

アルゴリズム論

2017年5月22日

樋口文人

目次

- 辞書
- ハッシュテーブル
 - ハッシュ関数
 - CPUの動作
 - 記号の変換
 - 情報の変換
 - ハッシュ関数の応用

辞書

辞書

- 日常的には...
 - 見出し語とその意味／解説のペアを
 - 見出し語に基づく順番に並べたもの
- 使用法
 - 見出し語を探し, その意味を知る
 - 見出し語の数と種類は決まっている
 - 見出し語が増減したり変化したりはしない

プログラムの中の辞書

- キーと値のペア
 - 数は膨大になり得る
 - 数の増減もあり得る
- キーを使って値を探す
- 連想配列とかハッシュと呼ばれることもある

辞書の構造が使われている例

- CSS
 - プロパティと値
- HTML要素
 - 属性と値
- JSON (JavaScript Object Notation)
 - キーと値
- KVS (Key Value Store)
 - キーと値

キーとは

- データ(値)を探す手がかり
 - キーが数値(整数)なら
 - キーを配列の添え字として使える
 - キーはデータの位置情報
 - 数値の範囲が一定範囲に収まるなら効率的
 - キーが文字列(数値でない)なら
 - 順序は考えられる
 - 整数のように最小の間隔というものはない
 - 整数に変換できると配列が使える
 - データの位置をキーから計算できれば良い

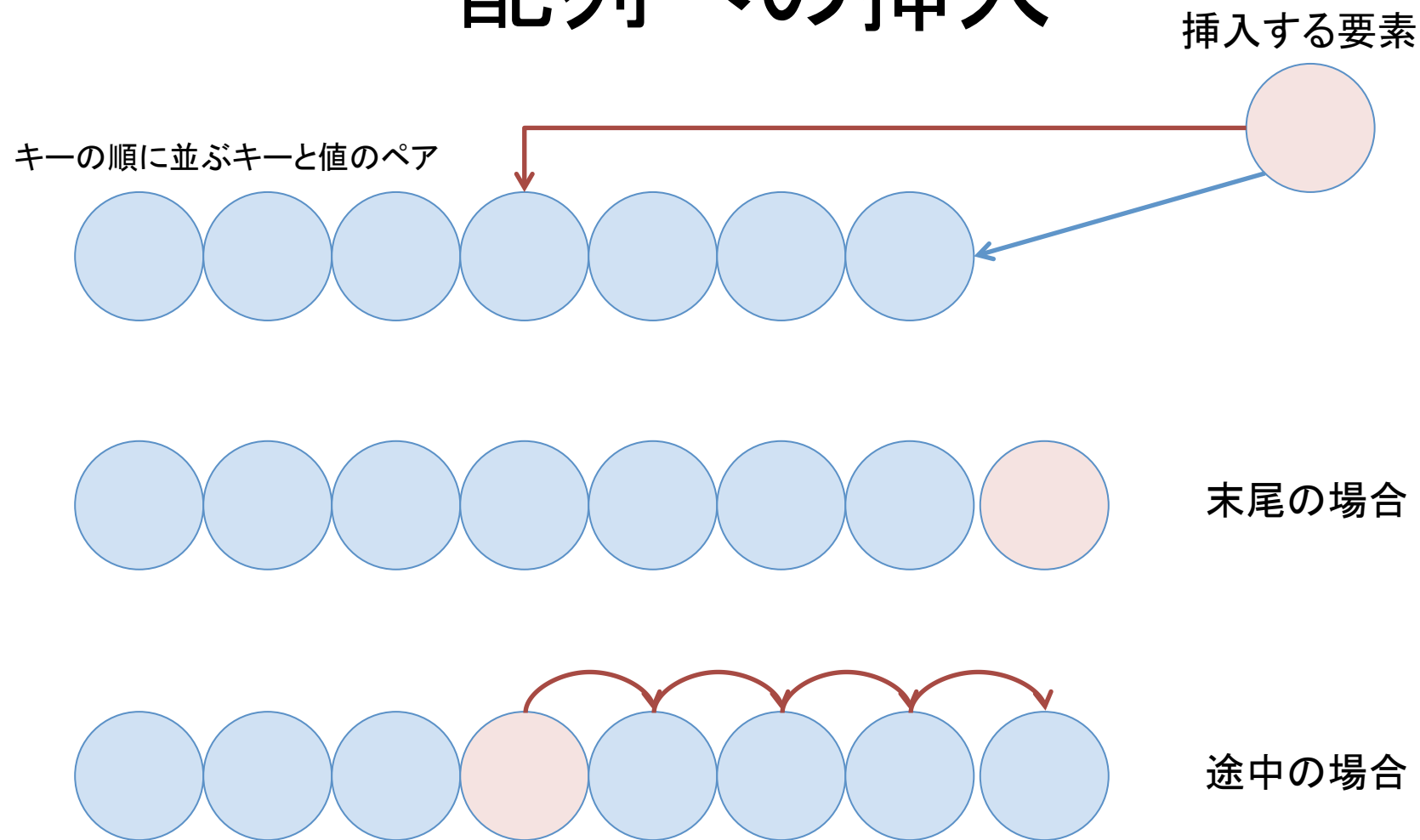
キーの作る空間

- 人名
 - 常用漢字: 約2000文字
 - 4文字の氏名: $2000 \times 2000 \times 2000 \times 2000$
 - 16兆
 - 日本の人口: 1億3千万
- 実際のデータ数はキーの作る空間よりもかなり小さい

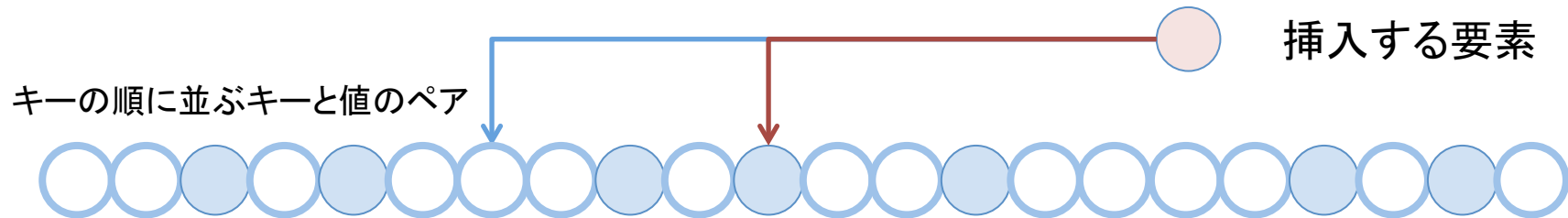
辞書を作る

- 配列に先頭から順にデータを入れるのは得策ではない
 - 挿入・削除に手間がかかる(平均 $n/2$ の移動)
- キー空間と同じ大きさの配列は無駄が多い
 - 1億3千万人に対して16兆個の名前？
- 解決策:
 - キー空間を圧縮して配列を使う
 - 挿入・削除が容易な配列以外のデータ構造を使う

配列への挿入



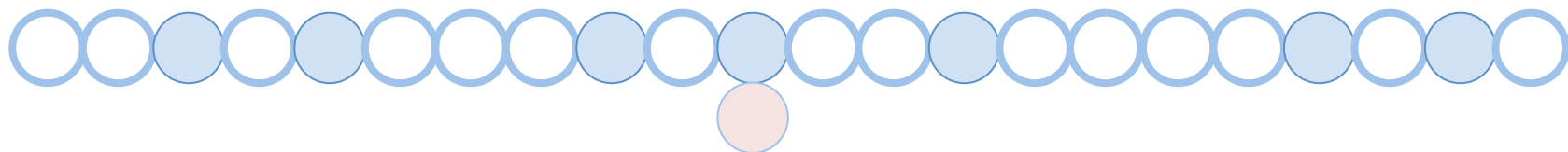
大きく疎らな配列への挿入



挿入する場所が空の場合



挿入する場所が空でない場合: 衝突



辞書まとめ

- キーと値のペアのデータ構造
- キーを元に値を探す
- 辞書の実現には複数の方法がある
 - 配列（ハッシュテーブル）
 - 二分木
 - リスト
- 同義語：
 - 連想配列

ハッシュテーブルとハッシュ関数

ハッシュテーブル

- 辞書を実現する配列
 - 文字列のキーを適当な範囲の整数に変換
 - この変換をする関数をハッシュ関数と呼ぶ
 - 辞書の規模や用途によって種々の作り方がある
- ハッシュ関数
 - 定義域: 可能なキーの文字列がつくる空間
 - 値域: データ数程度の整数値の空間
- 衝突
 - 異なるキーが同じ結果(ハッシュ値)を与える
 - チェイン法
 - リスト構造を使って同じ場所に複数のデータを格納
 - オープンアドレス法
 - 再ハッシュと呼ばれる別の場所を計算する手順を用意

衝突

- データ: “keya” : dataA, “keyb” : dataB
 - “keya”, “keyb” が同じハッシュ値nを与える
 - チェイン法
 - n-1: \rightarrow keyx:dataX
 - n: \rightarrow keya:dataA \rightarrow keyb:dataB
 - n+1: (no data)
 - オープンアドレス法
 - n: keya:dataA
 - r: keyb:dataB [$r \leftarrow (n + k) \bmod (\text{配列の大きさ})$]
 - k は1または素数
 - r で空きが見つかるまで繰り返す
 - 「データが削除されている」という情報が必要

検索・挿入・削除

- 基本的にすべて $O(1)$
- 検索
 - キーからハッシュ関数で直接アクセス可能
 - 衝突時には線形探索の必要あり $O(n)$
- 挿入
 - ハッシュ関数で計算した場所にデータを保存
 - 衝突時
 - リストに追加／挿入
 - 空きが見つかるまで再計算
- 削除
 - データがあれば削除可能
 - 削除の記録が必要な場合がある

ハッシュテーブルの特徴

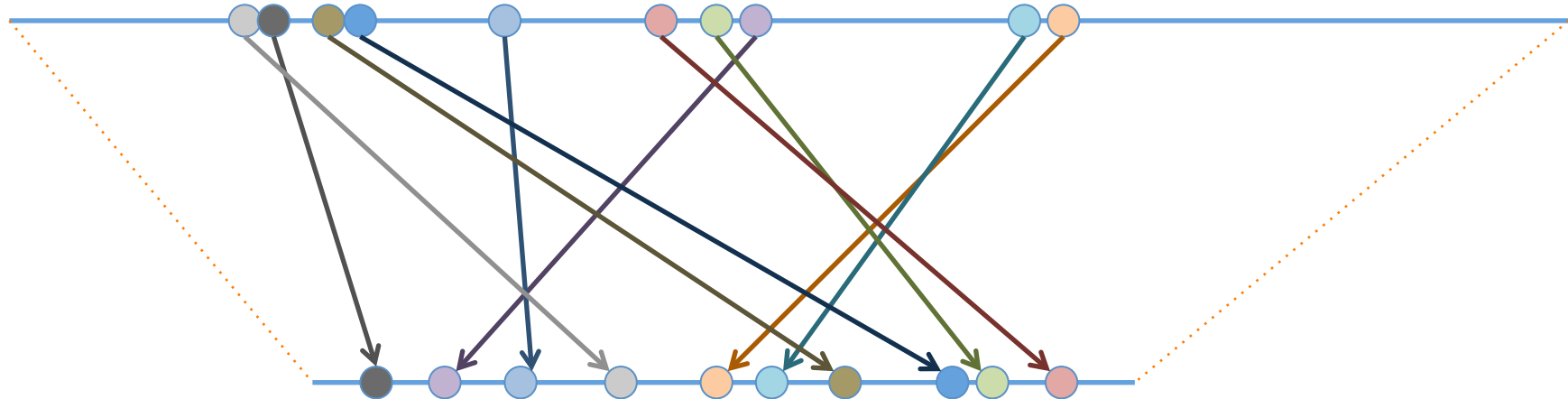
- キーからデータを(探索なしで)検索
 - キーが分かると検索できない
 - 線形探索しか探索の方法がない
- 衝突
 - 異なるキーが同じ位置を与える可能性がある
- キーの間に順序関係
 - 位置とは無関係

キーの分布

- 多くの場合、データが持つキーの分布は一様ではない
 - 例：辞典の項目
 - 英和辞典：S, C, P, ... の順に項目が多い
 - 国語辞典：力行とサ行, ついでア行, タ行, ハ行など
- 単純にキー空間を圧縮すると多数のデータが同じキーを持つことになる
 - キーの分布を(バラバラに)再配置する必要がある

キーの分布と再配置

キーの分布は一様ではない



再配置後にはできるだけ均一に分布させたい

ハッシュ値

- 非負の整数値 (配列の添字に使う)
 - キーから簡単に計算できる
- ハッシュ値からキーを求めることは難しい
 - 一方向性
- 均一に分布
 - キーとハッシュ値が1対1対応が理想
 - 衝突がなく, かつ必要最小限の範囲に収まれば良い

ハッシュ関数

- キーからハッシュ値を計算する関数
 - 具体的な関数形が決まっているわけではない
 - 使用するデータの特徴
 - キーの構造や分布
 - データ数の規模, 変動の有無
 - 時間的, 空間的計算コスト
 - 衝突への配慮
 - 計算要素
 - 剰余
 - ビットシフト
 - 積和演算
 - 排他論理和

計算要素: 剰余

- 剰余

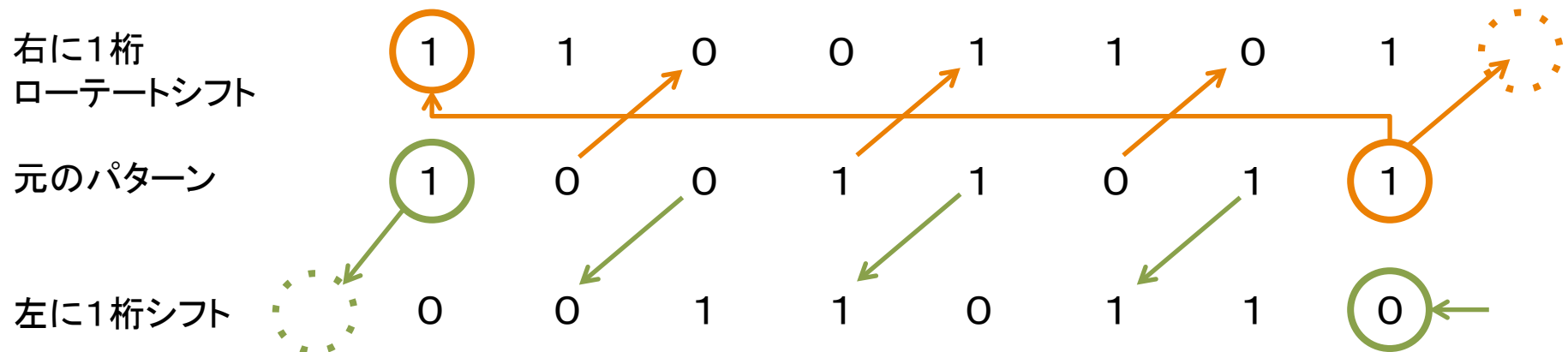
2^n を法とする剰余は2進数表現の下位 n 桁

例えば, 2^4 では:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| | | | | 1 | 0 | 1 | 1 |

計算要素:ビットシフト

- 右または左にビットパターンを移動
 - ロータート: 右端または左端から溢れるビットを反対の端に移動させる



計算要素：排他論理和

- Exclusive OR

| p | q | p XOR q |
|---|---|---------|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

元のパターン

1 0 0 1 1 0 1 1

00000000に対して

1 0 0 1 1 0 1 1

11111111に対して

0 1 1 0 0 1 0 0

例えば

00110110に対して

1 0 1 0 1 1 0 1

さらにもう一度

00110110に対して

1 0 0 1 1 0 1 1

計算要素：積和演算

- $s = w * d + c$
 - s : 変換後のデータ
 - w : 重み
 - d : データ
 - c : 別のデータまたは定数

ハッシュを用いる方法

- キー変換
 - キーの空間よりハッシュ値の空間は小さい
 - 多対一の変換
- 衝突
 - 同じキーに複数の値があり得る
 - キーを再計算
 - 複数の値を保存できる構造を使う
 - 単純な配列ではなくなる

ハッシュ関数の応用

- チェックサム
 - MD5 (Message Digest Algorithm 5)
 - SHA-2, SHA-3 (Secure Hash Algorithm 3)
- 電子署名
- 暗号化
- 疑似乱数

Pythonの辞書

```
dict_salary = {'東京都': '1494000', '神奈川県': '1450000', '大阪府': '1450000', '埼玉県': '1420000', '兵庫県': '1410000', '愛知県': '1403000', '千葉県': '1390000', '広島県': '1389000', '北海道': '1380000', '福岡県': '1350000', '茨城県': '1340000', '岐阜県': '1340000', '愛媛県': '1320000', '滋賀県': '1320000', '福島県': '1320000', '群馬県': '1310000', '宮城県': '1310000', '福井県': '1300000', '徳島県': '1300000', '石川県': '1300000', '富山県': '1300000', '京都府': '1292000', '岡山県': '1290000', '山口県': '1290000', '栃木県': '1290000', '静岡県': '1287000', '香川県': '1285000', '長野県': '1282000', '三重県': '1280000', '島根県': '1280000', '青森県': '1270000', '長崎県': '1260000', '山梨県': '1250000', '沖縄県': '1240000', '鹿児島県': '1240000', '宮崎県': '1240000', '大分県': '1240000', '熊本県': '1240000', '岩手県': '1240000', '新潟県': '1240000', '高知県': '1220000', '奈良県': '1214000', '山形県': '1212000', '秋田県': '1210000', '和歌山県': '1210000', '鳥取県': '1200000', '佐賀県': '1190000' }
```

```
print(dict_salary.keys())  
print(dict_salary.get('東京都'))  
a = dict_salary.keys()  
v = dict_salary.values()
```

Python : 簡単な例 (未完成)

```
a = dict_salary.keys()
v = dict_salary.values()
size = 100
p = 1
mydict = [0]*size
colision = [0]*size
```

```
def myhash(k):
    sum = 0
    for i in range(len(k)):
        sum = sum + ord(k[i])
    return (sum % size)
```

```
for k in a:
    index = myhash(k)
    while True:
        if mydict[index] == 0:
            mydict[index] =
dict_salary.get(k)
            break
        else:
            colision[index] = 2
            index = (index + p) % size

for i in range(size):
    print(mydict[i])
```

コンピュータ(CPU)の仕事

- 入力データを読み取って，処理，出力を書き出す
- 記号を受け取り，記号を書き出す
- 入力記号を，出力記号に変換
 - 英語を受け取り，日本語を出力
 - 平文を受け取り，読めない記号列を出力
 - ファイルを読み取り，サイズ小さなファイルを出力
 - 記号列を読み取り，改変を検査・訂正できる記号列を出力

宿題: ex06

- 以下のファイルについてmd5 と sha256 のハッシュ値(チェックサム)を求め報告してください
 1. Oh-o!Meiji クラスウェブ 第1回授業の講義資料(pdf)ファイル
 2. これまでに宿題で提出したプログラム(プレーンテキスト)
 - 1文字程度の変更の前後を比較する
 1. 改行・スペースを加える／取る
 2. コメントなどの修正, 誤字の訂正, など
- 以上についての感想をまとめる

ツール

- Mac OS X
 - ターミナル
 - md5, shasum
- Linux (Ubuntu)
 - ターミナル(端末)
 - md5sum, shasum, sha256sum
- Windows
 - PowerShell (v4以降)
 - Get-FileHash
 - それ以前
 - FCIV: md5 のみ, sha256 は未対応
 - その他, フリーウェア
 - MD5&SHA Checksum Utility: http://download.cnet.com/MD5-SHA-Checksum_Utility/3000-2092_4-10911445.html

ツール

Mac OS X

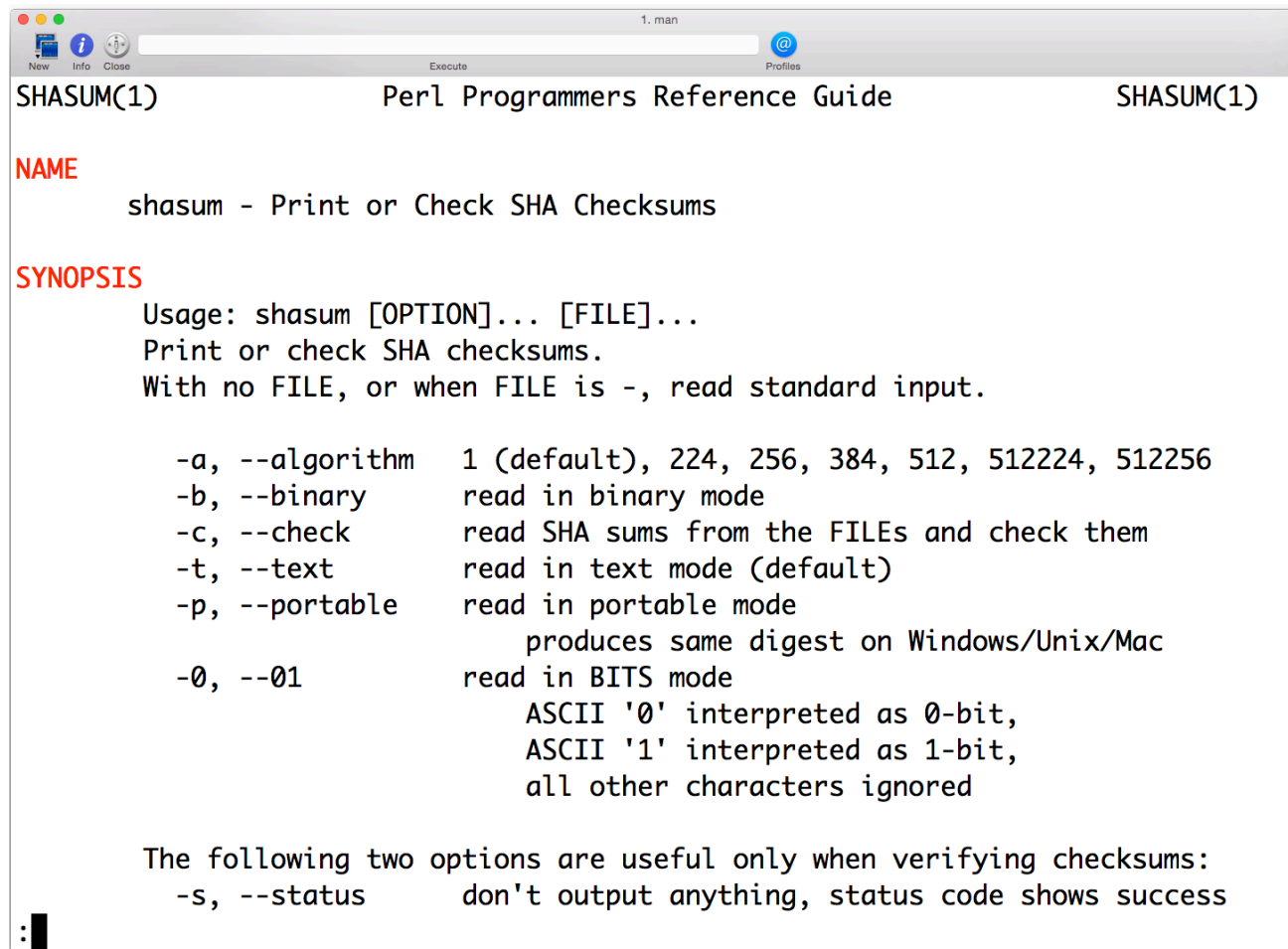


```
yuzu:~ wenren$ which md5
/sbin/md5
yuzu:~ wenren$ which shasum
/usr/bin/shasum
yuzu:~ wenren$
```

Windows

A screenshot of the Download.com website for the 'MD5 & SHA Checksum Utility'. The page features a green 'Get It Here' button at the top. Below it, there's a section for 'Quick Specs' with details like Version 2.1, 311,679 total downloads, and a price of 'Free'. The 'Editors' Review' section includes a byline from Download.com staff and a detailed description of the utility's cryptographic functions. The 'Publisher's Description' section mentions it's a freeware tool from Raymond's Personal Software. The page also includes social media links, a star rating, and several advertisements for local classifieds, utility amplifiers, and an Epson printer.

man shasum



```
SHASUM(1)                                Perl Programmers Reference Guide                                SHASUM(1)

NAME
    shasum - Print or Check SHA Checksums

SYNOPSIS
    Usage: shasum [OPTION]... [FILE]...
    Print or check SHA checksums.
    With no FILE, or when FILE is -, read standard input.

    -a, --algorithm 1 (default), 224, 256, 384, 512, 512224, 512256
    -b, --binary    read in binary mode
    -c, --check      read SHA sums from the FILEs and check them
    -t, --text       read in text mode (default)
    -p, --portable   read in portable mode
                     produces same digest on Windows/Unix/Mac
    -0, --01         read in BITS mode
                     ASCII '0' interpreted as 0-bit,
                     ASCII '1' interpreted as 1-bit,
                     all other characters ignored

    The following two options are useful only when verifying checksums:
    -s, --status      don't output anything, status code shows success
```

補足

- md5やshaはデジタル署名や認証などセキュリティのために使用されています
- 十分なセキュリティが確保できなくなった技術は、より安全な技術に移行するために使われなくなります
 - md5 や sha-1 に代わって sha-2 (sha224, sha256, sha384, sha512, etc.) が使われるようになりつつあります

提出についての注意

- 今回はレポートの提出です
 - Microsoft Word形式のファイルで提出
 - Pages からは ファイル＞書き出す＞Word
 - リッチテキストフォーマット(rtf)ファイルでもよい
 - 冒頭に氏名、学科、学年、クラス、番号等を記すこと
 - ハッシュ値の計算に使用したProcessingのプログラムの番号 (eg. ex01, ex02, ex03, ..., ex05) を記すこと
 - どのような変更かも知れずに
- Oh-o!Meijiから提出(次回の授業開始までに)

予告

- 次回は後半に簡単なクイズを行います。
 - 最終回の試験の予行演習

連絡先

樋口文人

wenren@meiji.ac.jp