電子計算機入門

池田, 大輔
九州大学附属図書館

電子計算機入門
第7回(06/16)

池田 大輔

z4id01in@cse.ec.kyushu-u.ac.jp

附属図書館

目次

- **前回課題の回答例**
- **辞書** (dictionary)
- **辞書のソート**
- **内包表記**
- **第 2 回レポート問題**

前回課題の解答例

- 入力として文字列を一つ与えたとき、この中に現われるすべての部分文字列とその頻度を出力せよ。出力の順番は頻度の高い順とする。

前回課題の解答例

- 入力として文字列を一つ与えたとき、この中に現われるすべての部分文字列とその頻度を出力せよ。出力の順番は頻度の高い順とする。

```
import sys
str = sys.argv[1]
res = [] # (n, substr) の形式で結果を格納する配列
for b in range(len(str)): # 2 重 for ループで全ての部分文字列を生成
    for e in range(b+1, len(str)+1): # str[b:e] が新たな一台
        # 以下の while 文で既出かどうかチェック
        i = 0
        while i < len(res) :
            (n, substr) = res[i]
            if substr == str[b:e]:
                res[i] = (n+1, substr)
                break
            i = i + 1
```

前回課題の解答例(Cont.)

```
        else: # while とそろえる
      # while 文中で break が呼ばれなかった時
          res.append((1, str[b:e]))
for i in res: # 結果を出力
    print i
```

多次元配列のソート

- 各要素が複数の要素からなる

  例えば`("a", 4)`が一要素
- 多次元配列のソートは、各要素の最初の要素で並べかえられる

多次元配列のソート

- 各要素が複数の要素からなる

  例えば（"a", 4）が一要素

- 多次元配列のソートは、各要素の最初の要素で並べかえられる

  →並べかえたい基準となる情報を最初に置く

多次元配列のソート

- 各要素が複数の要素からなる

  例えば`("a", 4)`が一要素

- 多次元配列のソートは、各要素の最初の要素で並べかえられる

  →並べかえたい基準となる情報を最初に置く

  例えば`(4, "a")`

実習：多次元配列のソート

- 以下のプログラムを作りを実行せよ

```
array = [("ab", 2), ("a", 3), ("bb", 4)]
array.sort()
print array
array = [(2, "ab"), (3, "a"), (4, "bb")]
array.sort()
print array
```

配列検索のイメージ

- 新しい一台を見たら、台数を記録している配列 (*cars* とおく) を検索して、その一台がすでに記録済みかどうか調べなければならない
  - さきほどの解答例では `while` 文がこれに相当

配列検索のイメージ

- 新しい一台を見たら、台数を記録している配列 (*cars* とおく) を検索して、その一台がすでに記録済みかどうか調べなければならない
  - さきほどの解答例では `while` 文がこれに相当
- チェックするには、左から順に一台一台見ていく必要あり

cars = `[(8, '`カローラ`'), (4, 'Fit'), ...]`

配列検索のイメージ

- 新しい一台を見たら、台数を記録している配列 (*cars* とおく) を検索して、その一台がすでに記録済みかどうか調べなければならない
  - さきほどの解答例では `while` 文がこれに相当
- チェックするには、左から順に一台一台見ていく必要あり

cars = `[(8, '`カローラ`'), (4, 'Fit'), ...]`

→台数が増えてくると、チェックに時間がかかる

車の位置を覚えていれば、もっと速くデータにアクセスできるはず

辞書(dictionary)

- ’物’と値の対応表
  - ハッシュ、連想配列などとも呼ばれる
  - 「記録ずみかどうか」が一瞬で分かる
- キーで添字づけした複数のデータ
- キーは文字列や数字など、変更不能であれば何でもよい
  - 配列は変更可能なので、キーにはできない
  - 配列のキーは、0 以上の連続な自然数

辞書 (Cont.)

- **辞書の作り方**

```
d = {} # 空の辞書。普通の配列は []
d = {'jack':4098, 'sjoerd':4127} # キー:値
```

- **キーが "key" である値へのアクセス**

```
d[key] = 1
```

- **あるキー "v" をもつ値が存在するかどうか**

```
d.has_key(v)
```

- **キーの一覧を配列として返す**

```
d.keys()
```

実習：辞書

- 以下のプログラムを作成しなさい

```
d = {'jack':4098, 'sjoerd':4127}
d['ken'] = {4048} # 値を追加
print d # 全体を出力
if d.has_key(tom): # あるキーがあれば
    print d['tom'] # 対応する値を出力
elif d.has_key(jack):
    print d['jack']

for key in d.keys(): # 全キーに対し
    print key, d[key]
```

辞書による頻度カウント

```
d = {} # 結果を格納する空辞書
# 新たな'物'vがきた場合
if d.has_key(v): # vがキーとして存在するか？
    d[v] += 1 # vの値を増やす
else:
    d[v] = 1 # 始めての時は初期値(1)
```

実習：頻度カウントと辞書

- 辞書を用いて、すべての部分文字列の頻度を数えるプログラムを作りなさい
  - 前回の課題の回答例を参考にして、`while` 部分付近を変更する

実習：頻度カウントと辞書

- 辞書を用いて、すべての部分文字列の頻度を数えるプログラムを作りなさい
  - 前回の課題の回答例を参考にして、`while` 部分付近を変更する

```
substr = str[b:e]
if res.has_key(substr):
    res[substr] = res[substr] + 1
else:
    res[substr] = 1
```

配列 v.s. 辞書

- 配列が便利な場合
  - 自然な番号づけがある
  - この番号 (のみ) でデータにアクセスする
  - 番号が連続している (連続してアクセスする)

  例えば、すべての学生に関するデータ
  - 学籍番号があり、番号の抜けもない
- 逆に、
  - どのデータへアクセスするかわからない
  - 頻繁にいろんなデータへアクセスする

  場合は辞書がよい

辞書のソート

- 辞書は `sort()` 関数を持たない

  `d.sort()` とはできない

- 配列に直してから、ソートする

  `d` を適当な辞書変数とする

  - ソートする基準となるものを最初にして、配列を作成する場合

```
array = [] # ソート用の配列
for key in d.keys():
    array.append((d[key], key))
array.sort()
```

辞書のソート

- 辞書は `sort()` 関数を持たない

  `d.sort()` とはできない

- 配列に直してから、ソートする

  `d` を適当な辞書変数とする

  - ソートする基準となるものを最初にして、配列を作成する場合

```
array = [] # ソート用の配列
for key in d.keys():
    array.append((d[key], key))
array.sort()
```

- ソートはよく使う処理なのに、長くて面倒...

内包表記

- 配列 (リスト) から別の配列を簡単に作る
  - 条件を満たす要素をとりだす
  - 要素を加工する

内包表記

- 配列 (リスト) から別の配列を簡単に作る
  - 条件を満たす要素をとりだす
  - 要素を加工する
- 例：辞書 d のキーが車名、値が台数のとき (台数、車名) の配列を作る

```
array=[ (d[key], key) for key in d.keys()]
```

  - さきほどの 3 行が 1 行でかけた

内包表記

- 配列 (リスト) から別の配列を簡単に作る
  - 条件を満たす要素をとりだす
  - 要素を加工する
- 例：辞書 d のキーが車名、値が台数のとき (台数、車名) の配列を作る

```
array=[ (d[key], key) for key in d.keys()]
```

  - さきほどの 3 行が 1 行でかけた

  数式の集合表現と同じ

$$array = \{(key, d[key]) \mid key \in d.keys()\}$$

今日の課題

- 入力として文字列を一つ与えたとき、この中に現われるすべての部分文字列とその頻度を出力するプログラムを**辞書を用いて**作成せよ。ただし、出力の順番は頻度の高い順とする。

第2回レポート課題

- 以下を行なうプログラムを作成しなさい
  - 入力：ファイル名と正の整数 n
  - 問題：長さ n の部分文字列の出現頻度を数え、頻度の高い部分文字列 10 個を

    ```
    aaaa : 2682
    bbbb : 2429
    ....
    ```

    の書式で出力する
- 締切：7/6(火)
- プログラムと一緒に、サンプルファイル (授業の Web ページに用意) と $n = 3$ に対する実行結果を添えること
  - サンプルファイルの名前は “large.txt” である

    よって、実行結果を採取する時、Arguments には “large.txt　3” と入力することになる

第2回レポート課題(Cont.)

- その他の条件
  - プログラムには辞書を用いること
  - プログラム中にそのプログラムの実行時間を計測する文 (付録参照) を入れ、実行結果には実行時間が出力されるようにすること
  - 複数の行にまたがる部分文字列は数えなくてよい
  - “large.txt” は非常に大きいので、小さめのサンプルファイル “small.txt”(同じく授業の Web に用意) を用いて実行結果を確認するとよい
- 加点項目
  - 長さ n ではなく n 以上の部分文字列とする
  - さらに、頻度が同じ場合は、長い方を上位とする

付録：全辞書データへアクセス

- *dict* を辞書変数とする

  dict.keys() 以外にも、以下の関数で辞書中の全要素にアクセス可能

- 値でアクセス

```
for val in dict.values():
    print val # この場合、キーは不明
```

- キーと値でアクセス

```
for (key, val) in dict.items():
    print key, val
```

付録：内包表記の例

- 例：文字列要素の配列から (長さ、文字列) 配列を作る

```
array=["baa","accc","abc","bab"]#入力
new = [(len(i), i) for i in array]
→ array = [(3, "baa"), (4, "accc"), ...
```

- 例：文字列要素の配列から長さが 4 以上の要素のみを抽出

```
array=["baa","accc","abc","bab"]#入力
new = [i for i in array if len(i) >= 4]
→ array = ["accc"]
```

- 例：複数の配列から

```
xs = (1, 2, 3, 4, 5)
ys = (9, 8, 7, 6, 5)
z = [(x, y) for x in xs for y in ys if
x*y > 25]
```

付録：時間の計測について

- ファイルの先頭に

```
import time
start = time.time()
```

を入れ、さらに、全部の処理が終ったあとに

```
end = time.time()
print end - start
```

を入れる