情報システム 2 学期

2024 年 9 月 11 日

1. データ構造とアルゴリズム

1.1. データの、型と列データ

1.1.1. 基本的なデータ型

1.1.1.1. 論理型

否定		論理積			論理和		
X	NOT X	X	Y	X AND Y	X	Y	X OR Y
偽	真	偽	偽	偽	偽	偽	偽
真	偽	偽	真	偽	真	偽	真
		真	偽	偽	真	偽	真
		真	真	真	真	真	真

1.1.1.2. 文字列型

Unicode
…文字をコンピュータで扱うための標準規格。
　世界で使われる多くの文字を扱えるように策定されている Unicode が、文字集合（文字コード）として広く採用されている。

1.1.1.3. 数値型

・実数型は、 **浮動小数点数** で表現される。
・2 進法で限られたビット数によって表現するため、表現できる数値の範囲が限られる。

1.1.2. 文字列

文字列
...文字を一列に並べたもの。
　文字列は、文字の配列として扱われることが多い。

C	o	m	p	u	t	e	r
0	1	2	3	4	5	6	7

文字列の長さ
...文字列の長さは、文字列に含まれる文字の数で表現される。
　例えば、“Computer” は 8 文字の長さを持つ。
　このとき、先頭の文字の添字は 0 であり、これを 0-origin という。

```
>>> s = 'Computer'
>>> s
'Computer'

>>> s[2]
'm'

>>> type('AAA')
<class 'str'>
```

1.1.2.1. 2 つの文字列の連結

```
>>> 'ABC' + 'DEF'
'ABCDEF'
```

1.1.2.2. 2 つの文字列の比較

```
>>> 'ABC' < 'ABE' # 小さい方が先
True

>>> 'AB' < 'ABA'  # 長い方が大きい
True
```

1.1.2.3. 部分文字列の取得

```
>>> 'ABCDEF'[2:4]
'CD'
```

```
>>> s = 'ABCDEF'
>>> s[2:4]
'CD'
```

1.1.3. 列データ構造

データ構造とはデータの集まりをコンピュータの中で効果的に扱うための形式のこと。

1.1.3.1. リスト

・リストは、複数のデータをまとめて扱うためのデータ構造。
・リストの要素は、インデックスを使ってアクセスできる。

1.1.3.2. 配列

・配列は、同じ型のデータを連続したメモリ領域に格納するデータ構造。
・配列の要素は、インデックスを使ってアクセスできる。

1.1.3.3. スタック

・スタックは、データを一時的に保存するためのデータ構造。
・データの追加や削除は、常に一方向から行われる。

```
>>> data = [4, 1, 7, 3, 2]
>>> data.append(8)

>>> data
[4, 1, 7, 3, 2, 8]

>>> data[-1]
8

>>> data.pop()
8

>>> data
[4, 1, 7, 3, 2]
```

1.1.3.4. キュー

・キューは、データを一時的に保存するためのデータ構造。
・データの追加は一方向から行い、削除は反対方向から行う。

```
>>> from collections import deque
>>> data = deque([4, 1, 7, 3, 2])
```

```
>>> data
deque([4, 1, 7, 3, 2])

>>> data.popleft()
4

>>> data
deque([1, 7, 3, 2])

>>> data.append(5)
>>> data
deque([1, 7, 3, 2, 5])

>>> data.appendleft(6)
>>> data
deque([6, 1, 7, 3, 2, 5])
```

1.1.4. レコード型と連想配列

1.1.4.1. レコード型

・レコード型は、複数のデータをまとめて扱うためのデータ構造。
・レコード型の要素は、フィールドと呼ばれる。

`フィールド名：値` の形式でデータを表現する。

1.1.4.2. 連想配列

・連想配列は、キーと値のペアを格納するデータ構造。
・キーを使って値にアクセスできる。

`キー：値` の形式でデータを表現する。

P.62 問 6

図書館で貸し出す本をレコード型で合わすとすると、どのようなフィールドが昼用になるか考えてみよう。図書館を利用する人の情報をレコード型で表すとすると、どのようなフィールドが必要になるか考えてみよう。

貸し出す本

フィールド名	型
タイトル	文字列
著者	文字列

利用者

フィールド名	型
氏名	文字列
住所	文字列

出版社	文字列
出版年	整数
貸出状況	論理型

電話番号	文字列
貸出中	リスト

1.2. 問合せと木構造

1.2.1. 問合せ

・正規表現

表現	意味
`a`	文字 a
`ab`	文字 a に続く文字 b
`a|b`	文字 a または文字 b
`^a`	文字 a で始まる
`a$`	文字 a で終わる
`^a$`	文字 a がふくまれる
`a*`	文字 a の 0 回以上の繰り返し
`a+`	文字 a の 1 回以上の繰り返し
`a[bc]`	文字 a の後に b または c
`[a-z]`	アルファベット小文字
`[A-Z]`	アルファベット大文字
`[0-9]`	数字
`a?`	文字 a の 0 回または 1 回の繰り返し
`a{n}`	文字 a の n 回の繰り返し
`a{n,}`	文字 a の n 回以上の繰り返し
`a{n,m}`	文字 a の n 回以上 m 回以下の繰り返し

1.2.2. 集計

・条件に合致した件数、合計、最大、最小など

繰り返しを用いて条件にあった件数を数える Python プログラムの例

```
>>> data = [4, 1, 7, 3, 2]
>>> kensuu = 0
>>> for i in range(0, 5):
...     if (data[i] % 2 == 0):
...         kensuu += 1
...
>>> print(kensuu) # 2
```

繰り返しを用いて条件にあった値の最大値を求める Python プログラムの例

```
>>> data = [4, 1, 7, 3, 2]
>>> max_value = 0
>>> for i in range(0, 5):
...     if (data[i] % 2 == 0 and data[i] > max_value):
...         max_value = data[i]
...
>>> print(max_value) # 4
```

※ Python には、条件に合致した件数を数える関数や、条件に合致した値の最大値を求める関数が用意されている。

関数	意味
len(data)	データの長さ
data.count(x)	データ中の x の個数
max(data)	データ中の最大値
min(data)	データ中の最小値
sum(data)	データの合計

1.2.3. 木構造

・木構造は、階層構造を表現するためのデータ構造。

・木構造は、根（ルート）となるノードから始まり、枝分かれするノード（内部ノード）と、枝の先端にあるノード（葉ノード）から構成される。

1.3. アルゴリズム

1.3.1. アルゴリズムと計算効率

マージソート

・マージソートは、分割統治法を用いたアルゴリズムの一つ。
・データを分割し、それぞれをソートしてから結合することで、全体をソートする。