提出日　令和2年08月31日

アルゴリズムとデータ構造　第12回課題レポート

学籍番号（A19117）

氏名（永尾優磨）

【課題】

1. BM法について調査し、その内容をレポートにまとめよ。

BM法は、かつて「最速」と言われた文字列検索アルゴリズム

どのようなものかを説明していきます。

図1.1の上のテキストからPatternを順番に見ていく。

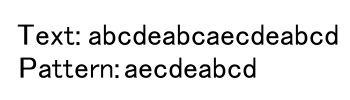


図1.1　文字列について

文字列の検索なので、Textを前から順番に見ていく

説明上、この「見ている位置」を「ポイント」と呼ぶことにする(通常のプログラムで使うポイントとは違う。)

このBM法が他のアルゴリズムと違うのはポイントの最初の位置(図1.2参照)だ。パターンの末尾にポイントを置き、末尾と一致しているところをまず探す。他の一般的な探索アルゴリズムは最初の文字を見る。

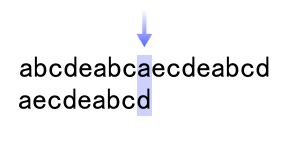


図1.2 BM法のポインタの位置

今回は”a”と”d”なので一致しない。末尾が一致しない場合、その位置にパターンの中の最も右寄りにある同じ文字合わせる(同じ文字がなかったらパターンの長さ分だけ一気に移動させる。)そうすると、次に一致する可能性がある場所まで行きに移動できる。このように移動させると、最大で文字の長さ、(図1.3なら9文字分)移動させることができる。そのため効率的だ。

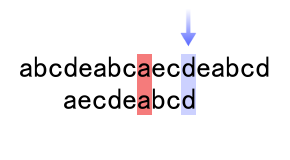


図1.3　BM法のポインタ位置の移動

BM法は、文字をスキップして、少ない計算量で実装することができるアルゴリズムです。

1. BM法を用いた文字列の検索プログラムを作成し、結果を確認せよ。

【作成したプログラム】

public class BM {

    public static void main(String[] args) {

        String text = "National INstitute of Technology,nuaa Numazu College";

        String key = "Numazu";

        String sec = "sjkoj";

        BoyerMoore res = new BoyerMoore();

        res.search(text, key);

        res.search(text, sec);

    }

}

プログラム1　BM

public class BoyerMoore {

    public void search(String text, String key) {

        int p = res(text, key);

        if (p != -1) {

            System.out.printf("見つかりました。pos=%d\n", p);

        } else {

            System.out.println("見つかりません。");

        }

    }

    public int res(String text, String key) {

        int[] skip = new int[256];

        int tlen = text.length();

        int klen = key.length();

        for (int i = 0; i < 256; i++) {

            skip[i] = klen;

        }

        for (int i = 0; i < klen - 1; i++) {

            skip[key.charAt(i)] = klen - 1 - i;

        }

        int tp;

        int kp;

        tp = klen - 1;

        while (tp < tlen) {

            System.out.printf("text=%s\n", text);

            System.out.printf("key =");

            for (int i = 0; i <= tp - klen; i++) {

                System.out.printf(" ");

            }

            System.out.printf("%s\n\n", key);

            kp = klen - 1;

            while (text.charAt(tp) == key.charAt(kp)) {

                if (kp == 0) {

                    return tp;

                }

                tp--;

                kp--;

            }

            tp += Math.max(skip[text.charAt(tp)], klen - kp);

        }

        return -1;

    }

}

プログラム2　BoyerMoore

【プログラムの解説】

繰り返しif文を書かないために配布されたプログラムの例に書かれていた、BM.java内のif文を関数化し、BoyerMoore.javaに書き加えた。元々あったsearch関数にBMで使われていたif文を書き加えた。resという関数にsearch内に書かれていたプログラムを書き写した。

そうすることで、search関数を使うと、if文の判定を入れることに成功した。

【結果】

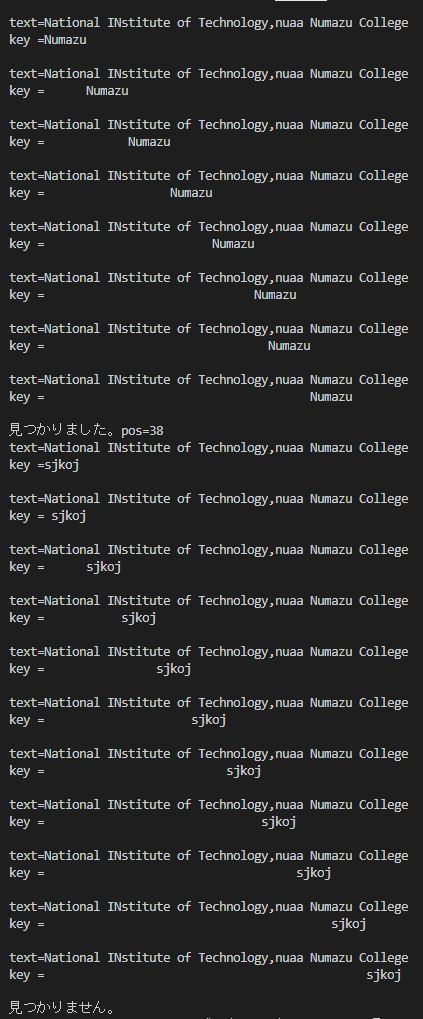


図2　BMの出力結果

【考察】

BoyerMooreの探索方法を始めて知ったときは、衝撃を受けた。なぜなら、今まで知っていた探索方法だと、前から探索をするという考え方だったからだ。BoyerMooreの方法では、探索する範囲を除外することで、検索する範囲を対象を狭めるという考えであり、新鮮だった。自分でプログラムを書くときは、この考え方を使って、少ない仕事量のアルゴリズムを作成していきたいと思う。

【参考文献】

1. 「アルゴリズムとデータ解析の授業スライド」
2. 増井敏克(2020)「Pythonで始めるアルゴリズム入門」翔泳社
3. 「Boyer-Moore法」<<https://g940425.hatenadiary.org/entry/20100522/1274520718>>
4. 「参考文献の書き方」<<http://www7a.biglobe.ne.jp/nifongo/ron/ron_04.html>>