

1. 序論

今使われているキーボード配列 (QWERTY) は、タイプライターの制限の中で作られたものであり、現代の入力環境に最適とは言えない。人が入力する内容は、言語や分野によって大きく異なるが、現在の配列はどんな文書にも一律に使われている。

ある人が普段入力する文書にはある程度の傾向があると考えた。例えば、日本人は日本語を入力する頻度が高いだろう。また、小説家と数学者の間にも入力内容の傾向に差があるだろう。ならば、それぞれに合ったキー配列を作るのが良い。そこで私は、「与えられた文書の内容」に合わせて最も効率よく入力できるキー配列を自動的に作り出すプログラム (以下、太郎と呼ぶ。) を開発することにした。

今回のテーマは「文書に特化した最適化」だが、「日本語に特化した最適化」という観点で開発されたキー配列として、「大西配列」がある。大西配列は、文字の出現頻度や連続して現れる文字の組み合わせを分析し、「片手の連続使用・指の移動・指の連続使用」を減らすように設計されている。

今回開発したプログラムでは、大西配列が日本語に対して行った最適化を文書に対して行うことで、文書に特化した最適化を実現した。

今回実施した分析の再現性を担保するため、実装したプログラム一式を GitHub にて公開した。

URL: <https://github.com/yuichiro-kurose/keyboard>

2. 方法

はじめに、太郎を開発した。太郎は与えられた文書に対して、次の 3 つの段階を踏んで最適化を行うように設計した。

A: 片手の連続使用を最小化

B: 指の移動を最小化

C: 指の連続使用を最小化

B は大西氏が開発した「〇等地」によって定義された「押しにくさ」を最小化することで実現した。「押しにくさ」が最小となる配置は複数あるので、その中で最も指の連続使用が少ないものを調べることで C を実現した。

A に関して、大西配列では「母音を左手に集める」という手法が取られていたが、今回は最小化を目指して独自のアルゴリズムを開発することで実現した。

キーを右手と左手に分ける方法は、 ${}_{26}C_{13} = 10400600 \sim 10^7$ 通りある。ただし $a \sim b$ は a と b が同程度であることを表している。片手が連続した回数が最小となる分け方を見つけるためには、「それぞれの分け方における片手の連続使用回数」を数える必要がある。 N を文書の文字数とすると、これには $10^7 N$ 回の計算が必要である。

$N = 10^6$ とすると、一般的な家庭用のノートパソコンではプログラムが終了するまでに約 1 日の時間を要する。そこで、グラフ理論を応用し、片手の連続使用回数を事前に計算することで高速化した。

頂点集合を $\{a, b, \dots, z\}$ とする重み付き完全無向グラフ K_{26} を考える。ただし、辺の重み w を、

$w(u, v)$ = 文書内で uv または vu が現れた回数

と定める。例えば、文書が「Yuichiro Kurose」のとき、

$w(a, b) = 0, w(k, o) = 1, w(o, r) = 2$

となる。このとき、「片手の連続使用回数が最小となる分け方」は、「各グループ内における w の合計が最小となるように K_{26} を分割する方法」に対応する。

このアルゴリズムでは、 K_{26} の構築に $N + 26^2$ 回、各分割方法における重みの総和の計算に ${}_{26}C_{13} \cdot {}_{13}C_2 \sim 10^9$ 回の計算が必要であるから、全体で約 $N + 10^9$ 回の計算が行われる。高速化前後での計算回数を比較すると次の図のようになる。



縦軸: 計算回数 t 横軸: 文書の文字数 N

赤: 高速化前 ($t = 10^7 N$)

青: 高速化後 ($t = N + 10^9$)

高速化後のグラフは、高速化前のグラフと比較してほぼ横ばいであり、劇的な高速化が行われていることがわかる。このアルゴリズムでは、 $N = 10^6$ のときも現実的な時間内でプログラムが終了する。

次に、太郎が生成したキー配列 (以下、次郎と呼ぶ。) の性能を数値的・視覚的に評価した。評価は次の手順で行った。

(1) A Christmas Carol in Prose の序盤 (約 30000 文字) を用いて、次郎を生成した。

(2) The Strange Case of Dr. Jekyll and Mr. Hyde の序盤 (約 10000 文字) を用いて、次郎と QWERTY の「片手の連続使用率」と「指の連続使用率」をそれぞれ計算し、比較した。

(3) アルファベットの文字頻度でキーを色分けすることで、次郎と QWERTY の「指の移動」を視覚的に評価した。

(2) は人力では現実的な時間内に終わらないため、(2) を行うプログラムを開発することで実現した。

3. 調査の結果

(1) の結果

	U	O	C	Z		V	G	H	W	
A	I	E	T	Y		M	N	S	R	D
Q	J	X	K				F	L	B	P

(2) の結果

	QWERTY	次郎
片手の連続使用率	47.8%	34.2%
指の連続使用率	10.5%	6.4%

(3) の結果

QWERTY

Q	W	E	R	T		Y	U	I	O	P
A	S	D	F	G		H	J	K	L	_
Z	X	C	V	B		N	M	_	_	_

次郎

_	U	O	C	Z		V	G	H	W	_
A	I	E	T	Y		M	S	N	R	D
Q	J	X	K	_		_	F	L	B	P

文字頻度は、赤 > 黄 > 緑 > 黒

4. 考察

次郎において、母音（A, E, I, O, U）が左手に集中している。この結果は、大西氏の「母音を左手に集める」という考えの正当性を示していると考えた。

キー配列がランダムである場合、片手の連続使用率の理論値は 50% であり、QWERTY はそれにおおよそ従っている。次郎の片手の連続使用率は 50% を大きく下回るが、この数値的な差が人間にとっての「入力しやすさ」にどれほどの影響があるのかはわからない。

次郎は QWERTY と比較して、ホームポジションに文字頻度の高いキーが集まっており、指の移動にかかる負担が少ないと考えた。

次郎の生成と評価には、それぞれ異なる文書を用いたため、今回の結果は、「次郎は次郎の生成に用いた文書だけでなく、それと同類の文書に対して、数値的・視覚的に高い性能を発揮する。」ということを示していると考えた。

5. 結論・展望

アルファベットからなる文書に最適化されたキー配列の生成を行うプログラムを開発した。このプログラムによって生成されたキー配列は、最適化に用いた文書と同類の文書に対して数値的・視覚的に高い性能を発揮した。

しかし、実際に人間にとって入力しやすいかは、現時点では不明である。また、QWERTY から新しいキー配列への乗り換えが困難である可能性が高い。

今回はアルファベットのみを対象としたが、文書をアルファベットに対応させるプログラムを用いることで、アルファベット以外の文書も最適化できる可能性が高い。また、記号への対応はプログラミングの効率化が期待できる。

6. 参考文献

[1] 大西拓磨．“ローマ字入力に最適なキー配列を考える（制作編）”．<https://note.com/illlillililil/n/n3b51f4aaf086> , (参照 2026-02-22) .

[2] Charles Dickens . “A Christmas Carol in Prose; Being a Ghost Story of Christmas” .
<https://www.gutenberg.org/cache/epub/46/pg46.txt> , (参照 2026-02-22) .

[3] Robert Louis Stevenson . “The strange case of Dr. Jekyll and Mr. Hyde” .
<https://www.gutenberg.org/cache/epub/43/pg43.txt> , (参照 2026-02-22) .

[4] 堀田隆一．“#5745. アルファベットの文字頻度” .
<https://user.keio.ac.jp/~rhotta/hellog/2025-01-18-1.html> , (参照 2026-02-22) .

[5] 秋葉拓哉・岩田陽一・北川宜稔．プログラミングコンテストチャレンジブック．株式会社マイナビ出版，2022 .