コンピュータ・サイエンス2

第10回
情報ネットワーク(続き),データ構造とアルゴリズム

人間科学科コミュニケーション専攻
白銀 純子

第10回の内容

- 情報ネットワーク(続き)
  - インターネット上のアプリケーション
  - セキュリティ
- データ構造とアルゴリズム

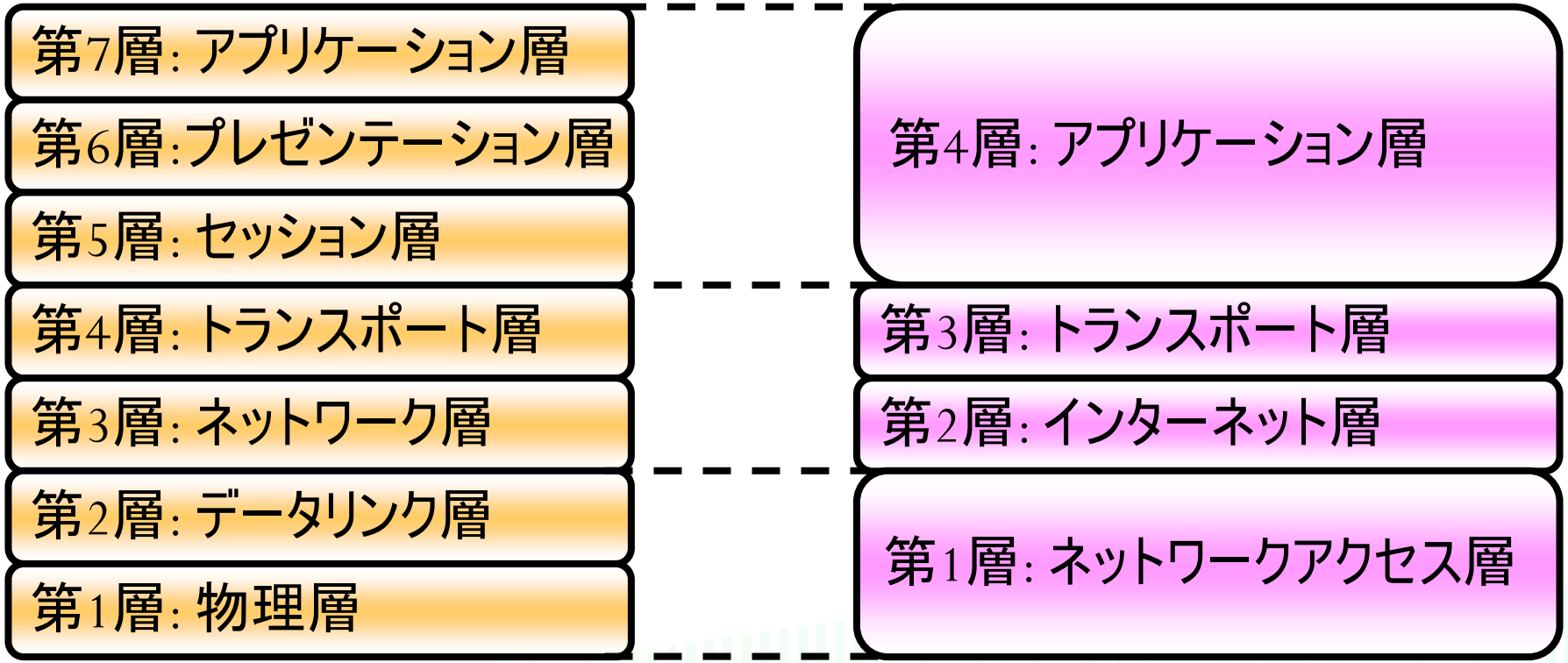
前回の復習

TCP/IPモデルとは?(p. 96)

- TCP/IP: データがインターネットを通るためのプロトコル
  - Transmission Control Protocol/Internet Protocol
  - インターネットでの標準規格
- コンピュータの通信機能を4つの階層に分割したモデル
  - 各階層ごとに必要な機能(プロトコル)を定義
  - 現在最もよく使われているモデル
    ※OSI参照モデルは、実際に利用するモデルの基礎

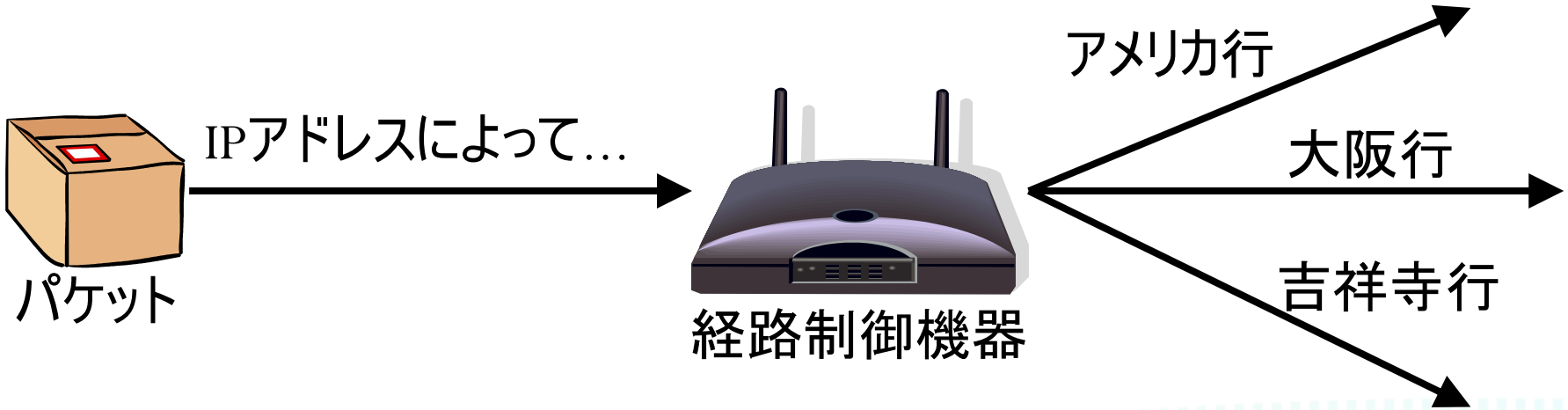
OSIとTCP/IPモデル(p. 96)

- OSIの7層とTCP/IPの4層との対応関係
  - OSI参照モデルと同じ名前の層があるが、必ずしも同じ役割をするわけではない



インターネット層のプロトコル(IP)(p. 96)

- IP
  - インターネットの世界でのコンピュータの住所(IPアドレス)を扱うためのプロトコル
    - 通信の宛先として指定される住所
  - IPアドレスに基づいて、送り先を決める経路制御機器で利用



トランスポート層のプロトコル[TCPとUDP](p. 96)

- TCP (Transmission Control Protocol)
  - データの通信前に、通信先との道筋を確保し、その上で送受信
    - データの破損・紛失があれば再送を要求　　コネクション型
    - 信頼性が高い分、様々な処理をするので速度は低
    - メールやWebのデータ、大きなファイルなどの破損・紛失があってはならないデータの送受信

- UDP (User Datagram Protocol)
  - 道筋を確保することなく、いきなりデータを通信
    - データの紛失・破損のサポートなし　　コネクションレス型
    - データのやり取り以外をしないので、速度は高
    - 小さいデータの送受信や実況中継(少々データが破損・紛失してもリアルタイム性が重要)に利用

デファクトスタンダード

デファクトスタンダード(p. 97)

- TCP/IPモデル
  - 階層化が不十分で厳密性が不足
  - but 最も広く使われていて、ネットワークの事実上の標準

デファクトスタンダード

デファクトスタンダード

- 市場で広く使われるようになったために、標準となること
  - ✓ 国際機関などが公的標準として定めたものではない
- 一度標準になると、関連する規格や商品が出て、さらに標準が地位を強化

標準化の流れ(p. 97)

- インターネット関連のプロトコル
  - RFC(Request For Comments)という文書により実現
    - IETF(Internet Engineering Task Force)という技術者組織の技術者が、新しい技術を提案(提案文書: RFC文書)
    - 提案に対して様々ン意見が出され、改良や修正
    - 最終的に、実証実験や正式な会議により、標準化が決定

議論の過程や標準化された規格は広く公開され、誰でも利用可能

➡ 自由で開放的な開発スタイルがインターネットの発展に寄与

but…自由で開放的なために、様々な問題も

- 知的財産の侵害
- コンピュータウィルス
- 不正アクセス

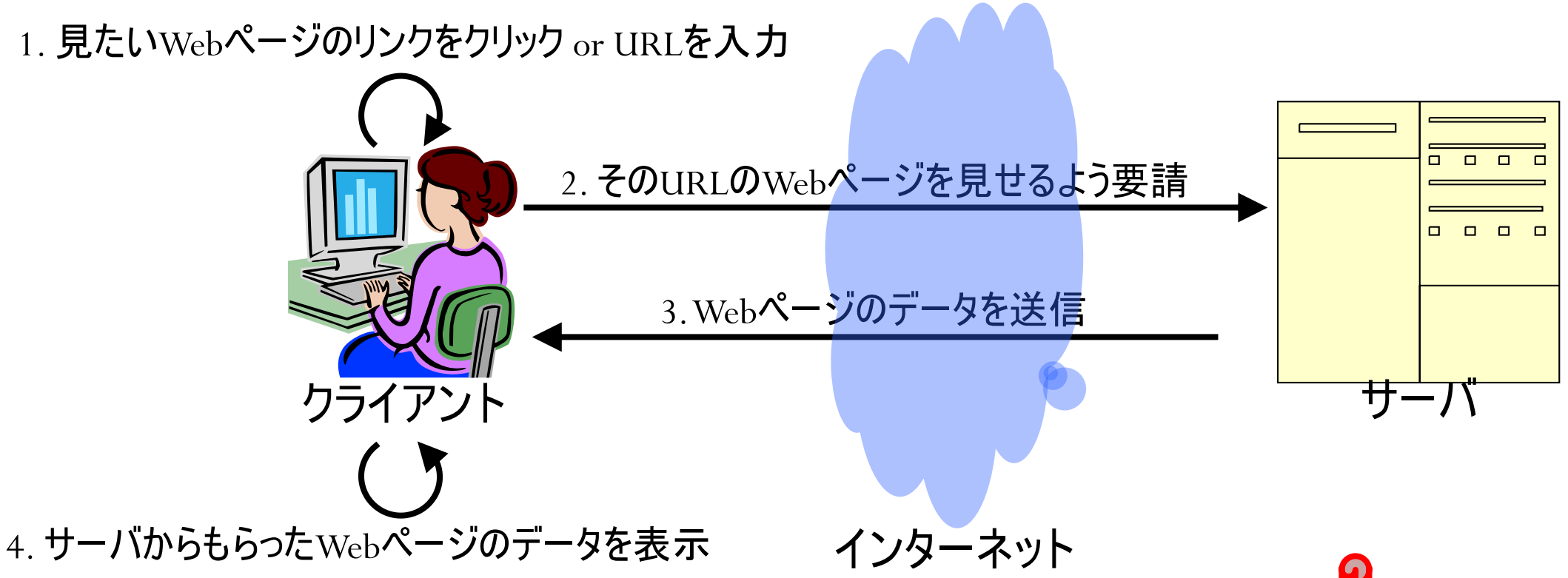
LANとインターネット

クライアントサーバ方式[1](p. 99)

- インターネットの世界で広く使われている、様々な処理を行うための方式
  - クライアント
    - サーバに要請をして、様々な処理をしてもらうコンピュータ
    - Ex. Webページを見せてもらう, 届いているメールを見せてもらう, etc.
  - サーバ
    - クライアントからの要請を受けて、様々な処理をするコンピュータ
    - Ex. Webサーバ, メールサーバ, etc.
  - インターネット
    - クライアントからの要請やデータをサーバに届けたり、サーバの返事やデータをクライアントに届けるための道路

クライアントサーバ方式[2](p. 99)

Ex. Webページの閲覧



Webページの管理をするサーバ:
Webサーバ

IPアドレスとドメイン

IPアドレスとは?(p. 99)

- インターネットの世界で通信を行うために、コンピュータの住所が必要
  - IPアドレス: 原則として世界中で一意(他のコンピュータと重ならない)住所
- 現在広く使われているIPアドレス:「.」で区切られた3桁の10進数を4つ並べた形
  - 1つ1つの10進数は、0～255(2進数で8桁)の間

IPアドレスの例
192.168.20.1 (11000000.10101000.00010100.00000001)

- 国際的な組織ICANN(The Internet Corporation for Assigned Names and Numbers)が管理
  - 各地の支部で実際の管理
  - IPアドレスを使いたい企業・組織は、ICANN(の自分の地域の支部)に申請

IPアドレスが足りない!!(p. 101)

- 現在の形式のIPアドレス: IPv4(Internet Protocol version 4)
  - 2^{32}個(約43億個)存在
- 現在の利用形態
  - IPアドレスを、世界中の人が分け合って利用
    - 世界の人口約70億人(使っていない人も多いが、使う人がどんどん増えている)
    - 1人で複数台の端末を利用(PC, スマートフォン, ゲーム機, etc.)

➡ いろいろな対処方法が考案・実践されたが、もう限界

IPアドレスの枯渇問題

抜本的な解決方法は??(p. 101)

- IPv6(Internet Protocol version 6)の形式のIPアドレス
  - 0011:2233:4455:6677:8899:aabb:ccdd:eeff
    という形式
    - 4桁の数(16進数)を「:」で区切って8つ並べて表現
  - 2^{128}個(約340澗(340×10^{36})個)のIPアドレスを利用可能
    - 数に限りはあるが、世界の人口などを考えても十分に足りる数
      (世界の人口が70億人として、1人あたり約5×10^{28}個)
    - 世界中のすべての端末(PC, スマートフォン, ゲーム機, 家電, etc.)に、重複なくIPアドレスを割り当てることが可能

IPv6への移行が必要!!(p. 101)

- IPv6へ移行しようとすると…(IPv4と扱い方が全く違う)
  - 古い機器ではIPv6に対応していない
    - 新しい機器の導入、新しいソフトウェアの開発や導入が必要
  - 一般利用者には、移行のメリットがわかりにくい
    - ネットワークが劇的に早くなったりするわけでなし
    - 機器やソフトウェアの導入が求められてデメリットに感じる人も…
  - 世界中での移行が必要
    - IPv4とIPv6が混在できるような仕組みもあるが、最終的には完全移行すべき

移行は急務だが、進んでいない

名前解決(p. 102)

DNS[1](p. 102)

- IPアドレス
  - インターネット上の住所を数値で表したもの

コンピュータにとってはわかりやすい

but 人間にとっては、数値の住所はわかりにくい!

Ex. 1: 電子メールアドレス

「利用者の名前@コンピュータの住所」の形になっている

→コンピュータは電子メールアドレスを

「利用者の名前@192.168.1.1」のように考えている

Ex. 2: WebページのURL

「http://コンピュータの住所/」の形になっている

→コンピュータはWebページのURLを

「http:// 192.168.1.1/」のように考えている

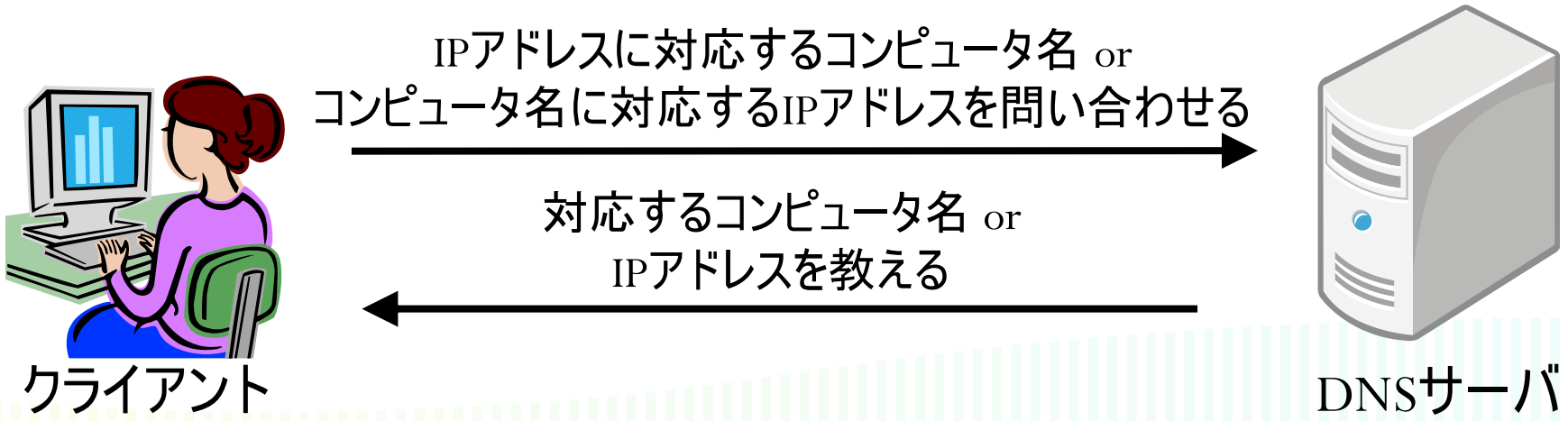
⇒ DNS(Domain Name Service)

DNS[2](p. 102)

- DNS: コンピュータの名前とIPアドレスの対応を管理するシステム
- コンピュータの名前を「ドメイン」と呼ばれる単位で管理
  - ドメイン: インターネット上での地域
- コンピュータの名前は、ドメインの前に追加
  - コンピュータ名＋ドメインで、インターネット上でのコンピュータのフルネーム (IPアドレスに対応する住所)
  - コンピュータのフルネームにドメインがついているので、世界中で一意の名前
    - それぞれのドメイン内で、コンピュータ名が重ならないようにすればOK
- Ex. コンピュータの名前: www.twcu.ac.jp
  - 「twcu.ac.jp」で、東京女子大学のドメイン
  - 東京女子大学の中の「www」という名前のコンピュータ、という意味

DNSサーバ(p. 102)

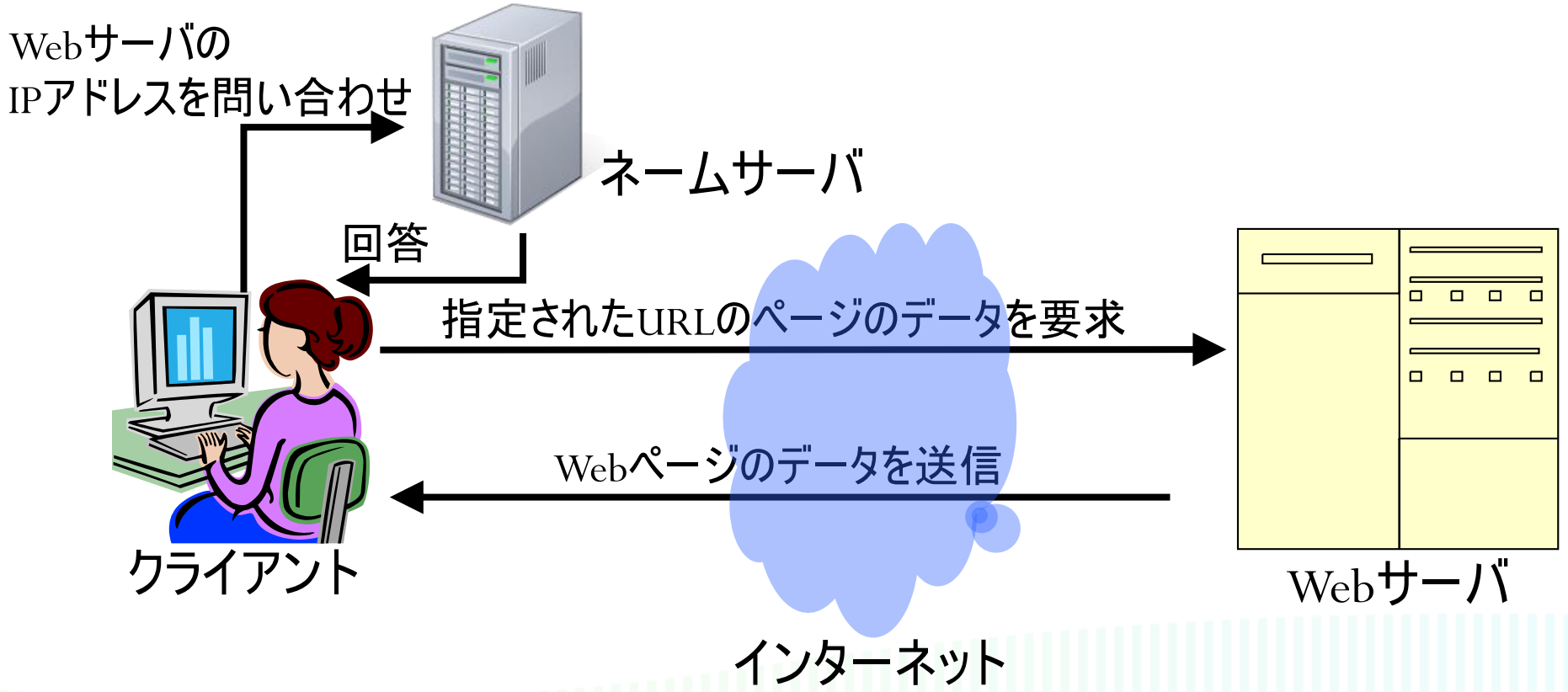
- IPアドレス⇔コンピュータ名の対応関係の管理:
  DNSサーバ(ネームサーバとも)
  - IPアドレスとコンピュータ名の対応関係の表の管理
  - クライアントからの問い合わせに応じて、対応するIPアドレス・コンピュータ名を返信



インターネット上のアプリケーション

WWW(p. 107)

- WWW: World Wide Web
  - 様々な情報を相互に参照しあえるようにした仕組み



URL(p. 109)

- Uniform Resource Locator
- Webページのありかを示す情報
- URLの形式: http:/ /Webサーバ名 /ファイルのパス

http - HyperText Transfer Protocolの略
(WebサーバとWebクライアントとのやり取りをするためのプロトコル)

Webサーバ名 - Webサーバのフルネーム(名前+ドメイン)

ファイルのパス - Webページの内容が書き込まれているファイルのパス(相対パス)

Ex: http://www.cis.twcu.ac.jp/~junko/Science/abc.html

- 東女のWebサーバの中の「~junko」というフォルダの中の「Science」というフォルダの中の「abc.html」というファイル

ネットワークセキュリティ

セキュリティの原則(p. 109)

- 不具合や設定ミスをなくした安全な情報機器を使うこと
- 重要な情報を、扱う権限がない者から隔離すること
- 権限がない者が情報を見てもわからないように隠ぺいすること

安全な情報機器(p. 109)

- 情報機器の機能: プログラムによって実現
  - 人間が作るものなので、不具合(バグ)やセキュリティホールをなくしきれない
    - セキュリティホール: 不正アクセスやウィルス侵入のもとになる不具合
- 初期状態で多数の機能が起動
  - 利用者が意識せずに機能が動いていて、不正アクセスの原因にも

- ソフトウェアのアップデートによるセキュリティホールやバグつぶし
- ウィルス対策
- 初期設定に頼らず、機能の要・不要を考えて利用

を心がけよう!

情報の隔離[1](p. 110)

- 重要な情報を守るために…
  - 守るべきものを隔離する
  - 必要最低限の人や機器だけが利用可能にする
    - 安全性と利便性は常に対立関係
      - ✓ 安全性を上げれば利便性が下がり、利便性を上げれば安全性が下がり…という関係

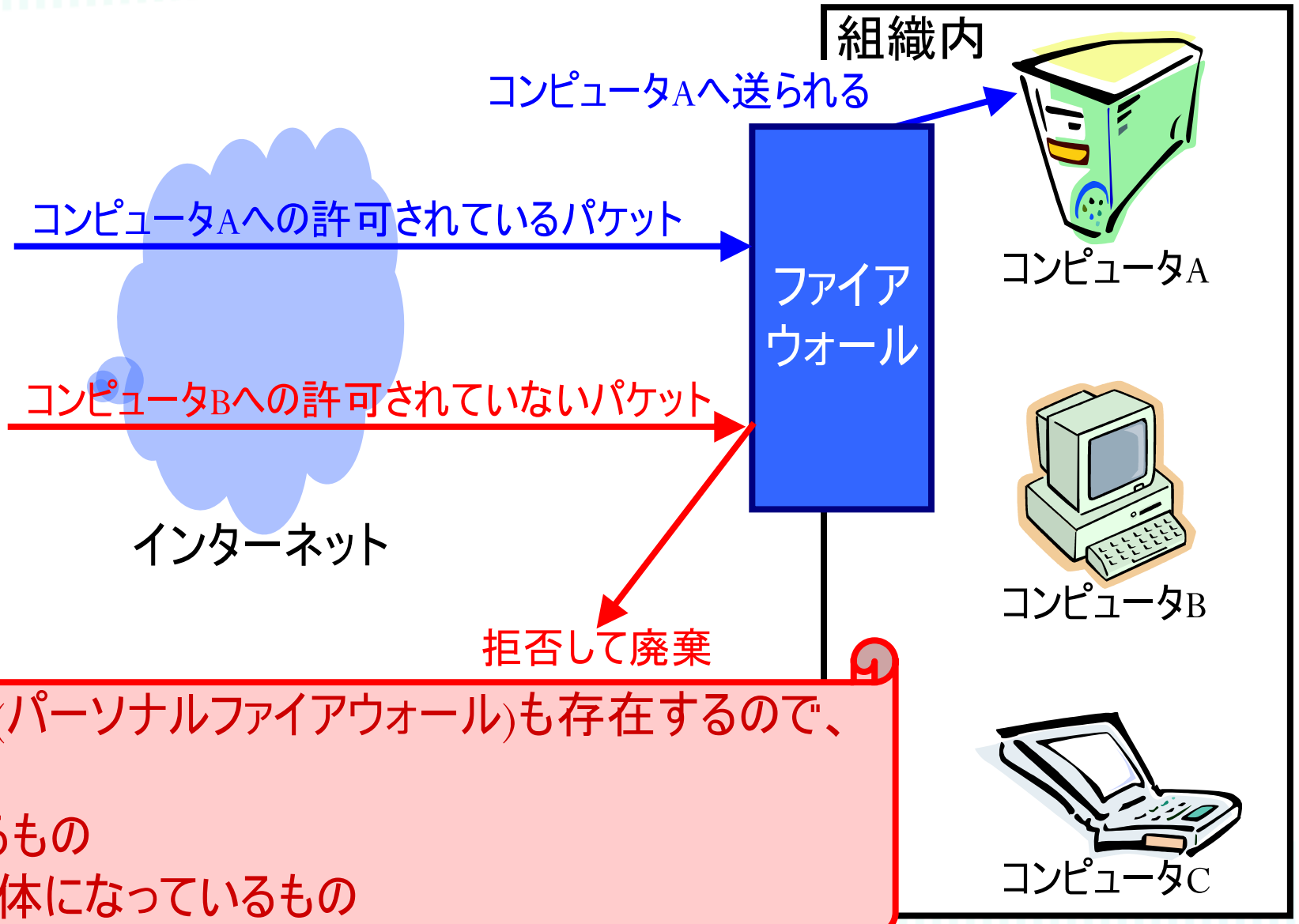
安全性と利便性のバランスを考慮してセキュリティポリシーで情報保護の方針を決定

情報の隔離[2](p. 110)

- ファイアウォールの設置
  - ファイアウォール: 組織内と外部との間に設置して組織内に不正にアクセスされないように監視するコンピュータ
  - 外部からのパケットの監視(アクセス制御)
    - 許可されていないIPアドレス(インターネット上の住所)からパケットが送信されていないか?
    - 許可されていないポート(データの出入り口)にパケットが送信されてきていないか?

許可されていないアクセスを遮断(フィルタリングと呼ぶ)

情報の隔離[3](p. 110)



個人用のファイアウォール(パーソナルファイアウォール)も存在するので、利用して情報を守ろう!

- OSに付属しているもの
- ウィルスソフトと一体になっているもの

情報の隠蔽[1](p. 111)

- インターネット上での通信(メール, Web, etc.)
  - データがそのままの形で送受信される
    = パスワードなどの個人情報がそのままインターネット上に流される

    = 途中で盗聴されてデータが盗まれる可能性もある

    ➡ インターネット上での盗聴は、仕組み上防ぐことは難しい

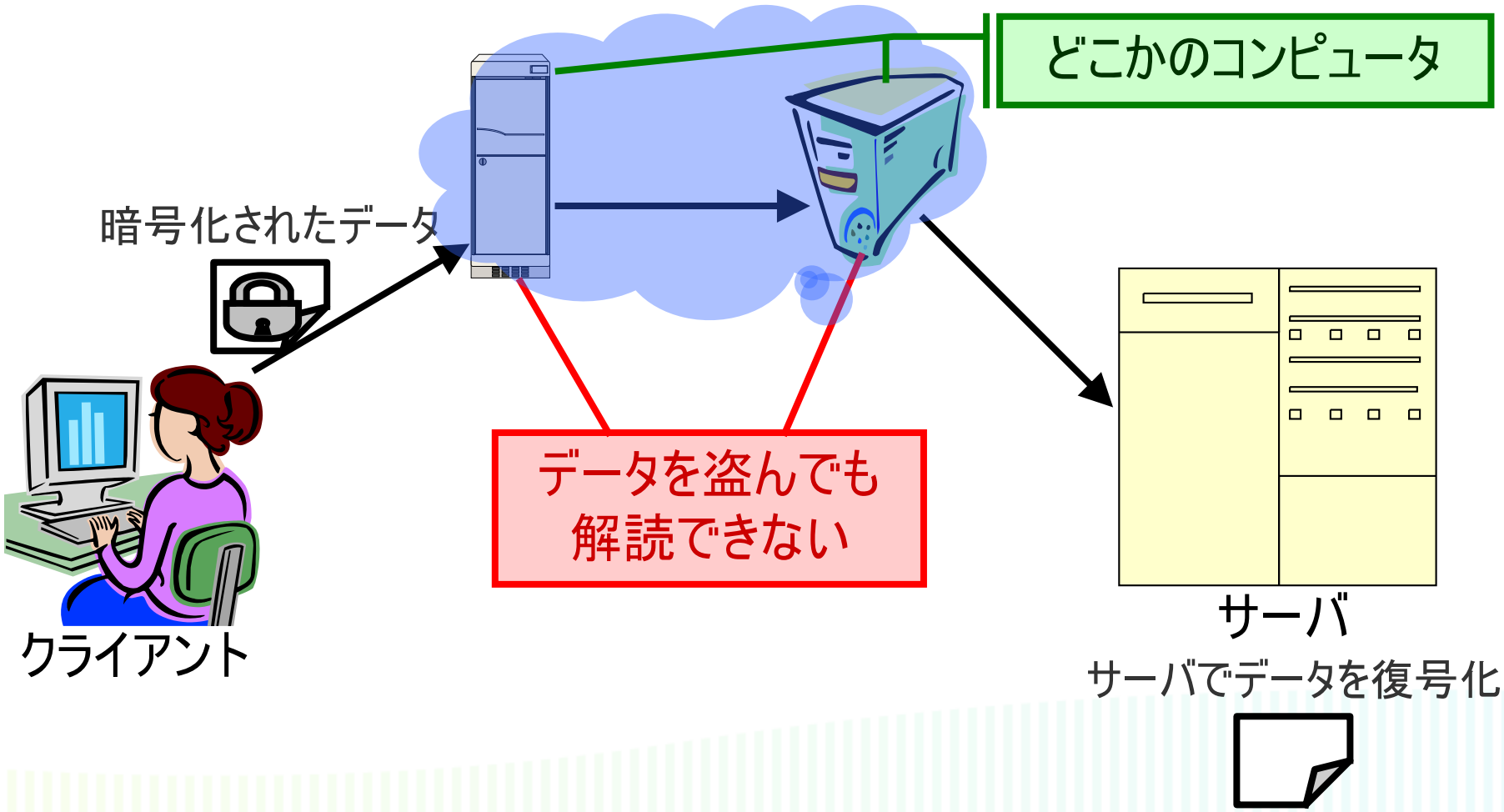
    - データを暗号化し、盗まれても中身を理解不能にする
    - ➡ 正当な受け取り主は、暗号を解読して本来のデータを見ることができるようにする

情報の隠蔽[2](p. 111)

- 暗号化: データを別の形に加工すること
  - データが元の形と違っているので、データを見ても内容がわからない
  - Ex. This is a pen. → Uijt jt b qfo.
    - 暗号化の方法: アルファベットを1文字後ろにずらす
- 復号化: 暗号化されたデータをもとの形に戻すこと
  - 復号化する方法を知らなければ、もとのデータの内容がわからない
  - Ex. Uijt jt b qfo. → This is a pen.
    - 復号化の方法: アルファベットを1文字前にずらす

情報の隠蔽[3](p. 111)

- **暗号化通信**: 利用者の使っているコンピュータで暗号化をして送り、サーバ側で復号化する通信方法



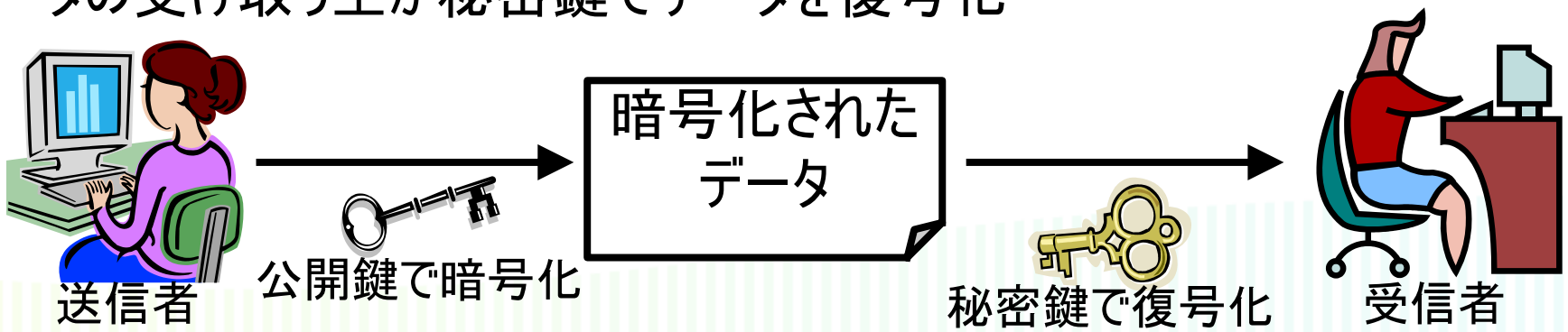
情報の隠蔽[4](p. 111)

- 共通鍵暗号方式(秘密鍵暗号方式とも呼ぶ)
  - データを暗号化するために「暗号鍵」を使う
    - 暗号鍵: データを暗号化するために使うキーワード(キーワードが長ければ長いほど、暗号が解読されにくい)
  - データを暗号化するときと復号化するときで、同じ暗号鍵を使う
  - 欠点1: データを送る側と受け取る側で暗号鍵を受け渡しする方法が難しい
    - 下手な方法では、途中で盗まれてしまう
  - 欠点2: 相手ごとに秘密鍵を用意する必要がある

情報の隠蔽[5](p. 111)

公開鍵暗号方式

- 「公開鍵」と「秘密鍵」という2種類の暗号鍵を使う方法
  - 公開鍵: データを暗号化するための暗号鍵
  - 秘密鍵: データを復号化するための暗号鍵
- データのやりとりの方法
  1. データの受け取り主が公開鍵と秘密鍵を作成
  2. データの受け取り主が公開鍵をデータの送信者に受け渡し
  3. データの送信者がデータを公開鍵で暗号化し、送信
  4. データの受け取り主が秘密鍵でデータを復号化



情報の隠蔽[6](p. 111)

- 公開鍵方式
  - 秘密鍵を知らなければ、データを復号化できない仕組み
  - 公開鍵と秘密鍵は対
  - 秘密鍵は、データを受け取る側しか知らない暗号鍵
    - 他の人に知られてはならない暗号鍵
  - 公開鍵は、他人に知られても良い暗号鍵
  - 利点: 秘密鍵を割り出そうとすると、膨大な時間がかかるので、事実上不可能
  - 欠点: 共通鍵暗号方式に比べて、復号化処理に時間がかかる

情報の隠蔽[7](p. 111)

- WWWでは、公開鍵暗号方式を利用
  - SSL(Secure Socket Layer)と呼ばれている
- Webの場合、URLが「https://」で始まっていれば、SSLでの通信
  - https: HTTP over SSL
  - 「http://」の場合は、普通の暗号化しない通信

Webでの個人情報の入力時には、URLがhttpsで始まっているかどうかを確認しよう!

認証(p. 113)

- 認証: 正しい権限を持った人かどうかを確認すること
- 認証の手段
  - パスワード: 最も一般的な方法
    - 手軽で広く用いられているが、推測や漏洩の危険性大
  - バイオメトリクス: 人体の身体的特徴を利用する方法
    - 指紋や虹彩、顔などの固有情報を利用
  - 電子署名: 文書を、作成者本人が作ったこと(改ざんされていないこと)を証明する方法
    - 秘密鍵で文書を暗号化
    - 公開鍵で文章を復号化

} うまく復号化できれば、改ざんされていない

データ構造とアルゴリズム

データ構造(p. 116)

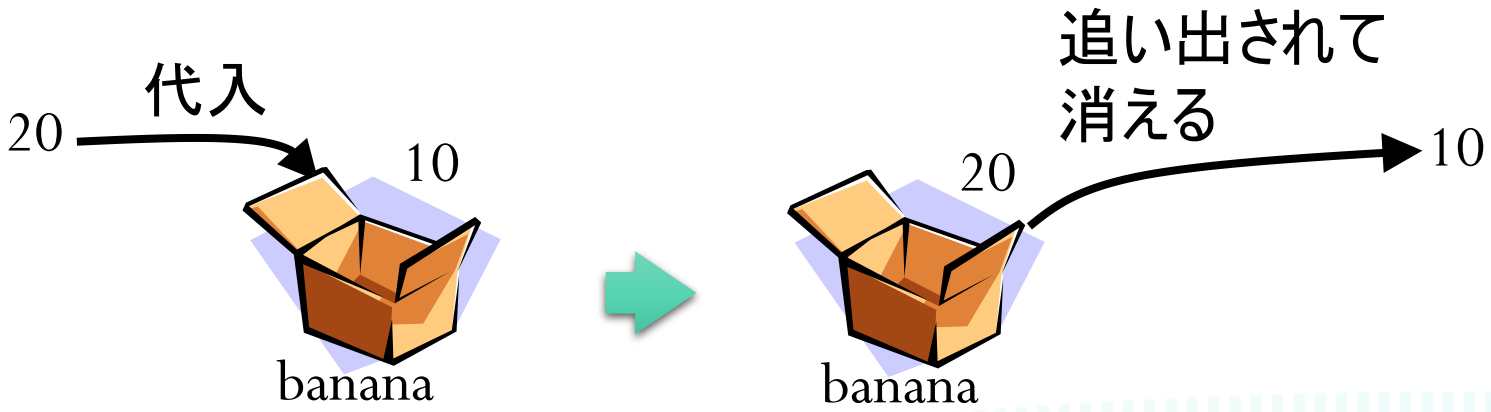
- データ構造: データをメインメモリに格納するときの形式
  - コンピュータはデータをメインメモリに記憶させて処理
  - コンピュータが扱いやすい形式で格納する必要
  - データの形式によって、プログラムの効率に大きく影響

変数(p. 116)

- 変数: 計算の対象や処理結果などのデータを記憶しておくための場所
  - データを入れておく「箱」と言うことができる
  - 変数には名前をつける
  - 変数にデータを入れることを「代入」と呼ぶ
  - 変数に入れられたデータのことを「値」と呼ぶ
  - 変数の名前を指定するとデータを取り出すことができる
  - 変数は扱うデータの個数分だけ用意する

変数の特徴[1](p. 116)

- 変数の中のデータを取り出しても、変数の中にデータは残っている
  - 「変数の値を参照する」とは、「箱の中のデータを見る」という意味
- すでにデータが入っている変数に別のデータを入れると、元のデータは消えてしまう
  - 1つの変数に入れておくことができるデータは1つだけ

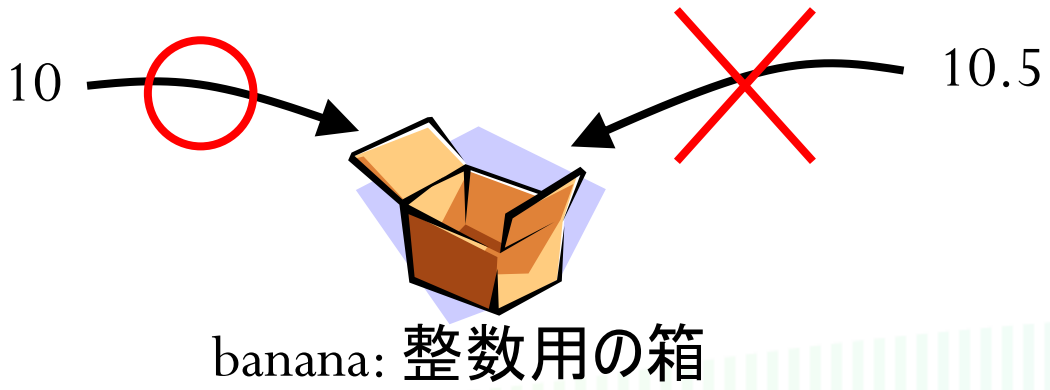


変数の特徴[2](p. 116)

- データの種類によって、違う箱を使う必要がある
  - 整数用の箱
  - 実数用の箱
  - 文字列用の箱
  - etc.

  - 整数用の箱に実数を入れることはできない
  - 文字列用の箱に整数や実数を入れることはできない

あらかじめ、「この名前の箱は整数用の箱」と、決めてコンピュータに知らせた上で箱を使い始める



「配列」って?[1](p. 116)

- データの種類(整数や小数)が同じで、処理方法も同じ変数をたくさん扱うときに利用する変数
  - たくさんの変数を1度にまとめて利用する方法

例えば…生徒の英語の成績を扱うプログラム(30人分)

出席番号1番の生徒の成績
出席番号2番の生徒の成績
….
出席番号30番の生徒の成績

} 30個の変数が必要!

→ english1, english2, english3, …, english30
のように用意して使うのは大変!

→ 「配列」を利用

配列って?[2](p. 116)

- 配列を利用するには…
  - 扱うデータのグループに名前を付ける(配列名)
    - Ex. 生徒の英語の成績: english
  - 配列名に番号(添え字)を「[]」でつけて、個々のデータを扱う
    - Ex. 生徒の英語の成績
      - 出席番号1番の生徒の成績: english[0]
      - 出席番号2番の生徒の成績: english[1]
      - ….
      - 出席番号30番の生徒の成績: english[29]

アルゴリズム

アルゴリズムとは[1](p. 118)

- アルゴリズム: ある問題を解決するときに必要な処理手順
- プログラムでの処理の方法を記述したもの
  - 何をどのように行うかを記述
  - コンピュータには手順を1つ1つ詳細に指示する必要
    - 人間には一言ですむような処理でも、コンピュータがその処理をこなすには、たくさんの手順が必要

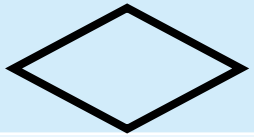
アルゴリズムの書き方(p. 119)

- 文章で書く
  - 箇条書きで書くことも多い
- プログラミング言語に自然言語を混ぜて書く(疑似言語)
  - 自然言語: 普段人間が話したり書いたりしている言葉
- 図で描く
  - フローチャート(流れ図)を使うことが多い

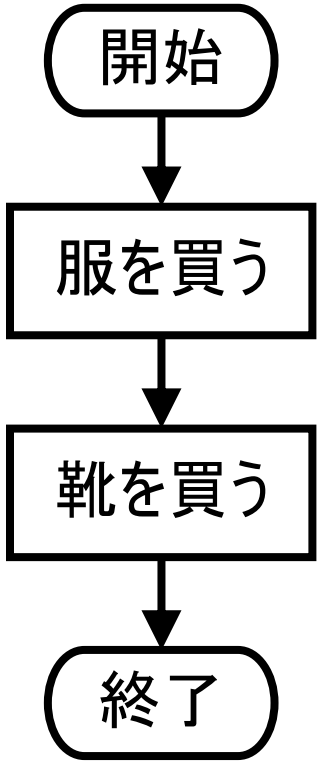
フローチャート[1](p. 119)

- 記号や矢印などを使って処理の流れを描いた図
- 順次処理、条件分岐、反復処理が基本
  - 順次処理:プログラム中に書いてある命令を、上から順に1つずつ処理すること
  - 条件分岐:ある条件を満たしたときとそうでないときで、処理内容が変わること
  - 反復処理:ある条件が満たされている限り、処理を繰り返すこと

フローチャート[2](p. 119)

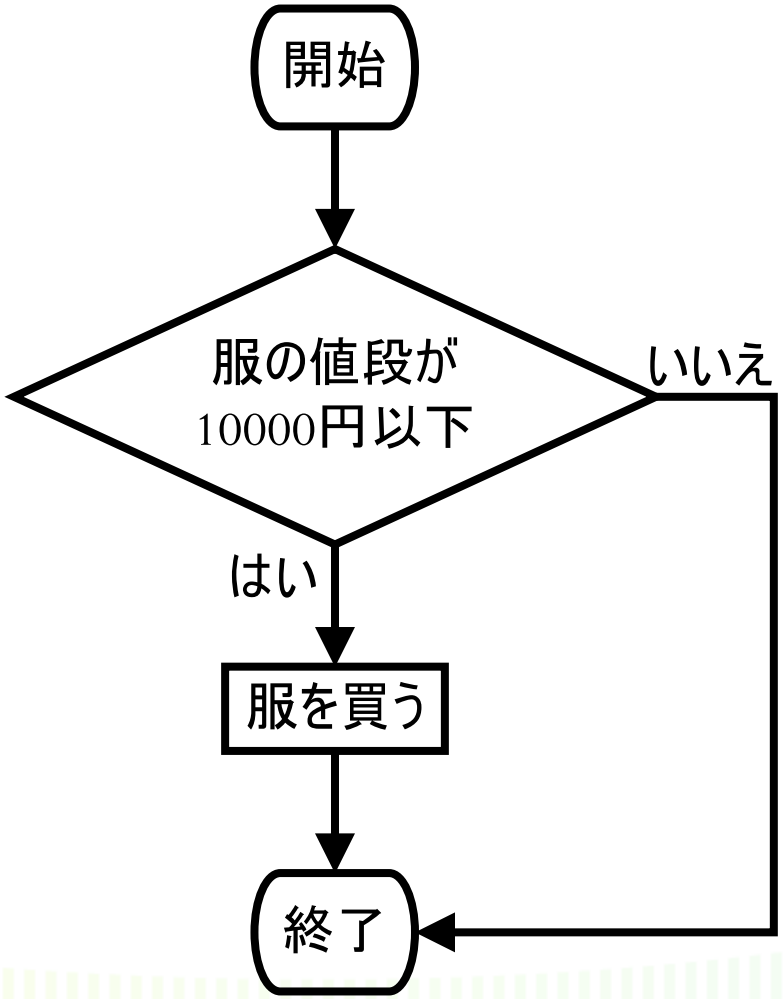
記号	意味
	開始と終了
	処理
	条件判断
	処理の流れ

順次処理の例

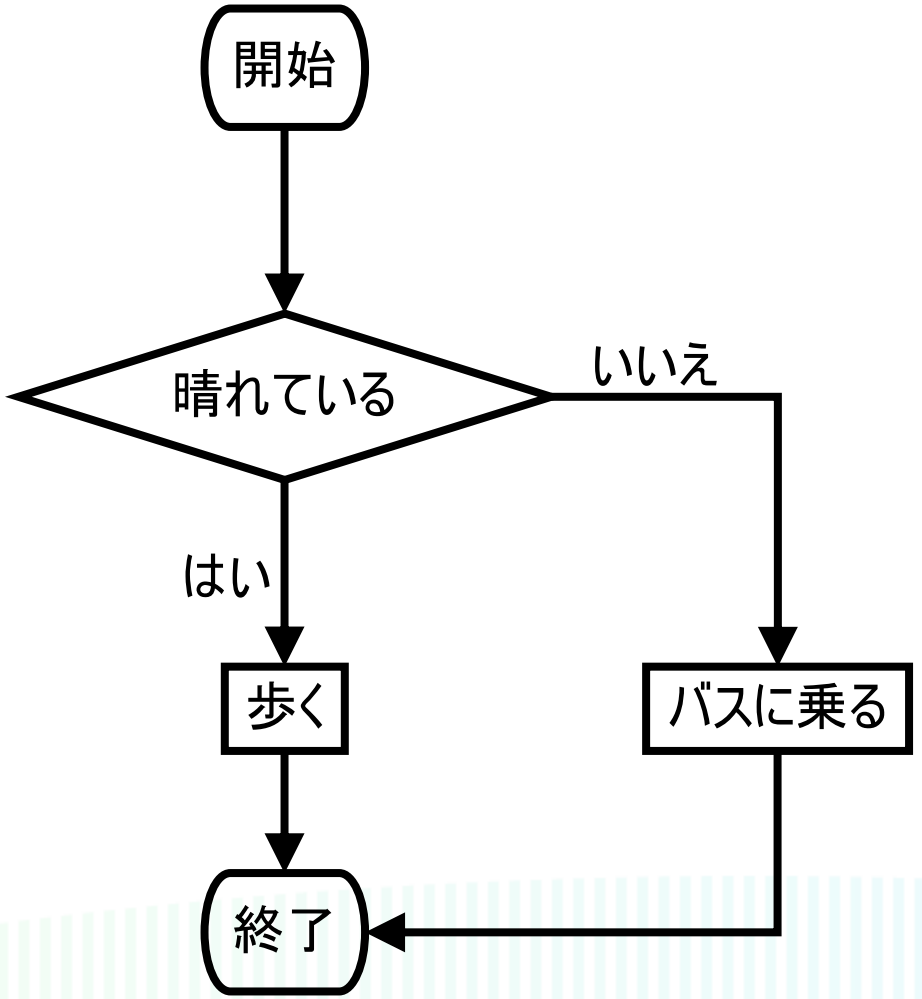


フローチャート[3](p. 119)

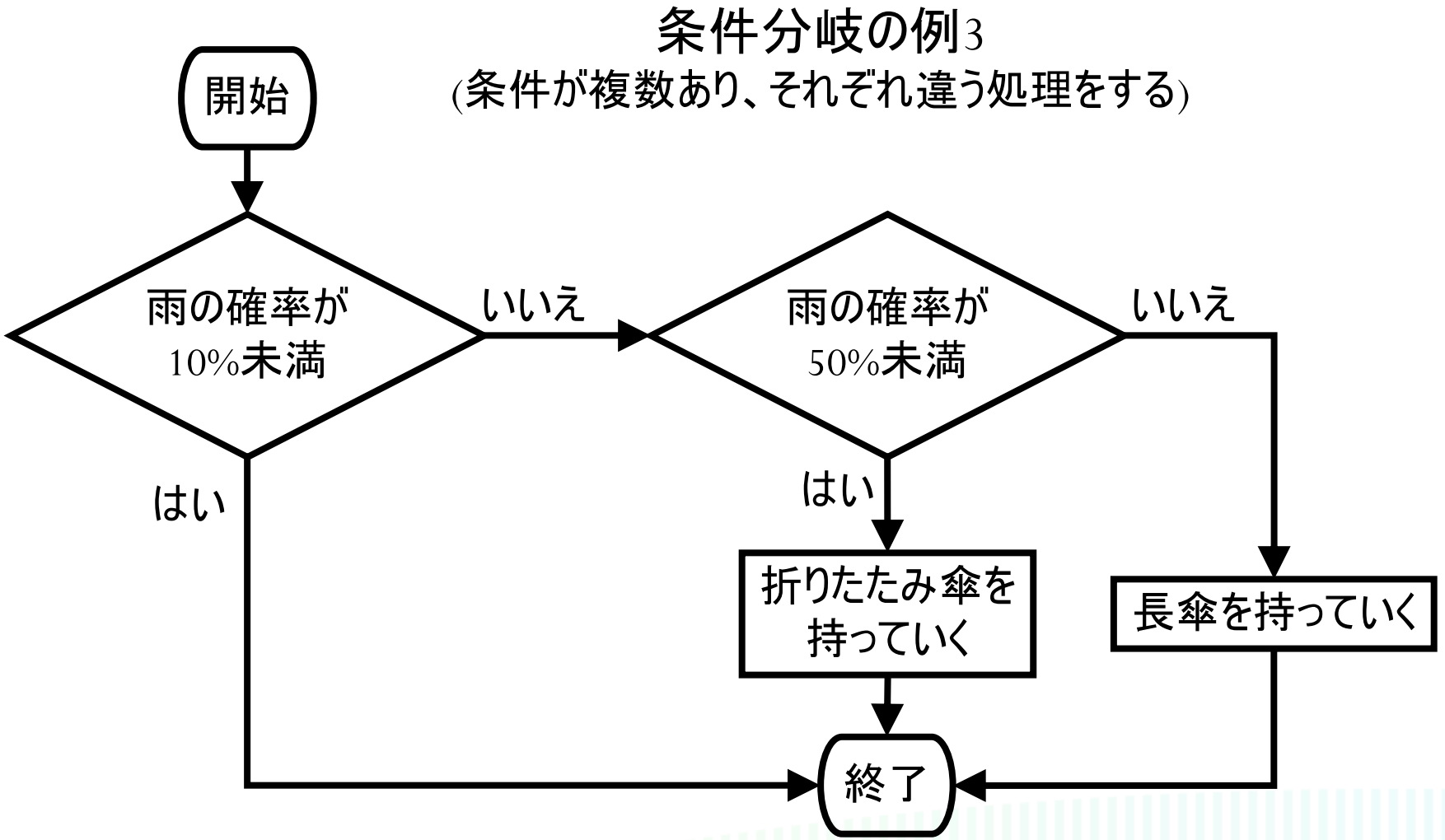
条件分岐の例1
(条件判断が「いいえ」の場合何もしない)



条件分岐の例2
(条件判断が「いいえ」の場合別のことをする)

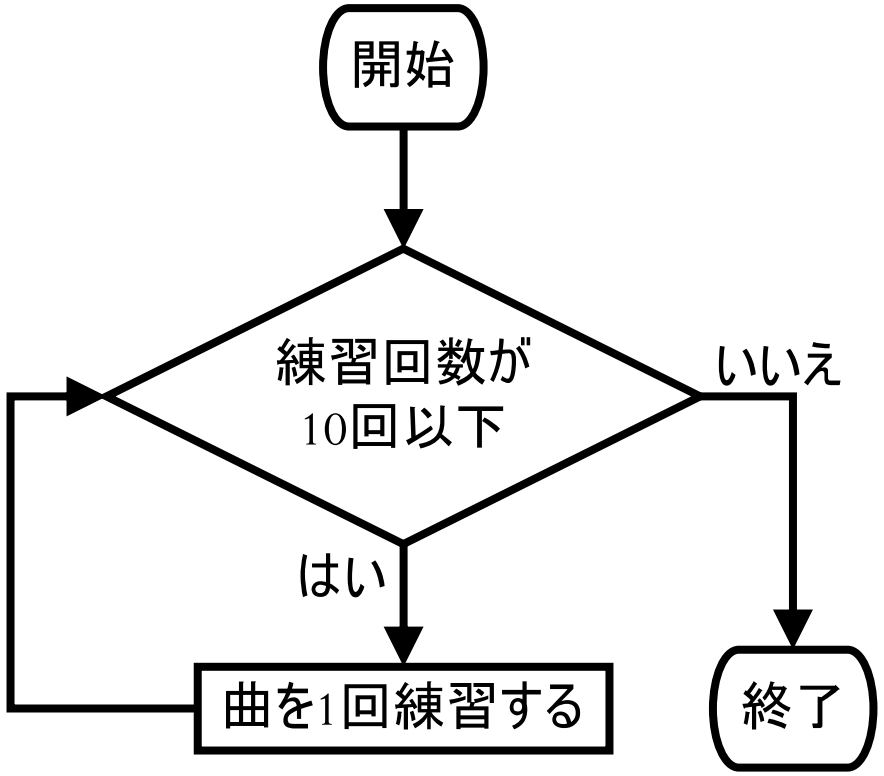


フローチャート[4](p. 119)

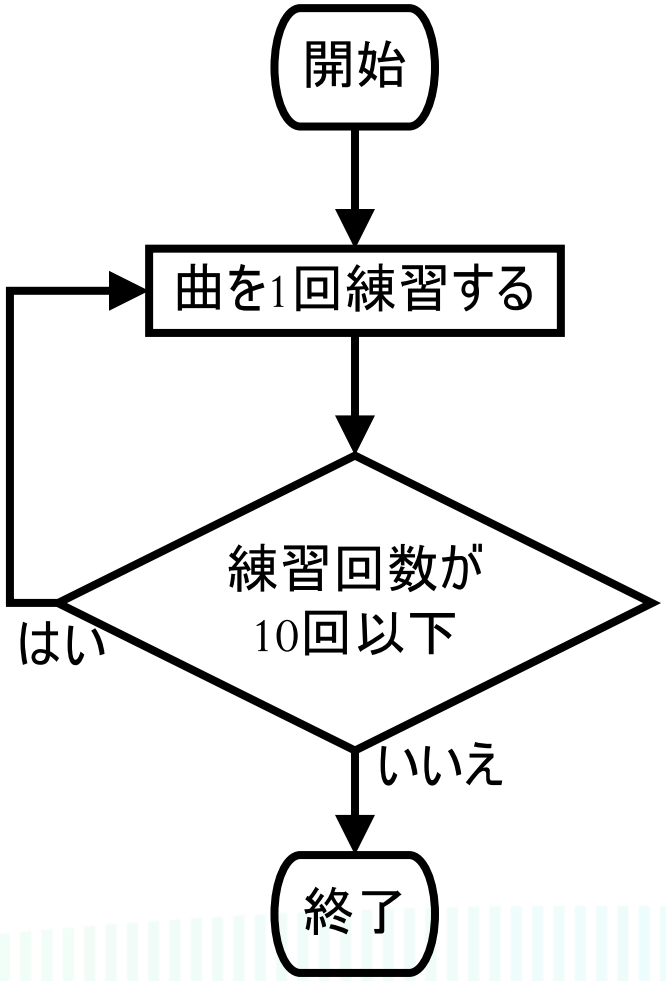


フローチャート[5](p. 119)

反復処理の例
(最初に条件を判断)



反復処理の例
(最初に処理をしてから条件を判断)



探索アルゴリズム

探索アルゴリズム[1](p. 121)

- **探索**: たくさんのデータから目的のデータを見つけること

Ex. 高校の生徒の得点を管理するプログラム

- 出席番号5番の生徒の英語の点数を知りたい
  →高校の生徒の配列から、出席番号が「5」というものを探す
- 「東京子」という生徒の国語の点数を知りたい
  →高校の生徒の配列から、名前が「東京子」というものを探す

探索アルゴリズム[2](p. 121)

- 様々な探索アルゴリズム
  - 逐次探索
  - 2分探索
  - 自己組織化探索
  - 2次元探索
  - 補間探索
  - ハッシュ法

逐次探索[線形探索](p. 121)

- データを前から順番に比較して探していく方法

データの中から「98」を見つけたい!

添え字	0	1	2	3	4	5	6	7
データ	21	13	98	31	44	87	72	50

1. 添え字「0」のデータをチェック: 違う

添え字	0	1	2	3	4	5	6	7
データ	21	13	98	31	44	87	72	50

2. 添え字「1」のデータをチェック: 違う

添え字	0	1	2	3	4	5	6	7
データ	21	13	98	31	44	87	72	50

3. 添え字「2」のデータをチェック: 違う

添え字	0	1	2	3	4	5	6	7
データ	21	13	98	31	44	87	72	50

二分探索[1](p. 121)

- 小さい順に並べられたデータの中から、中央のデータと目的のデータを比較することで、探す方法

1. 中央のデータと探したいデータを比較する
2. 1. の結果、中央のデータが大きければ、探したいデータは右半分、そうでなければ左半分を、探索対象とする

中央のデータと探したいデータが同じになるまで繰り返す

二分探索[2](p. 121)

データの中から「98」を見つけたい!

添え字	0	1	2	3	4	5	6	7
データ	13	21	31	44	50	72	87	98

1. 中央のデータ(44)と目的のデータ(98)を比べる
   →目的のデータ(98)の方が大きいので、右半分は探索対象にする

添え字	0	1	2	3	4	5	6	7
データ	13	21	31	44	50	72	87	98

2. 中央のデータ(72)と目的のデータ(98)を比べる
   →目的のデータ(98)の方が大きいので、右半分は探索対象にする

添え字	4	5	6	7
データ	50	72	87	98

二分探索[3](p. 121)

3. 中央のデータ(87)と目的のデータ(98)を比べる
   →目的のデータ(98)の方が大きいので、右半分は探索対象にする

添え字	6	7
データ	87	98

4. 中央のデータ(98)と目的のデータ(98)を比べる
   →同じなので見つかった!

添え字	7
データ	98

整列(ソート)アルゴリズム(p.123)

整列[ソート](p. 123)

- ソート: 複数の数を小さい順or大きい順に並べること
  - 選択ソート
  - バブルソート
  - 挿入ソート
  - クイックソート
  - etc.

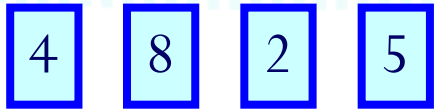
選択ソート[1](p. 123)

- 変数t: 並べ替えをする数の中で最も小さい数を入れておく変数
- 変数i: 並べ替えをする数の中で、tが最初から何番目の位置にあるかを表す変数
- 変数j: 何回繰り返したかを数えるための変数
- 並べ替えをする数はn個とする

選択ソート[2](p. 123)

1. tに並べ替えをする数の一番最初の数を代入する
2. iに1を代入する
   - 1: 並べ替えをする数の一番最初の数の位置
3. jに2を代入し、1ずつ増やしながらnになるまで以下を繰り返す
   - tがj番目の数より大きいならば、j番目の数をtに代入し、iにjの値を代入する
     - この処理を1回するごとにjの値を1増やす
     - jの値は、現在調べている数の位置になる
4. 並べ替えをする数の一番最後の数とtを入れ替える

選択ソート[2](p. 123)



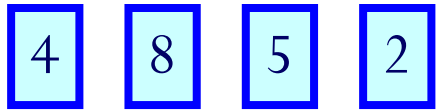
ステップ1:
- tに4を代入する
- iに1を代入する

ステップ2:
- jに2を代入する
- tの値(4)と8を比べる
- tの値の方が小さいのでそのまま

ステップ3:
- jの値を1増やす(3になる)
- tの値(4)と2を比べる
- tの値の方が大きいのでtに2を代入し、iにjの値(3)を代入する

ステップ4:
- jの値を1増やす
- tの値(2)と5を比べる
- tの値の方が小さいのでそのまま

ステップ5:
- tの値(2)を最後に置く
- これまで最後だった数(5)をi番目に入れる

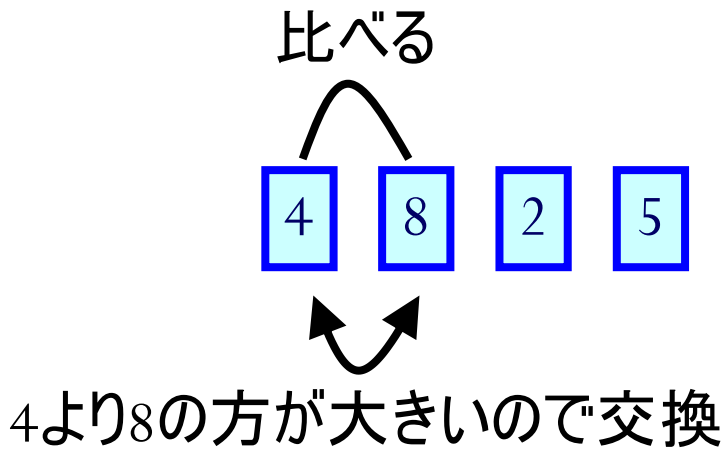


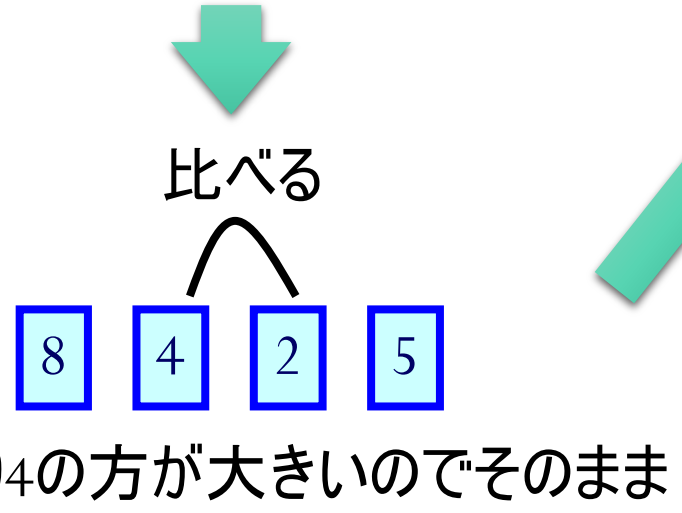
- 最も小さな数が一番後ろに来る
- 次は、一番後ろの1つ前まで(8, 4, 5)で同じようにする

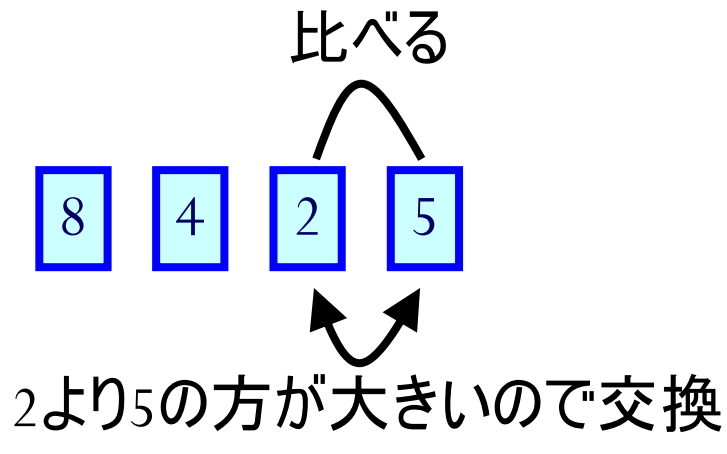
バブルソート[1](p. 124)

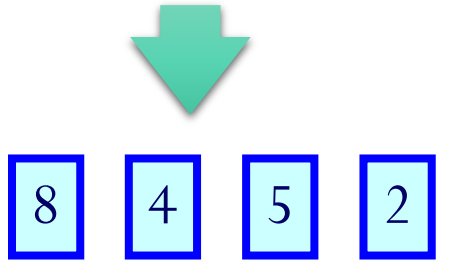
- 前から2つずつ、数の大きさを比較して、小さい数を後ろに送っていく
  - 最後まで調べると、最も小さな数が一番後ろにある
  - この作業を、並べ替える数の個数だけ繰り返すと、数が大きい順に並ぶ

バブルソート[2](p. 124)









- 最も小さな数が一番後ろに来る
- 次は、一番後ろの1つ前まで(8, 4, 5)で同じようにする

その他

- 挿入ソートのアルゴリズム(p. 125)は、教科書をよく読んでおくこと