4-2 アルゴリズム基礎
4-3 データ構造とプログラミング基礎

東京大学　数理・情報教育研究センター

2020年5月11日

概要

- アルゴリズム基礎
  - フローチャートによるアルゴリズムの可視化やアルゴリズムの基本構造である順次構造，選択構造，反復構造などについて学びます．また，具体例として探索と整列の代表的なアルゴリズムを学びます．
- データ構造とプログラミング基礎
  - コンピュータでデータを扱うための数値や文字の表現について学びます．また，アルゴリズムをコンピュータで実行するためのプログラムの基礎について学びます．

本教材の目次

デジタル表現

アナログとデジタル

- 連続的に変化する量を一定の間隔で区切った離散的な量で表すことをデジタルと呼びます.
- 一方，連続的な量で表すことをアナログと呼びます.

- 例えば，デジタル時計では通常，時/分/秒をそれぞれ1時間，1分，1秒の離散的な数で表しますが，アナログ時計では時間を連続的に変化する針の位置で表します.

デジタル時計 

アナログ時計 

変換誤差

- アナログ情報をデジタル情報に変換する時，元の情報の一部が失われることがあります．この失われた情報を変換誤差と呼びます.

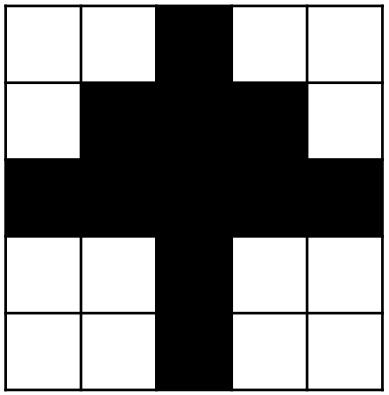
- 例えば，最小単位が0.1秒のデジタル時計では9.54秒は9.5秒と表され，0.04秒が失われた変換誤差となります.

デジタル表現

デジタル表現

- コンピュータでは，情報（数値，文字，画像，音声など）を0と1の離散的な値の組み合わせで表現します．

- 例えば，白黒画像では画像の各ピクセルの白の部分を0，黒の部分を1と表現することができます．



0	0	1	0	0
0	1	1	1	0
1	1	1	1	1
0	0	1	0	0
0	0	1	0	0

２進数

- 0と1の組み合わせで数を表現することを二進法と呼び，二進法で表された数値を2進数と呼びます．

情報量と単位

情報量と単位

- 2進数の1桁をビットと呼び，ビットは情報量の最小単位となります．1ビットでは2 (=2^1)通りの状態を表すことができます．

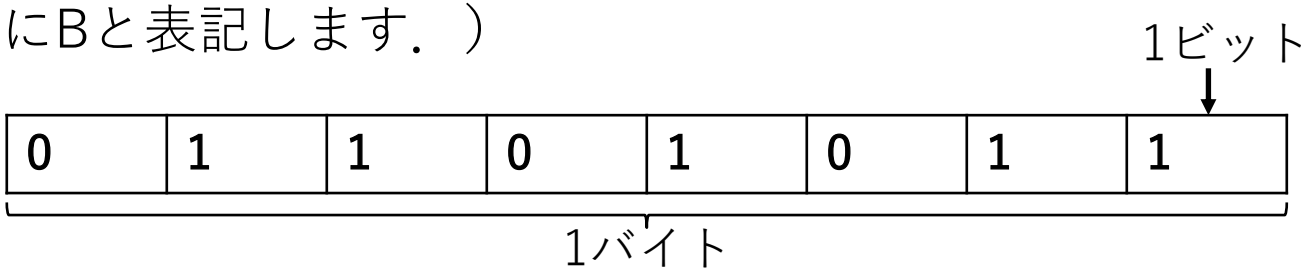
0	白
1	黒

1ビットでの情報表現例

0	0	白
0	1	赤
1	0	緑
1	1	青

2ビットでの情報表現例

- 8ビットをまとめたものは1バイトという情報量の単位となります．1バイトでは256(=2^8)通りの状態を表すことができます．（バイトはByteあるいは単にBと表記します．）



情報量と単位

情報量と単位

- 1バイトの1024（$=2^{10}$)倍の情報量の単位は1キロバイト（KB），1キロバイトの1024（$=2^{10}$)倍の情報量の単位は1メガバイト（MB），のように1バイトより大きな情報量の単位は1024（$=2^{10}$)倍ごとに単位が付けられます.

単位	読み方	関係
bit	ビット	
B	バイト	1B = 8bit
KB	キロバイト	1KB = 1024B
MB	メガバイト	1MB = 1024KB
GB	ギガバイト	1GB = 1024MB
TB	テラバイト	1TB = 1024GB
PB	ペタバイト	1PB = 1024TB
EB	エクサバイト	1EB = 1024PB

数値の表現

10進数と16進数

- 0から9の10種類の数字で表された数を10進数と呼びます．10進数は普段日常で使用される数値の表現です．

- 2進数は0と1の2種類の数字のみで表された数です．

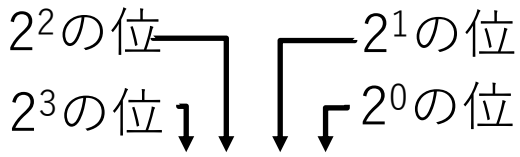
- この他に，0から9の10種類の数字とAからFの6種類の文字，計16種類の数字と文字で表された数である16進数もあります．16進数は2進数の下の桁から4桁ずつ区切り，それぞれを16種類の数字・文字で表したものになります．

10進数	2進数	16進数
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	10000	10

数値の表現

2進数から10進数への変換

- 2進数は以下のように各桁が10進数の$2^0(=1)$, $2^1(=2)$, $2^2(=4)$, $2^3(=8)$に対応しています.



$(1\ 1\ 0\ 1)_2$　　$(\cdots)_2$で2進数であることを表します

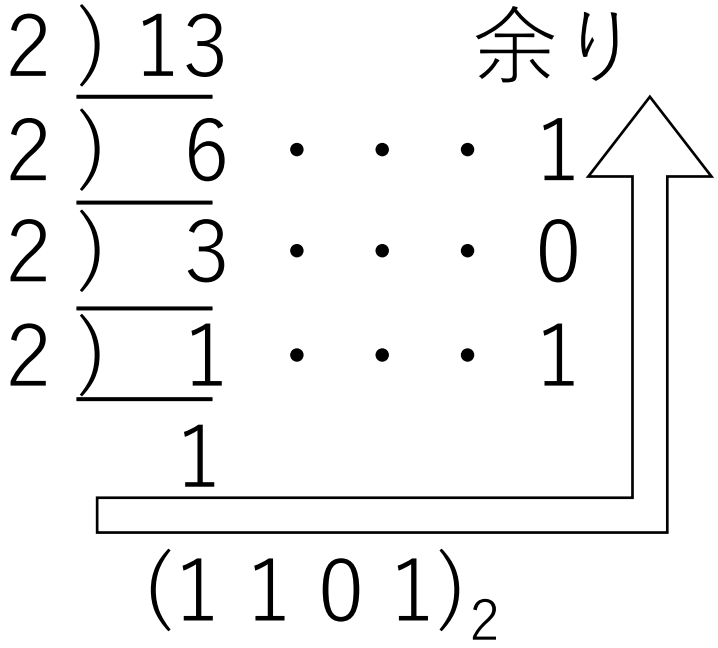
- 例えば2進数の1101は以下のように10進数の13に変換することができます.

$$1\times\ 2^3+\ 1\times\ 2^2+\ 0\times\ 2^1+\ 1\times\ 2^0=8+4+0+1=13$$

数値の表現

10進数から2進数への変換

- 以下のように10進数を商が1になるまで2で割り続けたときの，商とすべての余りを並べると2進数に変換することができます.



誤差

- コンピュータの数値表現で精度を保つために必要な数字の桁数を有効桁数と呼びます.
- コンピュータで計算した値と実際の値との差を誤差と呼びます.
- 誤差には，情報落ち，桁落ち，丸め誤差，打ち切り誤差，などがあります.
  - 情報落ち：有効桁数が限られている場合，絶対値が大きい数と小さい数を加減算で，小さい数は計算結果に含まれなくなり誤差が生じることがあります.
  - 桁落ち：大きさが近い数同士の引き算で，有効桁数が減ってしまい誤差が生じることがあります.
  - 打ち切り誤差：くり返しの計算で求められる値の計算を途中で打ち切って近似値を求めた場合，誤差が生じます.
  - 丸め誤差：数値を表現する際，有効桁数やデータ型の制約により桁数が制限された場合，誤差が生じます.

文字の表現

文字コード

- 2進数または16進数で表された数と文字・記号の対応を表したものを文字コードと呼びます.

- 一般にアルファベット・数字・記号は1バイト（2進数では8桁，16進数では2桁）で表し，漢字などは2バイト（2進数では16桁，16進数では4桁）以上で表します.

- 1バイトの文字コードとしてJISコードがあります.

- 文字コードにはこの他にもシフトJIS，EUC，Unicodeなどがあります.

文字の表現

文字コード

- JISコードでは文字・記号は以下のように16進数の数と対応付けられています.

https://www.jagat.or.jp/past_archives/content/view/3852.html

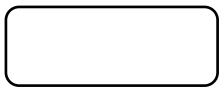
例えば「Apple」を16進数で表すと左のJISコードより

41 70 70 6C 65

となります.

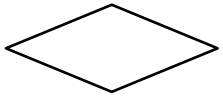
アルゴリズムとフローチャート

- 問題を解決するための手順を定式化したものをアルゴリズムと呼びます．また，アルゴリズムを視覚的にわかりやすく表したものをフローチャートと呼びます．
  （フローチャートをUMLで記述したものをアクティビティ図と呼ぶ）
- フローチャートは以下の記号の組み合わせとして表現することができます．

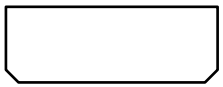

端子：フローチャートの開始と終了


データ：データの入出力


処理


判断：条件による分岐


ループ端（始端）：ループの始まり

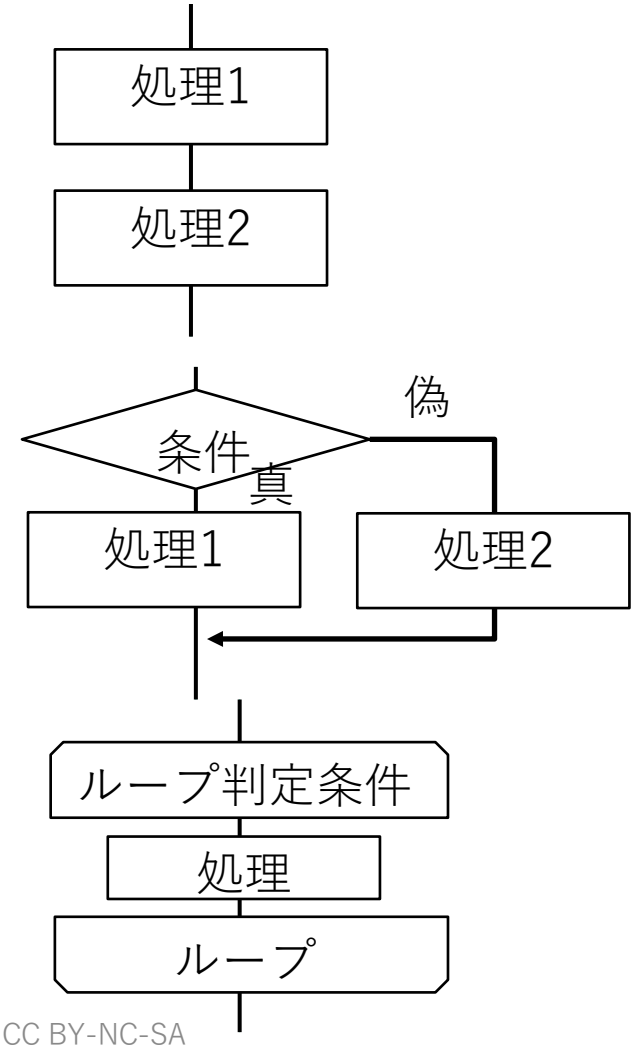

ループ端（終端）：ループの終わり

- フローチャートは一般に上から下へ，左から右へ記述し，記号の中には必要な処理を記述します．また，処理の流れを明示したい場合は矢印をつけます．

アルゴリズムの基本構造

アルゴリズムは順次，選択（分岐），反復（繰り返し）の3つの基本構造で表すことができます．

- 順次構造
  - 処理を順に行う構造
  - 処理1と2を順番に実行
- 選択構造
  - 条件により処理が分岐する構造
  - 条件が真の時は処理1，偽の時は処理2を実行
- 反復構造
  - 判定条件が満たされている間に一連の処理を行う構造
  - ループ条件を満たしている間，処理を繰り返す



プログラム

- 処理の手順であるアルゴリズムをコンピュータが実行可能なようにコンピュータが理解できる言語で記述したものをプログラムと呼びます.
- プログラムを記述するための言語をプログラミング言語と呼び，プログラミング言語でプログラムを作成することをプログラミングと呼びます.
- プログラム言語には，コンピュータが実行可能な命令を2進数で表した機械語，機械語の命令を英字で表したアセンブリ言語などがあります．これらの言語は低級言語とも呼ばれます.

プログラム

- プログラム言語を人間にとってわかりやすく表した言語は高級言語と呼ばれます.
  - 高級言語にはインタプリタ方式とコンパイラ方式があります.
    - プログラムを命令ごとに機械語に変換して実行するインタプリタ方式
      - JavaScript, Python, Rなど
    - プログラムをすべて機械語に変換してから実行するコンパイラ方式
      - C, C++, Javaなど

変数と代入

- アルゴリズムでは変数を利用して処理を行います.
- 変数はデータなどを格納する箱にあたるもので，実際はコンピュータのメモリ上に確保される領域となります.
- 変数にデータとなる値を設定することで，後から再利用することができます.
- 実際に変数に値を設定することを代入と呼びます．多くのプログラム言語では，代入「=」記号（代入記号）を使って記述します.

変数と代入

- 以下ではPythonのプログラムで=の左側にある変数（xやy）に=の右側にある値を代入している例です．変数（z）に変数の値（x + y）を代入することもできます

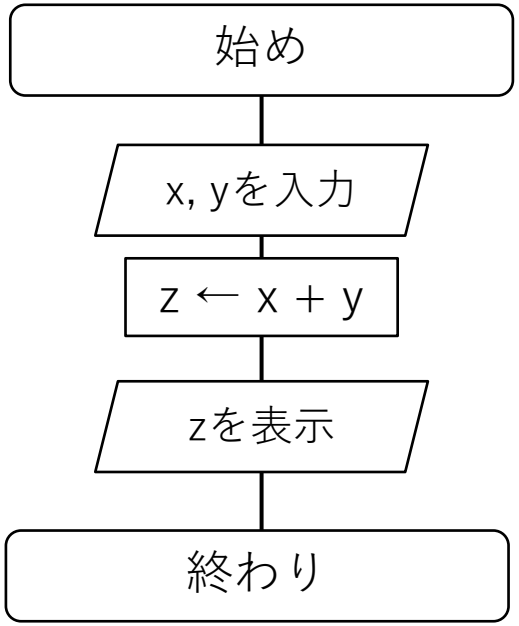
```
x = 1
y = 2
z = x + y
```

- 一般に変数は，整数，実数，文字列など型をあらかじめ指定する必要がありますが，Pythonでは変数に値を代入した際に型が決定されます．

順次構造

変数x，変数yそれぞれに数値を代入し，それらの和を変数zに代入して表示する処理は以下の順次構造で表すことができます．

フローチャート



Pythonプログラムと実行結果

```
x = 1 # 変数xに数値を代入
y = 2 # 変数yに数値を代入
z = x + y # 変数zにxとyの和を代入
print(z) # 変数zの値を表示
```

```
3
```

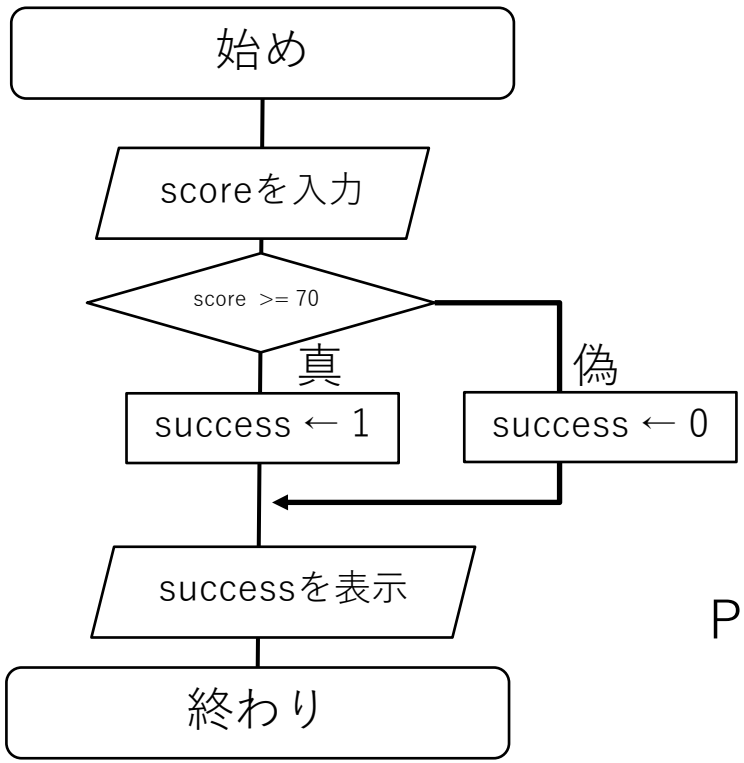
Pythonプログラミング入門：数値演算
https://utokyo-ipp.github.io/1/1-1.html

Pythonプログラミング入門：変数と関数の基礎
https://utokyo-ipp.github.io/1/1-2.html

選択構造

変数scoreの値が70以上の時は変数successに1，70未満の時は変数successに0を代入して表示する処理は以下の選択構造で表すことができます.

フローチャート



Pythonプログラムと実行結果

```
score = 75
if score >= 70:
    success = 1
else:
    success = 0
print(success)
```

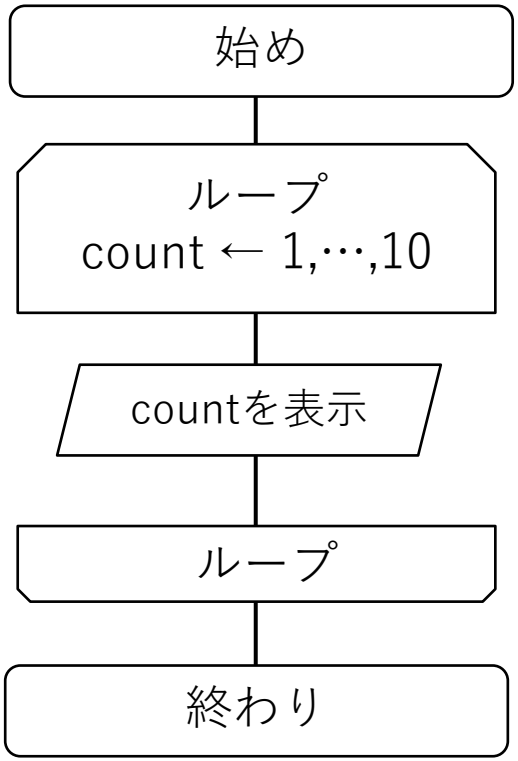
```
1
```

Pythonプログラミング入門：条件分岐
https://utokyo-ipp.github.io/2/2-3.html

反復構造（1）

変数countに1から10を代入して表示する処理は以下の反復構造で表すことができます.

フローチャート



countを1から10まで1ずつ増やしながら繰り返す

Pythonプログラムと実行結果

```
for count in range(1, 11):
    print(count)
```

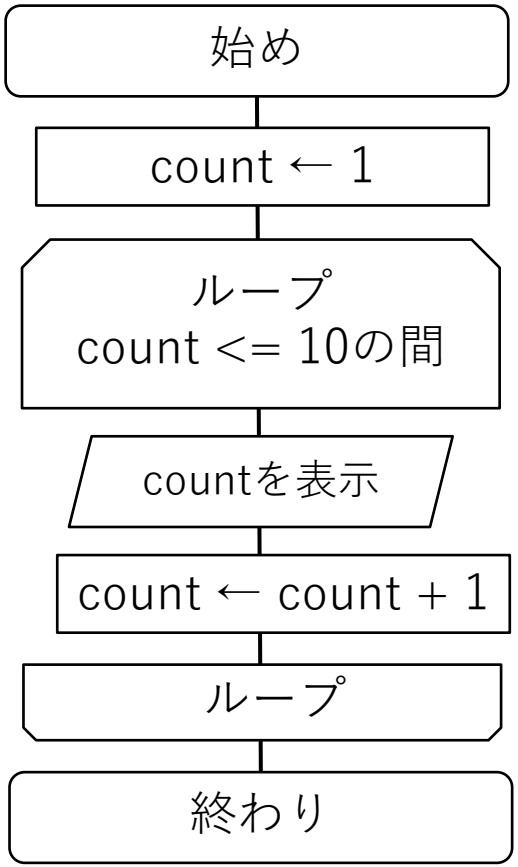
```
1
2
3
4
5
6
7
8
9
10
```

Pythonプログラミング入門：繰り返し
https://utokyo-ipp.github.io/3/3-2.html

反復構造（２）

変数countに1から10を代入して表示する処理は以下の反復構造で表すこともできます.

フローチャート



countが10以下の間
繰り返す

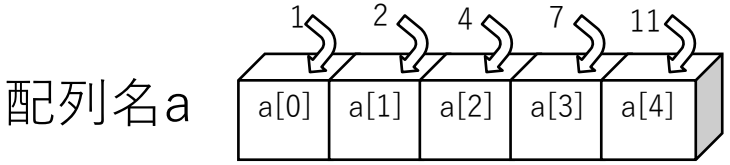
Pythonプログラムと実行結果

```
count = 1
while count <= 10:
    print(count)
    count = count + 1
```

```
1
2
3
4
5
6
7
8
9
10
```

配列

- 配列はデータ構造の1つで，配列により複数のデータをまとめて扱うことができます．
  - 配列では値を入れる箱が用意されており，それぞれの箱を配列の要素と呼びます．
  - すべての箱は同じ変数名（配列名）を持ち，それぞれの箱は異なる番号で区別されます．この番号を添字と呼びます．
    - Pythonでは配列の添字は「0」から始まります．
  - 配列を構成する要素は，配列名に添字をつけることで指定することができます．
    - 先頭の要素は，配列名[0]，のように指定できます．
  - 添字により配列の要素指定することで，要素への値の代入や参照ができます．



例：5つの数値データ[1,2,4,7,11]を配列aに格納

配列

- 配列は複数のデータを配列名と添字でまとめて扱うことができるので複数のデータを効率よく処理するのに使われます.

- Pythonでは配列と同じ役割を持つデータ構造としてリストがあります.
  - 配列では同じ型（数値や文字列）のデータのみ格納できますが，リストでは異なる型を混在して格納することができます.

Pythonプログラミング入門：リスト

https://utokyo-ipp.github.io/2/2-2.html

探索

- データの集合の中から指定されたデータ（キーと呼びます）を効率的に探し出すようなアルゴリズムを探索アルゴリズムと呼びます.

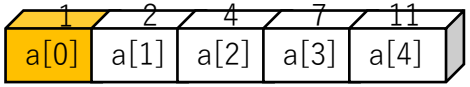
- 探索はデータを処理するための基本的な処理の1つとなります.
  - 例えば，探索により大量のデータから目的の情報を探すことができます.

- 探索の代表的なアルゴリズムとして線形探索と二分探索があります.

探索

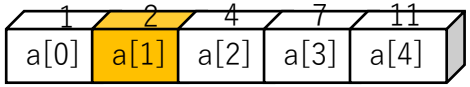
- 線形探索
  - データの集合（例えば配列）の先頭から順番にキーと比較していくことで探索を行います.

＜線形探索で数値データ[1,2,4,7,11]から4を探索する例＞

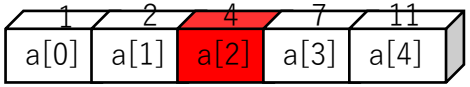


キー「4」の探索を配列の先頭a[0]から開始.

a[0]の値1は4でないので次の要素a[1]を探索



a[1]の値2は4でないので次の要素a[2]を探索

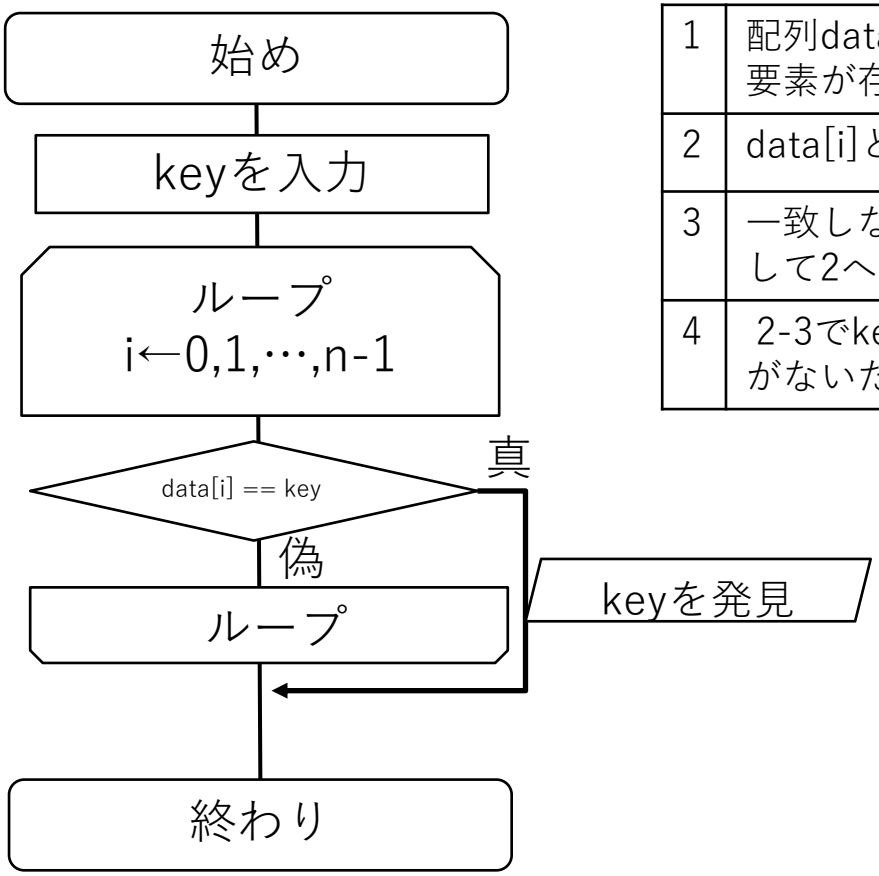


a[2]の値4がキー「4」と一致したので探索を終了

  - 線形探索は運が悪いと目的のキーを探すのにすべてのデータを調べる必要があり比較回数が多くなる場合があります.

線形探索のアルゴリズム

配列dataで管理しているn個のデータの中から，目的のキーであるkeyと一致するデータを線形探索するアルゴリズムのフローチャートは以下のようになります.



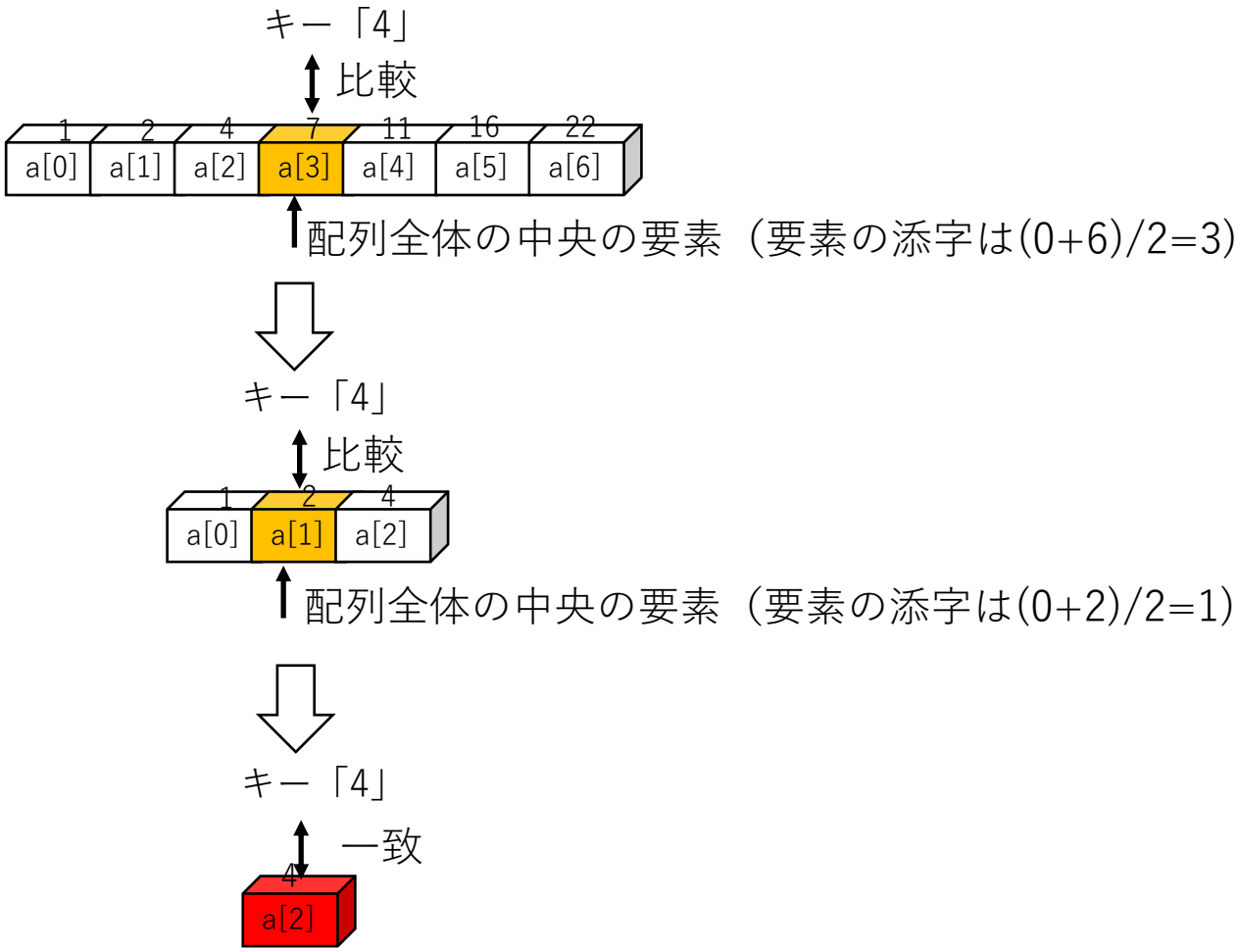
1	配列dataの添字iを先頭の要素の添字の値から始めて配列の要素が存在する間，2-3を繰り返す.
2	data[i]とkeyを比較し，一致したら探索終了.
3	一致しなければ配列の次の要素と比較するためにiを1つ増やして2へ
4	2-3でkeyを発見できなければ配列にキーと一致するデータがないため探索終了

二分探索

- 線形探索よりも効率的に探索を行う方法として二分探索があります.
- 二分探索
  - 二分探索では，データをあらかじめ決められた順番（数値であれば小さい順，文字列であれば五十音順など）で並べておき，探索の対象を半分ずつ狭めながらキーと比較していくことで探索を行います.
  - 例えば，小さい並べかえられた数値データの集合（例えば配列）の中央のデータとキーを比較します.
    - この時，キーが中央のデータより小さければ，次は中央のデータの前方（左半分）だけを調べればよいことになります.
    - 一方，キーが中央のデータより大きければ，次は中央のデータの後方（右半分）だけを調べればよいことになります.
  - 二分探索ではn個のデータについて最大で$\log_2 n+1$回の比較で済みます.

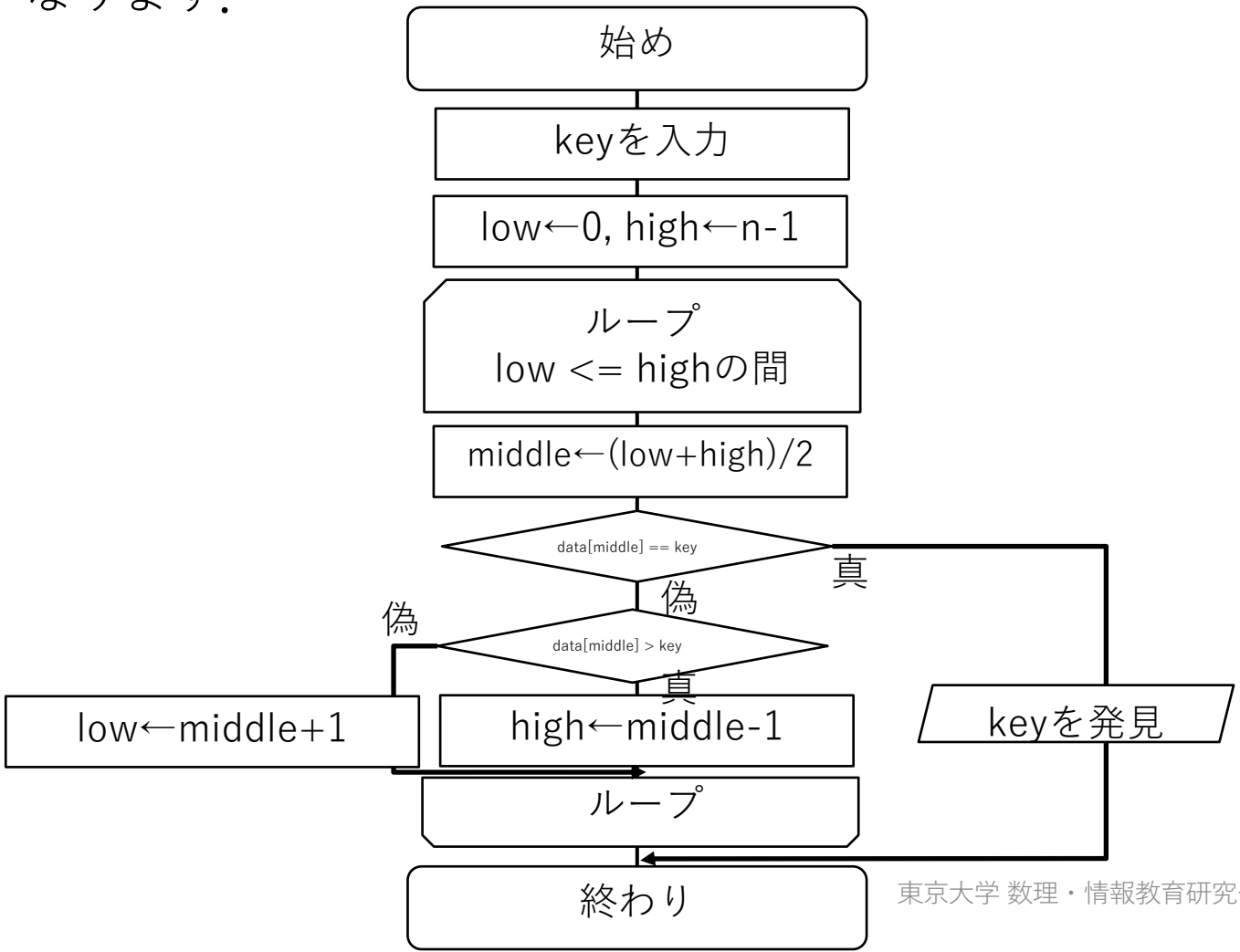
二分探索

<二分探索で数値データ[1,2,4,7,11]から4を探索する例>



二分探索のアルゴリズム

配列dataで管理しているn個のデータの中から，目的のキーであるkeyと一致するデータを二分探索するアルゴリズムのフローチャートは以下のようになります.



二分探索のアルゴリズム

1	配列の添字lowを探索範囲の先頭の要素を指す添字の値に，添字highを探索範囲の末尾の要素を指す添字の値にそれぞれ初期化
2	探索範囲が存在する間，3-6を繰り返す
3	添字middleをlowとhighの中央にする
4	data[middle]とkeyを比較し，一致したら探索終了
5	data[middle]とkeyを比較し，keyがdata[middle]より小さい場合，highをmiddle-1に設定し，探索範囲を前半分に狭め，4へ
6	data[middle]とkeyを比較し，keyがdata[middle]より大きい場合，lowをmiddle+1に設定し，探索範囲を後ろ半分へ狭め，4へ
7	3-6でkeyを発見できなければ配列にキーと一致するデータないため探索終了

整列

- データを決められた順に並べ替えるようなアルゴリズムを整列アルゴリズムと呼びます.
  - 例えば，昇順ではデータを小さい順に，降順ではデータを大きい順に並べます.

- 整列はデータを処理するための基本的な処理の1つになります.
  - 整列により大量のデータを順番に処理したりランキングしたりすることができます.

- 整列の代表的なアルゴリズムとして交換法があります.

交換法

- データの集合（例えば配列）の先頭から始め、隣り合うデータの大小を比較し、決められた順番と逆の順番の場合、データを交換することを繰り返していくことで整列を行います。

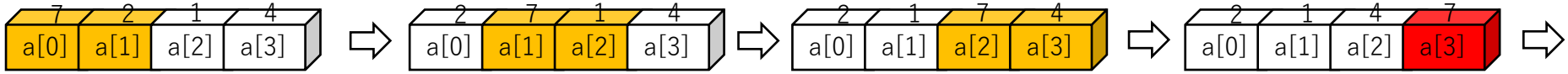
- 昇順に整列する場合は、データの比較と交換を最後まで行うと、最後のデータは最大値となります。この最後のデータを取り除いて再び交換法を適用することを繰り返していくことでデータの集合全体を並び替えることができます。

- n個のデータに対する交換法では最大でn(n-1)/2回の比較と交換を行うことになります。

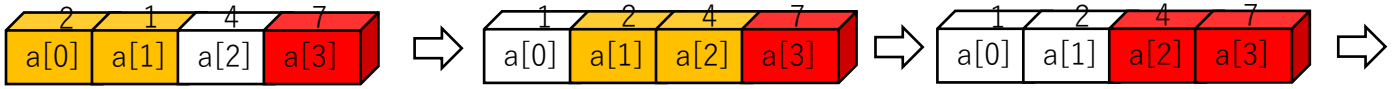
交換法

＜交換法で数値データ[7,2,1,4]を昇順に整列する例＞

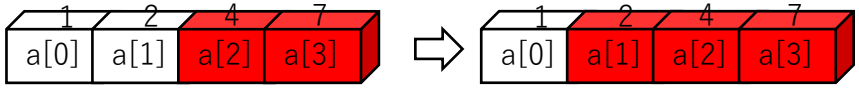
7と2を比較・交換　7と1を比較・交換　7と4を比較・交換　7が最大値と確定



2と1を比較・交換　2と4を比較・交換なし　4が次の最大値と確定

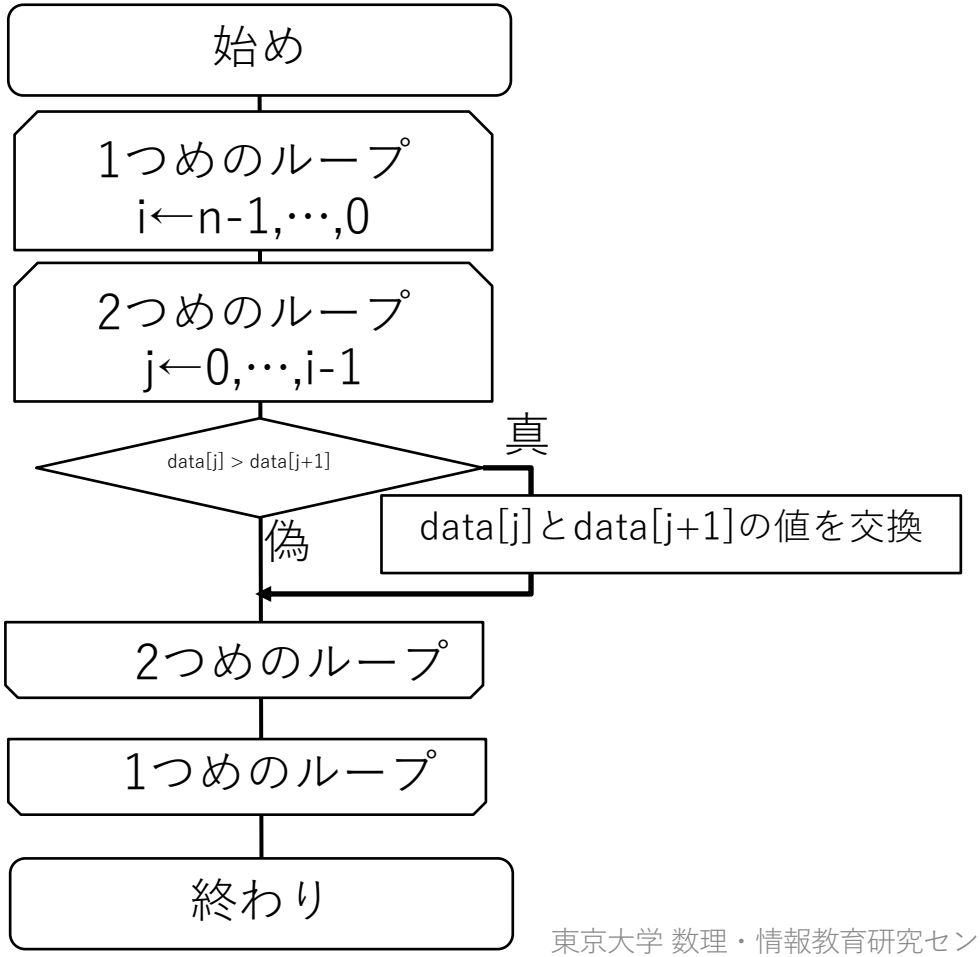


1と2を比較・交換なし　2が次の最大値と確定・整列終了



交換法のアルゴリズム

配列dataで管理しているn個のデータを昇順に並び替える交換法のアルゴリズムのフローチャートは以下のようになります.



1	整列範囲のデータ数iを配列dataの要素数に初期化
2	整列範囲が存在する間，3-6を繰り返す
3	配列dataの添字jを0に初期化
4	jが整列範囲の最後の1つ手前まで5を繰り返す
5	data[j] > data[j+1]であればそれぞれの値を交換
6	整列範囲の最後が確定したので整列範囲のデータ数iを1つ減らして整列範囲を狭めて2へ