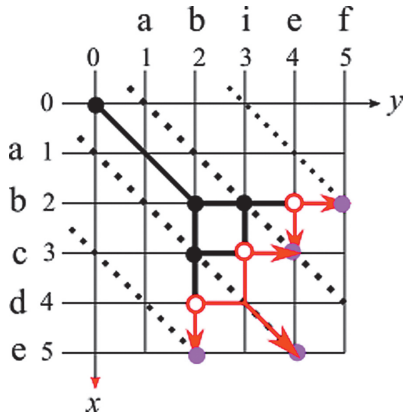


（図6　点線は$k = 0, \pm 2$で指定される直線。
赤線は$D = 2$で移動できる経路）

となるから，

- $k = -2$：$V_2(-2) = 4$
- $k = 0$：$V_2(0) = 3$
- $k = 2$：$V_2(2) = 2$

となる。次に，編集距離$D = 3$の場合は，

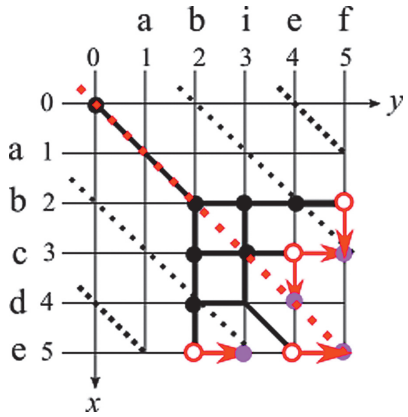


（図7　点線は$k = \pm 1, \pm 3$で指定される直線。
赤線は$D = 3$で移動できる経路）

となるから，

- $k = -3$：$V_3(-3) = 5$
- $k = -1$：$V_3(-1) = 5$
- $k = 1$：$V_3(1) = 3$
- $k = 3$：$V_3(3) = 2$

となる，次に，編集距離$D = 4$の場合は，



（図8　点線は$k = 0, \pm 2, \pm 4$で指定される直線。
赤線は$D = 3$で移動できる経路）

となるから，

- $k = -2$：$V_4(-2) = 5$
- $k = 0$：$V_4(0) = 5$
- $k = 2$：$V_4(2) = 4$

となる。この例の終了は$k = N - M = 5 - 5 = 0$上になるところに到達することだから，

$V_4(0)=5$となったので経路の検索は終了となる。そして，これで二つの文字列「abcde」を「abief」の編集距離が「4」であることが求まる。あとはこの検索過程でどのような経路を辿って$V_4(0)=5$になったのかを記録しておけば，SESを求めることができ，最長共通部分列を求めることもできる。

以上が，O(ND)アルゴリズムを用いた，差分検出用の最短経路探索アルゴリズムであり，このアルゴリズムを，PHP言語を使って実装した。

4．まとめ

今回は文字列の差分，即ち編集距離，エディットグラフを求めるアルゴリズムとしてMyersによって提案されたO(ND)アルゴリズム[8]を採用した。このアルゴリズムだと，計算量は$O((M+N)\times D)$に押さえることができる。これはエディットグラフ上の最短経路を導く手順で，すべての格子点を辿ることをせずに，検索範囲を徐々に広げていく方法をとっているためである。

ただ，HITsのTypingの問題では各行が高々30文字であるため，あまり「動的計画法」を用いたアルゴリズムを用いた場合と処理速度上差異を実感する可能性は低いと思われる。

参考文献

[1] 岩田員典，功刀由紀子，齋藤毅，谷口正明，長谷部勝也，松井吉光，古川邦之，「Excel, Word自動採点システムHITsの構築と運用」，愛知大学情報メディアセンター紀要 COM vol.20, No.1, 2010

[2] 岩田員典，松井吉光，長谷部勝也，谷口正明，池森均，梅垣敦紀，齋藤毅，澤田貴行，土橋喜，中尾浩，西本寛，古川 邦之，毛利元昭，「情報リテラシーのためのWord，Excel自動採点システムの構築と運用」，教育改革ICT戦略大会 pp.294-295（2013）

[3] 松井 吉光，谷口 正明，長谷部 勝也，「HITsにおけるWord文書の採点プログラムの開発」，愛知大学一般教育論集，(40)，25-40（2011）

[4] 長谷部 勝也，松井 吉光，谷口 正明，「HITsにおけるWord文書の採点プログラム2013年度版の開発」，愛知大学一般教育論集，(45)，41-53（2013）

[5] 松井吉光，長谷部勝也，谷口正明「HITsにおけるWord文書の採点プログラム2016年度版の開発」愛知大学一般教育論集，(52)27-36(2017)

[6] 松井吉光，谷口正明，「HITs におけるWord文書の採点プログラム2018年度版の開発」愛知大学一般教育論集，(54)43-49(2018)

[7] 長谷部勝也，松井吉光，谷口正明，「HITsにおけるWord文書の採点プログラム2019年度版の開発」愛知大学一般教育論集，(56)31-39(2019)

[8] E.W.Myers, “An O(ND) Difference Algorithm and Its Variations” Algorithmica 1, 251 (1986)

[9] Sun Wu, Udi Manber, G.Myers, W.Miller, “An O(NP) Sequence Comparison Algorithm” Information Processing Letters 35, 317 (1990)

[10]『アルゴリズムイントロダクション「第２巻」アルゴリズムの設計と解析手法 第３版』T.コルメン，C.ライザーソン，R.リベスト，C.シュタイン（浅野哲夫，岩野和生，梅尾博司，山下雅史，和田幸一共訳）近代科学社

[11] 久保達彦，「diffの動作原理を知る～どのようにして差分を導き出すのか」https://gihyo.jp/dev/column/01/prog/2011/diff_sd200906（2019/8/23参照）

[12]「差分検出アルゴリズム三種盛りプログラミング」https://susisu.hatenablog.com/entry/2017/10/09/134032（2019/8/23参照）

[13] N. Tsuda,「diff ってなんだ」https://gist.github.com/gurimusan/7e554eb12f9f59880053（2019/8/23参照）

[14] 文書比較（diff）アルゴリズム http://hp.vector.co.jp/authors/VA007799/viviProg/doc5.htm（2019/8/23参照）

[15] 典型的な DP（動的計画法）のパターンを整理 Part 1 ～ ナップサック DP 編 ～ https://qiita.com/drken/items/a5e6fe22863b7992efdb(2019/8/26参照)

[16] 松井吉光,「HITsにおけるTyping練習用採点プログラムの改良」愛知大学情報メディアセンター紀要 COM vol.30,15-22 No.1, 2020