組版におけるオペレーションズ・リサーチ

黒木　裕介

みなさんが何気なく目にしている印刷物も，最近ではほとんどがコンピュータ上でソフトウェアを用いて組版したものです．本稿では，コンピュータ上で「よい」組版を実現した TEX というソフトウェアについて解説した *The TEXbook* に基づいて，オペレーションズ・リサーチの一端を紹介します．どんな組版結果が望ましいかを定量的に捉え，高速に組版できるアルゴリズムをもって解決しようという発想は，オペレーションズ・リサーチの好例になっています．

キーワード：モデル化，モデリング，動的計画法，非巡回有向グラフ，最短路問題

1. はじめに

いきなりですが，ずっとこの調子で活字が組まれていたら，この文章を読む気になりますか？　高々数行であれば，何とか読んでもらえるかもしれません．しかし，これが延々と続くとしたら，私なら読むのをあきらめてしまいそうです．もしかしたら面白い内容なのかもしれないのに残念なことです．

この段落からは「通常通り」に戻りましょう．活字を組んで版（印刷用の板）を作ることを，組版（くみはん）と呼びます．元々「活字を組む」というのは，写真 1 のように金属の活字を並べて固定する作業のことを言いました．最近では，数十〜数百ページもある本に対して，元来の意味で活字を組むことはないでしょう．その代わり，コンピュータ上で組版をする DTP (desktop publishing) が一般的です．となれば，何らかのソフトウェアを用いて，「よい」組み方を実現しようとなります．

40 年近く前に，計算機科学者のクヌース博士がこの問題に取り組み，TEX（日本ではテフまたはテックと読みます）という組版ソフトウェアを開発しました．TEX は数式の組版が美しいことで有名ですが，本文の組版に関しても優れた機能を備えており，数式を多く含む学術分野を中心に商業印刷の世界でもいまなお使われています．筆者も，レポートをはじめとして多くの文書を，TEX を用いて出力しています．この原稿も，筆者の手元では TEX を用いて仕上がりを確認しながら執筆しています．

TEX の仕組みを解説したクヌース博士著 *The TEXbook* [2] の Chapter 14 において，TEX では本文一段落分をどのように組んでいるかが解説されています．その内容はオペレーションズ・リサーチの好例になっているので，その一端を紹介します．言語によって組版のルールは異なることから，話を単純にするために，原典に基づいて英文の組版の話に限ります．

くろき　ゆうすけ
kuroky@users.sourceforge.jp

写真 1　活字を組んでいる手元 [1]

2. 出力と入力の例

以下の三つの組版例を見てみましょう．

（イ）　Mr. Drofnats—or “R. J.,” as he preferred to be called—was happiest when he was at work typesetting beautiful documents.

（ロ）　Mr. Drofnats—or “R. J.,” as he preferred to be called—was happiest when he was at work typesetting beautiful documents.

（ハ）　Mr. Drofnats—or “R. J.,” as he preferred to be called—was happiest when he was at work typesetting beautiful documents.

それぞれに改行する位置が異なり，見た目が異なります．どれが最も「よい」でしょうか？

「よい」組版を一言で語るのは難しいのですが，あえて一言で言うならば，文字があるところ（黒いところ）とないところ（白いところ）のバランスが全体的によい，つまり遠目で見たときに均等な「灰色度合」で見える，となると思います．

出力を細かく見ると，preferred や document の途中で分綴（ぶんてつ）（ハイフネーション）されています．仮に分綴しないとすると，

(ニ) Mr. Drofnats—or “R. J.,” as he preferred to be called—was happiest when he was at work typesetting beautiful documents.

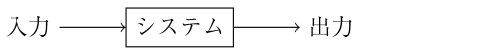
のようになり，ところどころ「白く」見えます．

改行できる場所は，単語間，分綴可能な位置のいずれかに限られます．どこで分綴してよいかは，辞書に当たらなければわかりません．なので，入力の時点で分綴できる位置すべてを人間に指定させるのは現実的ではありません．入力は，ノートなどに手書きするときと同じように，改行するときは単語間で行えばよいとしましょう．

改行する位置を自動で決めて，できるだけ均等な「灰色度合」を達成できたら，満足できそうです．そこで，この問題を改行位置決定問題と呼ぶことにします．次の節では，きちんと改行位置決定問題を定義してみましょう．

3. 改行位置決定問題のモデル化

この節では，改行位置決定問題をモデル化していきます．つまり，前節冒頭の文章のみが解決すればよいわけではないので，改行位置決定問題として考えたい普遍的な部分を抽出し[1]，さらに問題を定量的に捉えること[2]を目指します．加えて，改行位置決定問題を解くためのシステム（たとえばソフトウェア）があるとすると，



のように入力から出力を得るためにシステムを使うという構造になります．そこで，改行位置決定問題を定義するときも，入力と期待する出力をきちんと意識します．

改行位置決定問題では，対象のテキスト一段落分が与えられるものとします[3]．また，次のものは既知とします：

- 分綴できる位置[4]，
- 各行の仕上がりの行の長さ（行長）[5]，
- 各単語を自然に組んだときの長さ[6]．

では，調節できるものは何かというと，単語間の空白量です．空白量に対する標準の値と，許容される伸び量，縮み量は既知とします．もう一つ調整できると暗に言えるのは，段落末の位置です．つまり，

であればよく，以下のように段落末が，基準となる行長いっぱいになる必要はありません．

[1] 極端な言い方をすると，1 文字目が M の文章だけ適用できる問題が定義できても嬉しくありません．

[2] 卑近な例で言うと，クラスの中で「賢い」人を挙げてほしいというお題が出されたら，人によって回答がバラバラになりそうです．○月○日の数学のテストの点数が最も高かった人を挙げてほしいとなれば，（最高得点を取った人が複数いなければ）何の紛れもなく一人に決まります．

[3] 前節で挙げた例で言えば，入力は，次のように与えます：
```
Mr.~Drofnats---or ``R. J.,'' as he preferred
to be called---was happiest when he was at work
typesetting beautiful documents.
```
TEX が開発されたころにはディジタル世界で多言語を扱うための仕組み（Unicode など）が整っておらず，入力はおおよそタイプライタで表現できる範囲内の文字で構成しなければなりませんでした．そのためにいくつかの約束ごとがあり，この入力でも何種類かが垣間見えます．~（チルダ）は，改行してはいけないという条件の付いた，単語間空白を指定するための記号です．自動で改行を決めようという問題で，改行したくないという意思を人間が入れられるわけです．--- はエムダッシュ（M とおよそ同じ幅のダッシュ）に，‘‘（バッククォート二つ）は開きのダブルクォーテーションマークに，’’ は閉じのダブルクォーテーションマークになります．

[4] TEX では，英語の分綴のルールを分析し，一部の単語を除いてアルゴリズムで分綴位置を推定させました（*The TEXbook* [2] の Appendix H 参照のこと）．いまの計算機環境であれば，辞書を丸ごと読み込んで用いるということもできるでしょう．

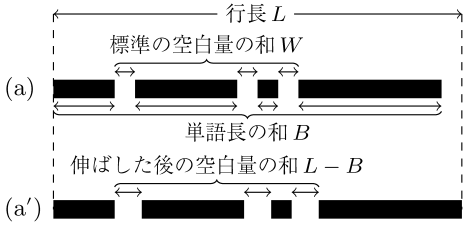
[5] 基準となる行長（和文の組版について詳しい JLREQ [3] の用語で言えば，「基本版面の行長」）のほかに，段落先頭での字下げ（インデント）が先に決定していることを指します．

[6] 自然に組むとは，次のようなことを考慮することを指します．一つは，たとえば `difficult` という単語を自然に組むと，フォントの定義によりますが，この原稿のフォントでは，中ほどの `ffi` がリガチャ（合字）されて，difficult となります（difficult ではなく）．もう一つは，たとえば `VA` のような並びだと，文字の形の影響で，V と A をそれぞれの文字の幅を保って並べると (VA) 空きすぎに見えるので，少し詰めて（カーニングして）VA のように組まれます．

改行位置決定問題で制御できるのは，改行の位置です．ある場所で改行すると，デメリットが発生すると考えることにしましょう．そして，そのデメリットの総和が最小となる改行の位置たちが，最も「よい」と決めるのです．

デメリットの一例は，空白を伸び縮みさせなければならないことに対する悪さ，badness です．行長 L に対して，各単語を自然に組んだときの長さの和を B，単語間の空白量の標準値の和を W，許容される伸び量の和を P，許容される縮み量の和を N としましょう．

空白を伸ばさなければならない場合を先に説明します．標準の空白量を用いて単語を詰めたところ，(a) のようになったとします．空白を伸ばして (a′) のように組みたいわけです．

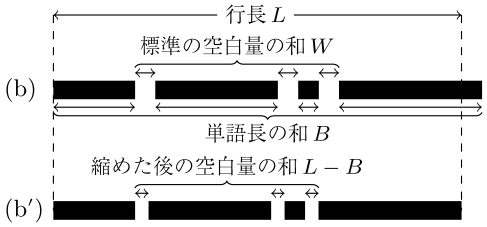


このとき，［許容される伸び縮み量］に対する［伸び縮みさせる量］を表現する，r という指標を導入します．伸ばす場合は，

$$r = \frac{\text{伸ばす量}}{\text{許容される伸び量}} = \frac{(L-B)-W}{P}$$

のように計算します．

空白を縮めなければならない場合は，標準の空白量を用いて単語を詰めたところ (b) のようになり，空白を縮めて (b′) のように組みたいとなります．



縮める場合は，指標 r は以下のように計算します：

$$r = \frac{\text{縮める量}}{\text{許容される縮み量}} = \frac{W-(L-B)}{N}.$$

許容される伸び縮み量と書いてきましたが，やむを得ない場合は r が 1 より大きな値を取ることも許されます．ただし縮めるほうは，単語間空白がある程度残っていないと，単語と単語の切れ目がわからなくなってしまいます．そこで，許容される縮み量だけ空白を縮めても行長に収まらない場合は，行長をはみ出して組むことにします（できる限り避けるべきです）．

badness としては，r をそのまま使うと，伸ばしすぎまたは縮めすぎに対する抑制が効きにくいことから，r^3 に比例した値を使うことにします．なお，計算機ではあまり大きくない整数値を用いるほうが四則演算が速いため，具体的には $\min(100r^3, 10000)$ の値を用いることにします（$100r^3$ は適当に整数値に丸めます）．

後の説明のため，badness の値によって行の仕上がりパターンに名前を付けておきます（表 1）．

表 1　badness の値による行の仕上がりパターン名

伸／縮	badness b の範囲	名前
伸び	$100 \le b$	very loose
伸び	$13 \le b \le 99$	loose
両方	$b \le 12$	decent
縮み	$13 \le b$	tight

badness b 以外にも，以下の要素を考えます：

- 改行すること自体の悪さ l（改行するときは 10），
- 分綴することに対するペナルティ p（分綴するときは 50，しないときは 0）．

また，以下のような事象に対してデメリットを加算します．

d_1：行の仕上がりパターンが隣接していないとき 10000（たとえば，very loose の前の行が decent または tight であるとき）

d_2：2 行続けて分綴されるとき 10000

d_3：最終行の直前の行に分綴があるとき 5000

以上の情報を基に，ある場所で改行すると発生するデメリット d を，

$$d = (l+b)^2 + p^2 + d_1 + d_2 + d_3$$

と計算します[7]．

これらのデメリットの定義により，もう一つ工夫できることがあります．それは，badness が大きすぎる場所と，行長をはみ出して組まざるを得なくなる場所では，原則として改行しないとすることです．前節で

[7] 正確には，ペナルティ p に関する部分でもう少し複雑なことをしています．

挙げた例では，単語の切れ目や分綴可能な位置は 28 にのぼります．これらすべての場所で改行するかどうかを考えるとなると，素朴には 2^{28}（約 3 億）の組合せを確かめなければなりません．しかし，badness が大きすぎる場所を除くと，改行する位置の候補は 5 カ所になります．確かめなければならない組合せの数は単純に計算しても 2^5（約 30）にまで減ります．

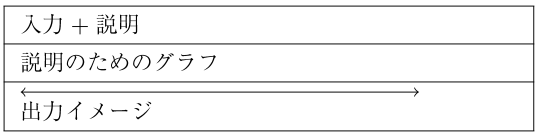
現実の文章は，1 段落にもっと多くの単語が含まれることもあるでしょうし，本 1 冊の組版をすることを考えると，確かめなければならない組合せの数をもっと減らして，計算時間を短くしたいところです．次節では，高速に改行位置を決定するためのアルゴリズムを説明します．

4. アルゴリズム

前節で述べた問題設定では，ある場所で改行して行を定めると，その改行で増えるデメリットは，新たに定まる行 v とその前の行 u の情報のみで計算できます．つまり，u より前の行までの改行の仕方にはよりません．よって，v の時点でのデメリットの総和を最小にするためには，u の時点でのデメリットの総和が最小となる改行の仕方に v を付け加えればよいということになります．異なる言い方をすると，v の行まででデメリットの総和が最小になっているならば，u の行までの改行の仕方は，u の行まででデメリットの総和が最小になるという性質を満たします．

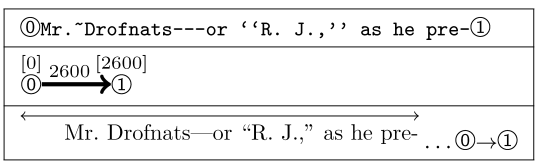
このような性質——ある最適化問題 X に対応してサイズの小さな問題 X' を定めることができて，X の最適解のうち X' に含まれる部分を取り出すと X' の最適解になっているような性質——を満たすとき，動的計画法というアルゴリズムを使うと高速に最適解を計算することができます．動的計画法では，サイズの小さな問題から順に，最適解を計算しては記録し，重複した計算をしなくて済むようにすることによって高速化します[8]．

以下では，2 節で導入した例を用いて，どのようにアルゴリズムが動くか説明します．本稿では，アルゴリズムが動いていく様子を示すために，それぞれの場面を次のようなフォーマットで描き記します．これを仮に場面図と呼ぶことにします．



場面図下段の「出力イメージ」上の矢印は，基準となる行長を示します．

まずは 1 行目の決定です．



1 行目を①の位置で改行した場面は上のようになります．`preferred` という単語は，pre の後でのみ分綴可能だということになっています．仮に `preferred` の前で改行した場合と後で改行した場合の出力イメージは次の (c)，(d) のようになります．

(c) Mr. Drofnats—or “R. J.,” as he

(d) Mr. Drofnats—or “R. J.,” as he preferred

(c) では空白をたくさん伸ばさなければならず，badness が 1472 と大きすぎて改行の候補になりません．(d) では空白を最大限縮めても行長を超えてしまうので，改行の候補になりません．

ここで，場面図の読み方を少し説明します．

場面図上段には，入力に対応させて，どの場所で改行したときを考えるかを示します．改行を考える位置を丸数字（①，②，...）で示します．段落の初めを指し示すときは，便宜的に⓪を使います．

場面図中段には「説明のためのグラフ」を描きます．グラフといっても，x–y 軸を取って描く棒グラフや折れ線グラフのようなものではなく，点と辺の接続関係を用いて，物事の関係性を論じたりネットワーク[9]の性質を調べたりする，グラフ理論・グラフアルゴリズムの世界[10]の「グラフ」です．①の位置で改行したときには，便宜的に置いた始点⓪の次に①で改行するため，⓪から①へ，向きの付いた辺を引いています．辺の周囲に書いてある数値は，その改行で増えるデメリットを示しています．点の周囲に [~] で挟んで書いてある数値は，その点までのデメリットの総和の最小値です．

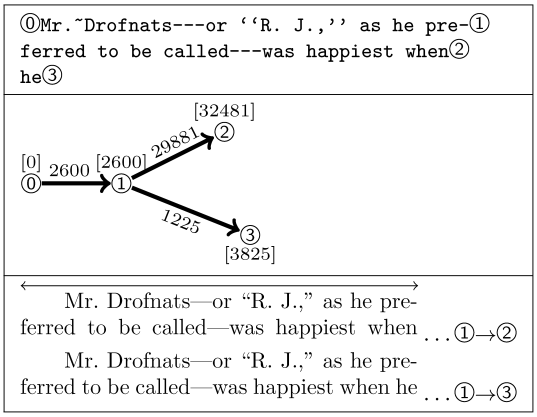
[8] 動的計画法は，アルゴリズムやオペレーションズ・リサーチの多くの書籍で触れられています（たとえば，杉原 [4]，秋葉ら [5]，松井ら [6]）．

[9] 鉄道網，道路網や電力網などを思い浮かべてください．

[10] 参考図書は山ほどありますが，たとえば，伊理ら [7] や浅野 [8, 9] を挙げておきます．

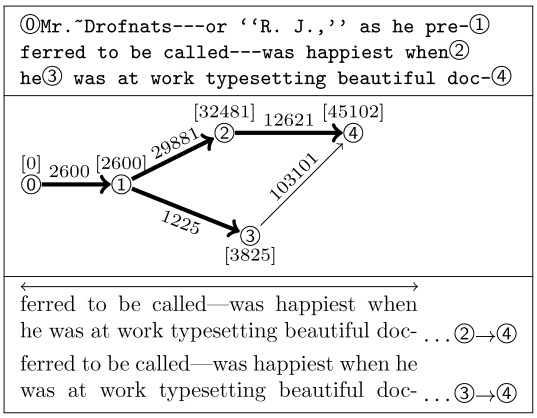
最小値を与えるときに選ばれる辺を太く示します（もう少し後の例を見て理解してください）．

次に，2 行目です．



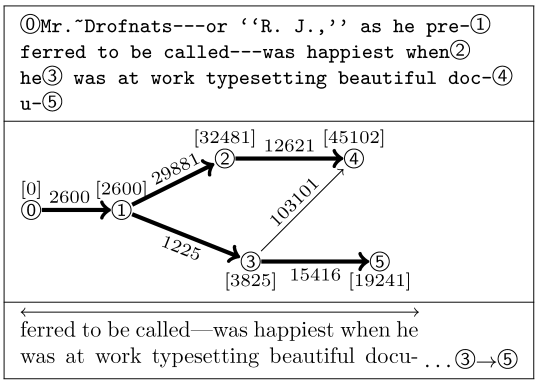
2 行目を決めうる改行は，②と③の 2 カ所あります（紙面節約のため同時に描きました）．いずれも，①の後の改行となるべきなので，グラフには①から②への辺と①から③への辺が張られます．

次に 3 行目です．

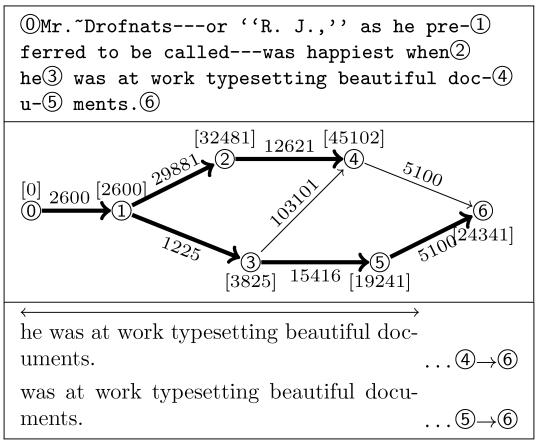


3 行目を決めうる改行は，②から数え始めると④があります．実は④は，③から数え始めても候補になりえます．そこでそれぞれのデメリットを計算すると，②から④への増分は 12621，③から④への増分は 103101 となるので，デメリットの総計が小さくなる②→④ を選びます．④までのデメリットの総和も，32481＋12621 ＝ 45102 と計算し，記憶しておきます．

②から数え始めた，次の改行候補は④しかありませんが，③から数え始めるともう一つ改行候補があります．



そして，段落末を迎えます．



最後の改行④，⑤の両方から段落末になりえます．いずれも挟む空白がありませんし，最終行の 1 行手前で分綴されている点も共通していることから，デメリットの増分は 5100 と同値になります．④，⑤でそれぞれ記憶していたそこまでのデメリットの総和と足し合わせて，総和が小さくなる⑤→⑥ を選びます．

終点である⑥から，選ばれた辺を逆向きにたどると，①，③，⑤と改行するのがよいと求まります．2 節冒頭のクイズの答えは，（イ）が最もよいとなります．

5. おわりに

5.1 オペレーションズ・リサーチとモデル化

一見数値とは関係なさそうな組版の世界に，デメリットという定量的な指標を導入することで，何が最適なのかを明確に定義でき，既存のアルゴリズムを適用することで最適な改行位置を高速に見つけられるようになりました．

オペレーションズ・リサーチは，定量的に物事を扱って解決策を見つけるための方法論の集まりです．では，

この話で肝になったのはどんなところでしょうか？

一つは，既存のアルゴリズムや（数学的な）道具を知っているというところでしょう．動的計画法という単語自体，初めて知ったという人も多いに違いありません．アルゴリズムの説明では，高校の段階では見慣れないであろうグラフを用いました[11]．

もう一つは，現実の問題をどうやって既存のアルゴリズムや道具に当てはまりのよい形で捉えるかという，技術または技能だと言えると思います．このことを指して，モデル化（モデリングとも言います）という用語を使うこともあります．

改行位置決定問題の例では，「行ごとにその行と直前の行で定まるデメリットを定義し，その総和を最小化したものを最適解とするという定量的な問題を定義した」ことから動的計画法が使え，高速に解けるよいモデル化ができたということになります．

ところで，説明を読んでいるうえでは，問題を解き始めたら暗黙の仮定がないようにするというのが，オペレーションズ・リサーチの文化だと思います．しかし，実際に問題を解く立場（モデル化する立場）になったときには，何が真に使える情報なのか，どんな結果が求められるのかを聞きまわったり，何を定量的に考慮するか，どんな既存の問題で捉えられるか，どんな既存の手法が使えるかといったことを試行錯誤したりすることになるでしょう．

既存の問題や既存の手法は，知識としてどんどん蓄える（勉強する）必要があります．それと同時に，発想の柔らかさや，何をもって「よい」と考えるのかといったセンス，調査に協力してもらえるような，ものの聞き方や人柄といったものも，ないよりはあるに越したことはありません．オペレーションズ・リサーチに興味をもった高校生の読者のみなさんには，貪欲にいろいろなものを吸収できる時期を大いに活用してもらって，将来ぜひオペレーションズ・リサーチ分野で活躍していただければと思います．

5.2 TEX について

最後になりますが，ここで紹介したような組版を手元でも実現したいと思う方もいるかと思いますので，題材にした TEX についても少し触れておきます．

本稿では，ずっと TEX と言及してきましたが，われわれが通常和文の原稿執筆に使うのは，pLATEX と呼ばれるものです．TEX は，改行位置決定問題のような，組版において非常に原理的な機能を提供していますが，長文を執筆するうえでは，章立てをしたり，ページ番号の参照をテキストの増減に追随させたりなど，もっと高度な機能があったほうが便利です．TEX には，マクロを記述することで機能を追加できる仕組みがあります．テキストの特定の部分を，強調したり章見出しとして章立てしたりといった，いわゆるマークアップ機能を提供するのが，LATEX マクロです．また，アスキー社（当時）が開発した，日本語組版に対応した TEX の拡張を pTEX と呼びます．pTEX と LATEX の合わせ技が pLATEX で，日本語組版をマークアップ機能をもって利用できます．

(p) LATEX の日本語の解説書としては，「LATEX 2ε 美文書作成入門」シリーズが有名です．約 3 年ごとに改訂を繰り返しており，現在は改訂第 6 版 [10] が最新です．システム一式を計算機にインストールするための DVD-ROM が付いていますので，TEX の世界を気軽に体験し始められます．

今の時代なら，Web サービスのほうが身近かもしれません．Cloud LaTeX [11] や TeX を使ってみよう [12][12]を参考までに挙げておきます．

謝辞　筆者をオペレーションズ・リサーチの世界に誘ってくれたのは，久保・松井の書籍 [13] でした．高校時代には 1/4 も読めなかった本が，大学に入ってすんなり読めるようになって[13]，成長したことを感じさせてくれたことも，この世界を楽しむきっかけになったのかもしれません[14]．私に書籍を贈ってくださった知り合いと，著者でその後指導教員になっていただいた松井知己教授にこの場を借りて御礼申し上げます．

また，本稿の草稿を読んでコメントをいただいた，土村展之博士と阿部紀行准教授および本特集号編集担当の宮代隆平准教授ほか編集委員の皆さまに感謝いたします．

[11] 筆者の頭の中では，改行位置決定問題は「動的計画法が使えるから高速に解ける」というよりも，グラフの世界の言葉で「非巡回有向グラフ上の最短路問題だから高速に解ける」と理解していることもあり，グラフを用いてアルゴリズムの説明をしました．表現の仕方を多く知っているということも，問題を多面的に理解するために有利です．

[12] このサイトでは，入出力の情報が秘匿化されないので，あくまでも TEX を試す目的でのみ用いることを想定しています．

[13] なので，（自分を引合いに出して語るのはどうかと思いますが，）本稿を途中飛ばし飛ばしに読んでここまでたどり着いた高校生がいても何ら不思議はありません．読み切れなかったからといって悲観することはありません．

[14] それだけでは，どんな分野でもよかったのではないか，という結論になってしまいますが，もちろんそれだけではなく，「問題を解き始めたら暗黙の仮定がない」と感じられたことも大きな理由を占めたように思います．

参考文献

[1] TeX Users Group, TUG 2013（第 34 回 TEX Users Group 年次大会）Booklet，p. 18, 2013, http://tug.org/tug2013/booklet-web.pdf

[2] D. E. Knuth, *The TEXbook*, Addison-Wesley Professional, 1984.（斉藤信男監修，鷺谷好輝訳，『改訂新版 TEX ブック―コンピュータによる組版システム―』，アスキー，1992.）

[3] 千葉弘幸ほか（編），『日本語組版処理の要件（日本語版） W3C 技術ノート 2012 年 4 月 3 日』，World Wide Web Consortium (W3C), 2012, http://www.w3.org/TR/jlreq/ja/

[4] 杉原厚吉，『データ構造とアルゴリズム』，共立出版，2001.

[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（第 2 版）―問題解決のアルゴリズム活用力とコーディングテクニックを鍛える―』，マイナビ，2012.

[6] 松井泰子，根本俊男，宇野毅明，『入門オペレーションズ・リサーチ』，東海大学出版会，2008.

[7] 伊理正夫，白川功，梶谷洋司，篠田庄司ほか，『演習グラフ理論―基礎と応用―』，コロナ社，1983.

[8] 浅野孝夫，『情報の構造（上）―データ構造とグラフアルゴリズム―』，日本評論社，1994.

[9] 浅野孝夫，『情報の構造（下）―ネットワークアルゴリズムとデータ構造―』，日本評論社，1994.

[10] 奥村晴彦，黒木裕介，『改訂第 6 版 LATEX 2ε 美文書作成入門』，技術評論社，2013.

[11] アカリク，Cloud LaTeX, https://acaric.co.jp/acaric-cloudlatex/

[12] 奥村晴彦監修，「TeX を使ってみよう」，http://oku.edu.mie-u.ac.jp/~okumura/texonweb/

[13] 久保幹雄，松井知己，『シリーズ「現代人の数理」14 組合せ最適化［短編集］』，朝倉書店，1999.

