数理情報工学演習第一 C 最終レポート（再提出）

所属: 数理情報工学コース 3 年
学生証番号: XX-XXXXXX 氏名: 佐藤 瞭
提出日: 2019/08/08
再提出: 2019/08/15

修正内容
修正したのはの「**5. 考察**」の部分である。
- $O(\log N^2)=O(\log N)$ なので、そのように書き直した。
- グラフの縦軸のスケールを修正した。それに伴い考察内容も修正した。

1. 課題内容
2 つの文字列の編集距離を求めるアルゴリズムの実行時間の比較を行う。具体的には、動的計画法とグラフの最短路探索を比較する。

入力:
- 標準入力から与えられる。
- 1 行目: 1 つ目の文字列。
- 2 行目: 2 つ目の文字列。

出力:
- 標準出力に表示する。
- 1 行目: 入力された文字列の編集距離。

なお、2 つの文字列の編集距離とは、一方の文字列に①文字の挿入、②文字の削除、③文字の置換の操作を繰り返してもう一方の文字列に変換する際の、最小の操作回数のことである。
以下に例を記す。
「ACGTT」と「CGCAT」の編集距離は 3 である。
1. ACGTT の A を削除→CGTT
2. CGTT の「G」と「T」の間に C を挿入→CGCTT
3. ひとつめの T を A に置換→CGCAT

2. 手法
以下の 2 通りの組み合わせで実験した。入力されるふたつの文字列を A, B とする。
①動的計画法(スクリプト: dp.c)。(A の長さ+1)×(B の長さ+1)の行列を用意し、[i, j]要素の値を、A の頭から i 文字目までの文字列と B の頭から j 文字目までの文字列の編集距離とすることで、i や j が小さい要素から逐次的に部分問題の解を求めていく。
②二分ヒープによるプライオリティーキューを用いたダイクストラ法(スクリプト: dijkstra.c)。先述の動的計画法の行列の、各要素の右隣や下隣、右下隣の要素とのコストの差に注目し、このコスト差をグラフの枝の重みとみなすことで最短路探索の問題に帰着する。ダイクストラ法での、探索対象のノード決定を高速に行うために二分ヒープを用いた。

実行時間は、bash の time コマンドで測定し、real の値を用いた。
以下にマシンのスペックを示す。学科で貸与されたラップトップの Ubnutu の環境で行った。

OS: Ubuntu 18.04.2 LTS 64bit, Memory: 15.6 GiB, Processor: Intel Core i7-6820HQ CPU @ 2.70GHz × 8

3. データ
円周率の、小数第 1 位から小数第 N 位までの数列と、小数第(N+1)位から小数第(2.2N)位までの数列を入力とした。動的計画法では N ≦ 50000、ダイクストラ法では N ≦ 9000 の範囲でデータを生成した。入力文字列の長さは N と 1.2N となるため、各アルゴリズムで用いる行列、あるいはグラフの要素数(ノード数)は $1.2N^2 = O(N^2)$ となる。この「N」による表記は次項以降でも用いる。

4. 実験結果

手法ごとの実行時間を図 1 に示す。「文字列の長さの平均」は、「3.データ」の N を用いると、1.1N となる。

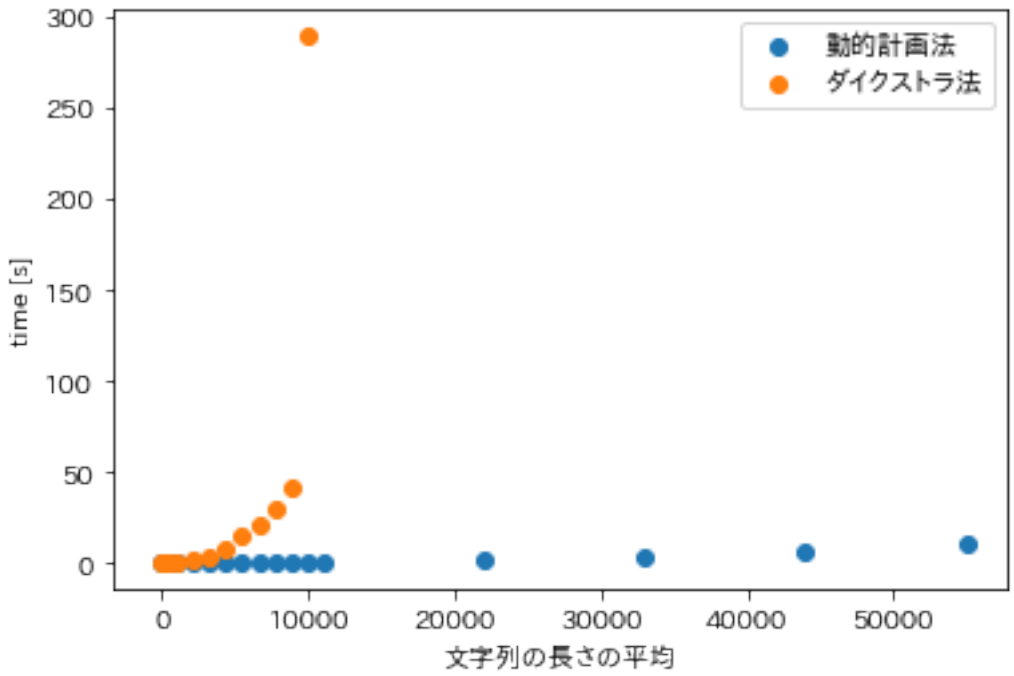

図 1 手法ごとの文字列の編集距離計算の実行時間

5. 考察

図 1 をみると、文字列を長くしていくと、ダイクストラ法は動的計画法と比べて計算時間の増加の勢いが大きいことがわかる。そのため、ダイクストラ法と動的計画法のうち、文字列の編集距離により有用なのは動的計画法であるといえる。

手法の比較を定量的に行うために、各手法の計算量のオーダーを比較してみる。

まず、動的計画法の計算量を見積もる。動的計画法は、「2.手法」で説明しているような行列が完成したときに計算が終了する。「3. データ」で述べたように、行列の要素数のオーダーは $O(N^2)$ である。あとは、各要素の計算にかかる時間を調べれば良い。今回の動的計画法は、各要素ごとに以下の処理を行っている[1]。現在の要素の位置を[i, j]とする。

1. (右隣要素について)
   部分問題の 2 つの文字列のうち一方に、新しく 1 文字が追加されるので、編集距離は 1 増える(挿入あるいは削除のぶん)。
2. (下隣要素について)
   1.と同様に、編集距離は 1 増える。
3. (右下隣要素について)
   部分問題の 2 つの文字列の両方に、新しく 1 文字がそれぞれ追加される。追加される文字が同じであれば編集距離は変化しない。追加される文字が異なる場合は置換してやる必要があるので編集距離は 1 増える。

この一連の操作は文字列の長さに依存しないので、定数時間で実行できる。よって、動的計画法の計算量は $O(N^2)$ と予想することができる。

つぎに、ダイクストラ法の計算量を見積もる。グラフのノード数は動的計画法の行列の要素数と同じで $O(N^2)$ である。エッジは各ノードから高々 3 本しか出ておらず、各要素から出る枝の重みは上述の動的計画法の処理 1～3 により、各要素について定数時間で実行できるので、グラフ作成に必要な計算量は $O(N^2)$ である。二分ヒープを用いたダイクストラ法では、①暫定距離が最小のノードを一つ選び、②そのノードから出ている各枝に対して、枝の先のノードの暫定距離の更新及びヒープの更新を行う。①の操作はノードの数だけ行われる、すなわち $O(N^2)$ 回行われる。最小値の取り出し 1 回にかかる時間は $O(\log N^2)=O(\log N)$ である。②の操作は、最大でエッジの数だけ行われる。エッジは各ノードから高々 3 本しか出ておらず、エッジの総数のオーダーはノードの数のオーダーと同じ $O(N^2)$ である。ヒープの更新 1 回にかかる時間は $O(\log N^2)=O(\log N)$ である。
よって、全体での計算量は $O(N^2 \log N)$ と予想できる。

両手法について求めた計算量の妥当性を、実験の結果から評価する。横軸を計算量のオーダー、縦軸を(実行時間/計算量のオーダー)としてプロットする。予想される計算量が妥当であれば、プロットされた点は水平な直線上に位置する。まず、動的計画法についてのプロットを図 2 に示す。

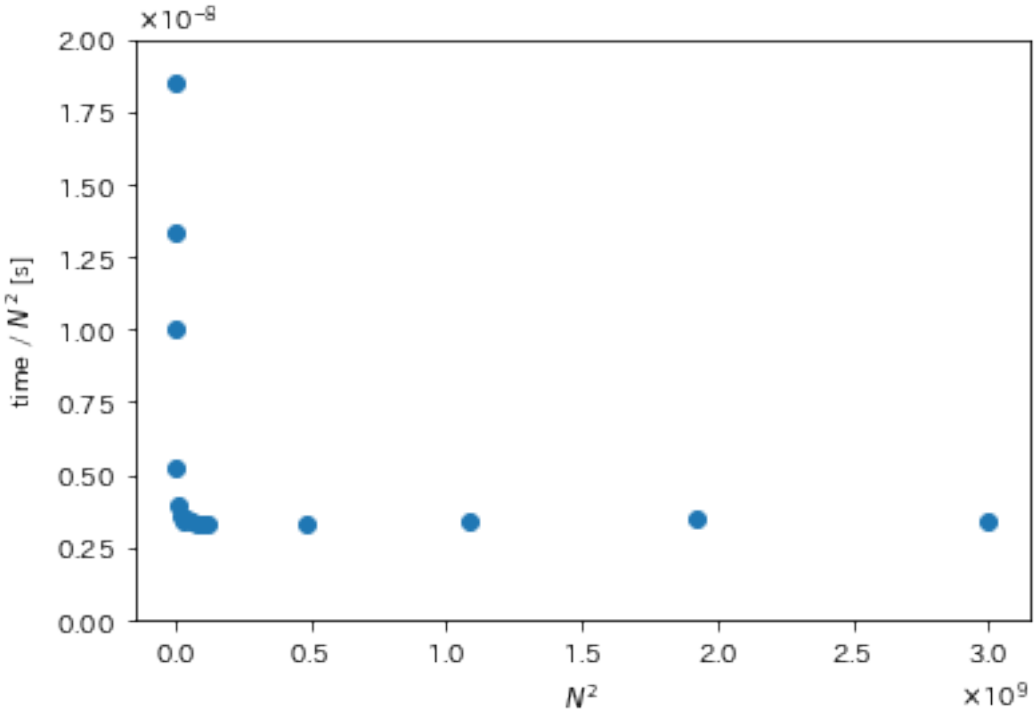

図 2 計算量のオーダーと実行時間の比較(動的計画法、全データ)

まず、図 2 をみると、データのサイズが小さいところで、(実行時間/計算量のオーダー)の値が急激に小さくなっていることがわかる。データサイズが小さいうちは、動的計画法の処理にかかる時間に対する、標準入出力の処理にかかる時間の比率が高くなることが考えられる。そのため、N≧10000 のデータに絞ってプロットし直すことにした。その結果が図 3 である。

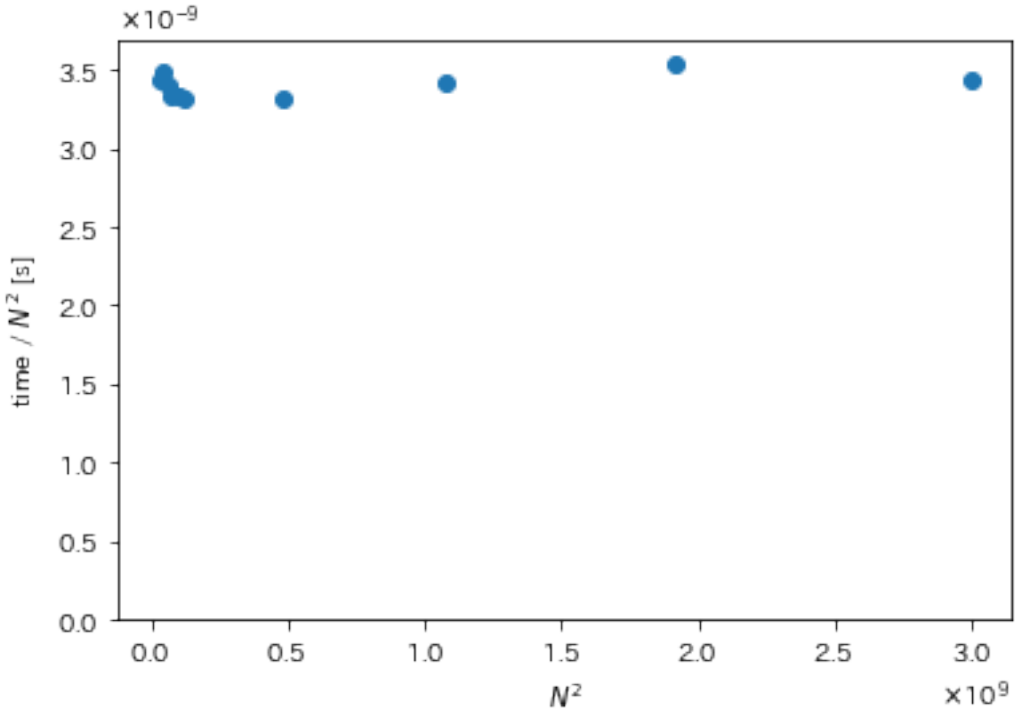

図 3 計算量のオーダーと実行時間の比較(動的計画法、N≧10000)

図 3 をみると、大きなデータに対してプロットされた点は概ね水平な直線上にあるように思われる。そのため、動的計画法の計算量のオーダーは $O(N^2)$ である、という予想は妥当であると判断した。

次に、ダイクストラ法についてのプロットを図 4 に示す。

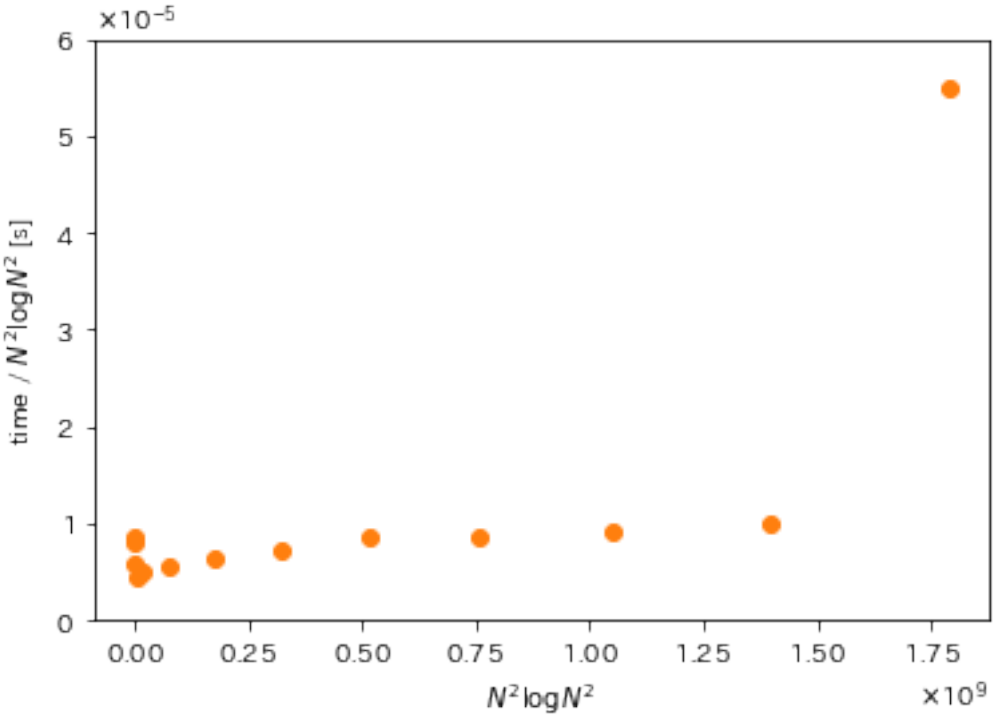

図 4 計算量のオーダーと実行時間の比較(ダイクストラ法、全データ)

図 4 をみると、N=9000 の最もサイズが大きいデータだけ値が突出して大きい。この原因としては、大きなメモリは中央演算装置から「遠い」位置にあるため、大きなデータに対する処理では、動的計画法のような、メモリに順番にアクセスする処理を除いてメモリアクセスが極端に遅くなることが考えられる（この考察は、一回目に提出した際に TA の隈部さんからコメント頂いたものである）。そこで、N=9000 のデータ以外について再度プロットしてみる。その結果が図 5 である。

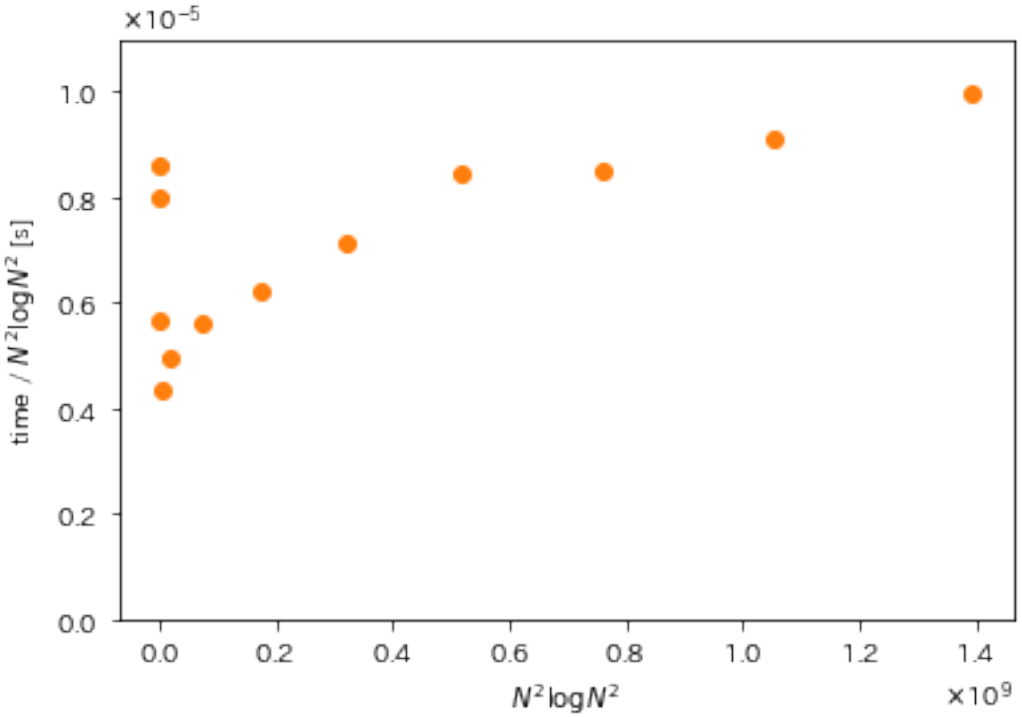

図 5 計算量のオーダーと実行時間の比較(ダイクストラ法、N=9000 のデータ以外)

図 5 をみると、データが小さいところでは急激な値の減少がみられ、その後はデータサイズが増えるにつれて値はゆるやかに増加していくことがわかる。仮に、今回のデータに対するダイクストラ法の計算量のオーダーは $O(N^2 \log N)$ である、という予想が正しいとすると次のような説明ができる。まず、小さなデータでの値の減少は、動的計画法と同様に、標準入出力の処理にかかる時間の、ダイクストラ法の処理にかかる時間に対する比が、データが小さいうちは大きいためであると考えることができる。次に、その後のゆるやかな増加については、前段落で述べたような、メモリが大きくなることによるアクセス速度の低下が原因として考えられる。

しかし、以上の原因が必要十分であるかは判断できなかったため、計算量のオーダーは $O(N^2 \log N)$ であるかどうかは判断できないと考えた。より詳細な評価のためには、アルゴリズムだけでなく、メモリアクセスなど計算機の仕組みに対する考察も必要だと考えられる。

6. 参考文献

[1] 講義資料「最終レポート」pp.2-3.