HITsにおけるTyping練習用採点プログラムの改良

松井　吉光（愛知大学法学部）

要旨：

近年，情報機器の普及が進む反面，PCの扱い，特にキーボード入力に不慣れな学生が再び増加しつつある。今後しばらくはPCにおける入力の主体はキーボード入力であり続ける可能性が高く，タッチタイピングによるキーボード入力スキルを身につけることは，学生にとって大学での勉学を修める上でも，卒業後社会に出てからも必須であると言えよう。このような状況にあるため，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練習の問題を多数用意し，自習できる環境を整備している。そのHITsにおけるTyping採点プログラムについて，2019年度の講義に向けて，差分検出アルゴリズムを用いてエラーメッセージ表示機能の改良を行った。それにより，学生のTyping 学習の効率化を図った。本稿では改良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インター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将来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ため，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，不正解個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，不正解個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に改良を行った。このメッセージを作成するのにあたり用いたのが，本稿で紹介する差分検出アルゴリズムである。

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TextDiff等のライブラリーが存在するが，メンテナンスが終了していたり，ライセンスが明記されてなかったり等の問題があったため，今回は，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。そのうち，SESを考えるときは，文字列に対する編集操作として

- 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。

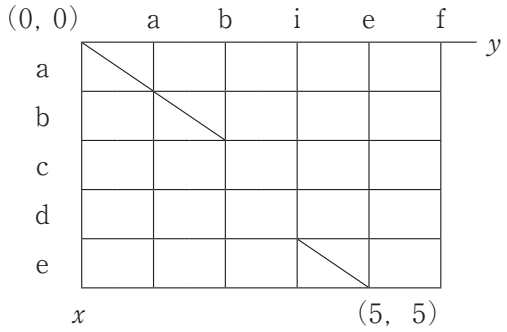
最長共通部分列（LCS）

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i番目の文字と文字列Bのj番目の文字が等しい場合のみ，$(i-1, j-1)$から(i, j)へ斜め線で結んだものである。

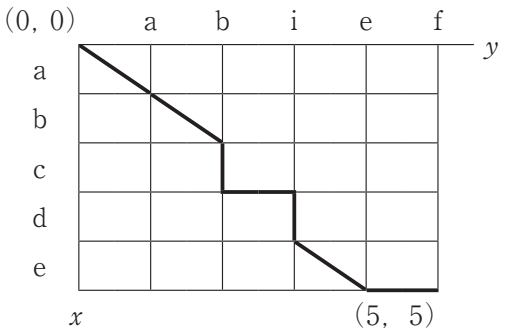
例えば，「abcde」と「abief」の場合であれば，編集グラフは以下のようになる。



（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。今回の改良では最も単純な「動的計画法」を用いるアルゴリズムを採用した。

3.「動的計画法」アルゴリズム

単純な「動的計画法」を用いるアルゴリズムでは，まず編集グラフの各頂点に辿り着くまでの最長共通部分列の長さを計算する。文字列A（文字数M）と文字列B（文字数N）を考えるときは，（$M+1$）×（$N+1$）の表を作成し，先頭の行，列を以下のように「0」で初期化する（ここでは，文字列Aを「abcde」と文字列Bを「abief」を例にとる）。

		a	b	i	e	f
	0	0	0	0	0	0
a	0					
b	0					
c	0					
d	0					
e	0					

（表1）

この表の算出する規則は以下の通りとする。

- 文字列Aのi番目の文字と文字列Bのj番目の文字の，それぞれ一つ前の文字（一つ左，または一つ上）の最長共通部分列の長さを基準として考える。
- 一つ左と一つ上の二つの最長共通部分列の長さのうちどちらか一方が長い場合は，長い方がその格子点における最長共通部分列の長さとなる。
- 文字列Aのi番目の文字と文字列Bのj番目の文字が等しい場合は，最長共通部分列の長さが一つ伸びるので「1」を加算する。

この規則を用いると先の例の場合は，各格子点における最長共通部分列の長さは以下のようになる。

		a	b	i	e	f
	0	0	0	0	0	0
a	0	1	1	1	1	1
b	0	1	2	2	2	2
c	0	1	2	2	2	2
d	0	1	2	2	2	2
e	0	1	2	2	3	3

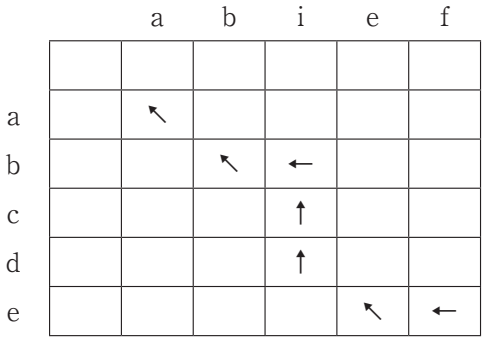
（表2）

次に，この表を用いてSESと最長共通部分列を求めることになるが，このとき座標（M，N）から原点（0，0）に戻ることを考え，その時最短となる経路を以下の手順で決定していく。

- 文字列Aのi番目の文字と文字列Bのj番目の文字が等しい場合は，編集グラフで（i，j）から（$i-1$，$j-1$）に進む。（編集グラフ上では左斜め上に進む）
- 文字列Aのi番目の文字と文字列Bのj番目の文字が等しくない場合は，格子点（$i-1$，j）と（i，$j-1$）に最長共通部分列の長さを比較し，長い方に進む（編集グラフ上では左ま

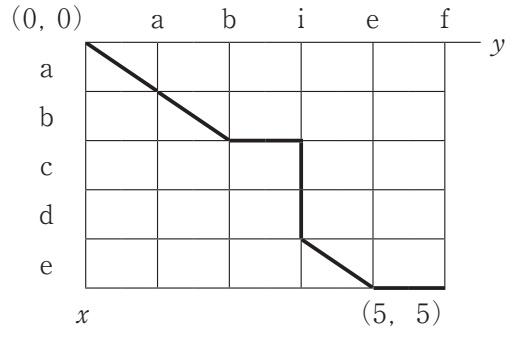
たは，上に進む)。なお，同じ場合はどちらに進んでも良いので，実装上は上に進むことにした。

今回の例でこの手順を実行すると，以下のような図になる。



（図3）

この経路の中で斜め上に進んだ箇所の文字列を左上から経路を順に辿って集めると「abe」となり，最長共通部分列になっていることが分かる。この場合の経路は，編集グラフでは以下のようになる。



（図4）

これを解釈すると，

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

となり，SESになっていることが分かる。また，このSESから編集距離は「4」であることも導き出すことができる。

以上が，単純な「動的計画法」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言語を使って実装を行った。

4. まとめ

今回はHITsのTyping採点プログラムの改良を行い，不正解個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にした。その際，文字列の差分を検出するアルゴリズムとして，前節で紹介した単純な「動的計画法」を用いるアルゴリズムを採用した。

単純な「動的計画法」を用いるアルゴリズムは実行時間が最長・平均ともにO(MN)となり，文字列が長くなると計算量，メモリの使用量とも膨大になってしまうという欠陥が存在する。それは，最短経路の探索で，編集グラフ上の全ての

格子点をいったん全て辿って最長共通部分列の長さを計算し，それから経路を見つけ出すという手順になっているためである。それは共通部分の多い文字列同士での比較ではとても冗長な作業となってしまうからである。それを改善する方法としては，最短経路の探索範囲を絞って計算量とメモリの使用量を減らす方法があり，それが O（ND）アルゴリズム［8］や O（NP）アルゴリズム［9］等である。

ただ，HITs の Typing の問題では各行が高々30文字であるため，その欠陥が大きな問題になることはほぼ想定する必要はないと判断している。しかし，今後としては，他のより効率の良いアルゴリズムを採用することも視野にいれて，改良をして行いたいと考えている。

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）