提出日　令和2年07月29日

アルゴリズムとデータ構造　第10回課題レポート

学籍番号（A19117）
氏名（永尾優磨）

【課題】

1.　ハッシュについてレポートにまとめよ

ハッシュとは、0(1)という計算量で探索を行える優れたアルゴリズム。なぜ、ここまで計算量が少なく探索を行うことが可能なのか？

これが原理だ。データを登録するときに、そのデータ自身の値を使って何らかの計算を行って、格納位置（通常、データ小僧としては配列を使うので、その添え字のこと）を決定します。

登録するデータが整数と仮定する。そこで、登録するデータの値をnとした時、ｎを100で割ったあまりを格納位置にするというルールにする。そのときの格納位置は、537 % 100という計算により、37に決定される。

探索するときも、同じ計算をすれば、即座に格納位置を取り出せるというアルゴリズム。

2.　適当なハッシュテーブルを作り、データ投入と検索を行ってみよ。

- 名前に対応する電話番号
- 名前と住所　etc

【作成したプログラム】

```
public class HashElem {
    String key;
    String value;
    Boolean empty;

    HashElem() {
        key = "";
        value = "";
        empty = true;

    }

}
```

プログラム1　HashElem

```
public class HashTable {
    final int size1 = 10;
    final int size2 = 15;
    int n;
    HashElem table[] = new HashElem[size2];
```

```
HashTable() {
    n = 0;
    for (int i = 0; i < table.length; i++) {
        table[i] = new HashElem();
    }
}

public void insert(String k, String v) {
    int h = hash(k);
    if (table[h].empty != true) {
        while (h < size2) {
            if (table[h].empty == true) {
                break;
            }
            h++;
        }
    }
    if (h < size2) {
        table[h].key = k;
        table[h].value = v;
        table[h].empty = false;
        n++;

    } else {
        System.out.println("テーブルがいっぱいです.");
        System.exit(1);

    }
}

public void search(String s) {
    int h = hash(s);
    if (table[h].key.equals(s) != true) {
        while (h < size2) {
            if (table[h].key.equals(s) == true) {
                break;

            }
            h++;
        }
    }
    if (h < size2) {
        System.out.printf("見つかりました。:%2d, %s,¥t %s¥n", h, table[h].key, table[h].value);
    } else {
        System.out.println("見つかりません");
    }
}
```

```
    public void printHashTable() {
        System.out.println("*** ハッシュテーブル ***");
        for (int i = 0; i < size2; i++) {
            System.out.printf("%2d, %s,¥t %s¥n", i, table[i].key, table[i].value);
        }
        System.out.println("*********");
    }

    public int hash(String s) {

        return s.charAt(0) % size1;
    }
}
```

プログラム2　HashTable

```
public class HashApp {
    public static void main(String[] args) {
        HashTable h = new HashTable();

        h.insert("SATO", "佐藤city");
        h.insert("SUZUKI", "鈴木city");
        h.insert("TAKAHASHI", "高橋city");
        h.insert("TANAKA", "田中city");
        h.insert("ITOH", "伊藤city");
        h.insert("WATANABE", "渡辺city");
        h.insert("YAMAMOTO", "山本city");
        h.insert("NAKAYAMA", "中村city");
        h.insert("KOBAYASHI", "小林city");
        h.insert("KATO", "加藤city");

        h.printHashTable();
        h.search("TANAKA");
    }
}
```

プログラム3　HashApp

【プログラムの解説】

HashTableの配列を作り、そこにデータを格納し、出力した。
java上で、変数を出力するための方法を知った。

【結果】

図1　HashMapAppの出力結果

【考察】

ハッシュを用いた探索の方法を知った。

現在、自分がプログラミングをする時は、このような探索の仕方をせず、ライブラリに任せた探索の仕方をすると思う。しかしながら、Hashという新しい方法を知れたことは、良かったと思う。

今回のプログラムの改善点として、HashTableの最大値まで全てを展開している。これをもっとシンプルに記述したいと考える。例えば、入っている配列の全てを出力した時、最大値は、ooで、使っている配列の大きさは、xxですと出力をしたい。現在のプログラムだと、配列の大きさを1000などにした時の出力が長くなりすぎるからである。

【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. 「ハッシュ探索①（チェイン法）｜Programming Place Plus　アルゴリズムとデータ構造編【探索アルゴリズム】　第６章 」<https://programming-place.net/ppp/contents/algorithm/search/006.html>
4. 「参考文献の書き方」<http://www7a.biglobe.ne.jp/nifongo/ron/ron_04.html>