情報科学概論

本講義の位置づけ

- 情報系以外の出身者を対象
- 情報科学の基礎を初等的に解説

担当：楫，中島，関，杉本（講義順）

日時：5/23, 5/30, 6/6, 6/13，各 4, 5限

- 基本的に，各講義は独立した内容

評価，成績

- 出席とレポートによる（本講義中で説明）

本日の講義

情報処理の理論と仕組み

- オペレーティングシステム概論（4限）
  - 計算機のオペレーティングシステム（OS）について知る
  - 歴史的経緯をなぞりつつ，現代のOSが備える機能を概観

- 情報理論入門（5限）
  - 「情報」を数学的・工学的に取り扱うための基礎技術
  - 情報の計量，圧縮，誤り訂正について学ぶ

本資料：http://narayama.naist.jp/~kaji/lecture/

オペレーティングシステム概論

オペレーティングシステム:

- Windows, MacOS, Linux etc....
- 工業製品であると同時に，情報科学の成果の集大成
- science と engineering を結ぶ「橋」（のひとつ）

本日の講義内容:

- オペレーティングシステム（OS）とは何か
- OSの役割とは
- OSが備えている機能について

OSとは何か：OSのない（？）計算機

昔のパソコン（マイコン）：

- プログラムを実行するには，カセットテープからロード
- 同時に一つのプログラムしかロードできない
- メインメモリのどの部分を使用するか，強く意識
- 入出力装置へのアクセスも直接的

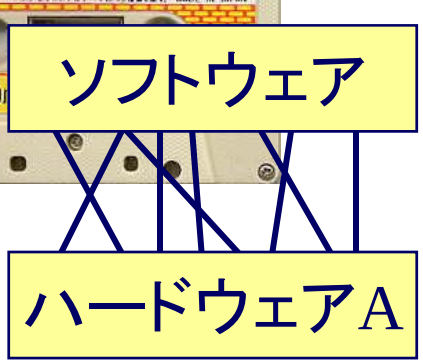
ハードウェアとプログラムが
直接的に結びついている

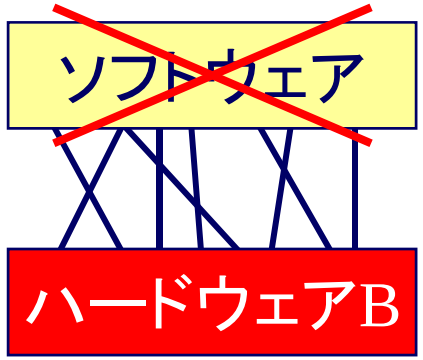
シャープ MZ-80C，1980年
Z-80 (2MHz), 4KB ROM + 48KB RAM
写真出典 http://www.itoi.co.jp/attic/time3.html

OSがないと何が困るか

ハードウェアとプログラムが直接的に結びついていると．．．

- それぞれの能力を最大限引き出せる
- 一部の構成が変わると，全体の構成にも影響





MZ-80C

日本電気 PC-8001

http://www.itoi.co.jp/attic/time41.html

OSのある世界

現在のパソコン：OS搭載

- プログラムの格納場所やロード方法は意識不要
- ハードウェアを直接駆動する必要もない
- 機器が多少変化しても，プログラムは共通利用可能
- 複数のプログラムを安全に同時実行可能

- ハードウェア構成の本質的な部分は，昔と同じ
- 昔は人間（プログラム）が直接やっていたことを，
  オペレーティングシステム（OS）がサポートしてくれる
- OSは，裏方仕事を引き受ける特別なプログラム

OSの役割

- OS：裏方仕事をひきうける特別なプログラム
  - システムを使いやすくする
  - 計算機資源を効率よく利用する
  - 複数のサービスを同時かつ安全に提供する

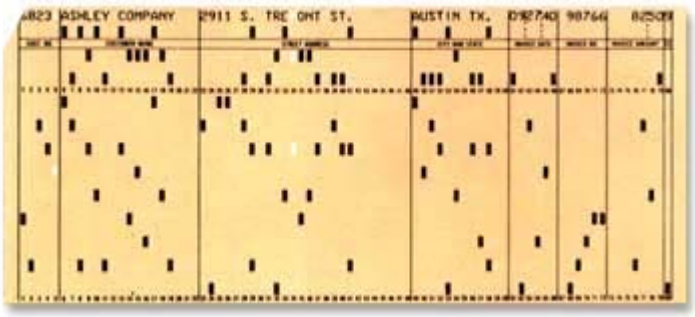
  etc...

- 現在のOS：非常に複雑で，多様な役割を負う
- 昔はもっとシンプル：

  歴史を追うことで，実像が見えてくる．．．かも

初期のコンピュータシステム

- 草創期のコンピュータ:
  - パンチカードからプログラムやデータを入力
  - パンチカードには機械語でプログラムを記述
  - 出力は紙テープやラインプリンタ
  - 別プログラムに切り替えるには，システムリセット



パンチカード

- プログラムの作成が面倒
- カードの取り扱いが大変
- プログラムの切り替えも面倒

高級言語，コンパイラ

オペレーティングシステム

第1世代のOS

- 第1世代のOS（1950年代）
  - 主として，プロセッサの利用効率の向上が目的
  - プログラムの切り替え中は，プロセッサが遊んでいる
    - 切り替えを自動的に行い，遊び時間をなくそう

      ⇒ バッチジョブの概念が登場

- バッチジョブ＝「コンパイル，プログラム起動，データ投入などの一連の操作を一まとめにしたもの」

第1世代OSの機能

- 第1世代OSが提供する機能
  - バッチジョブの受付
    ユーザから投入されたジョブを受け付ける
  - バッチジョブの切り替え
    ジョブ実行環境を初期化し，新しいジョブを開始する

第2世代のOS

- 第1世代OS：各ジョブが順番にプロセッサを占有する
  - 柔軟性の欠如．．．「急ぎの仕事」に対応できない
  - 応答性の悪さ．．．プログラムミスの発見が遅れる

- 第2世代のOS（1960年代前半）
  - ユーザの利便性を改善することが強く意識される
  - 「複数のプログラムが同時並行的に実行」．．．
    されているように見せる仕組みの実現

ユーザは，「自分が計算機を占有している」ように使える

タイムシェアリングシステム

タイムシェアリングシステム（TSS, 時分割処理）

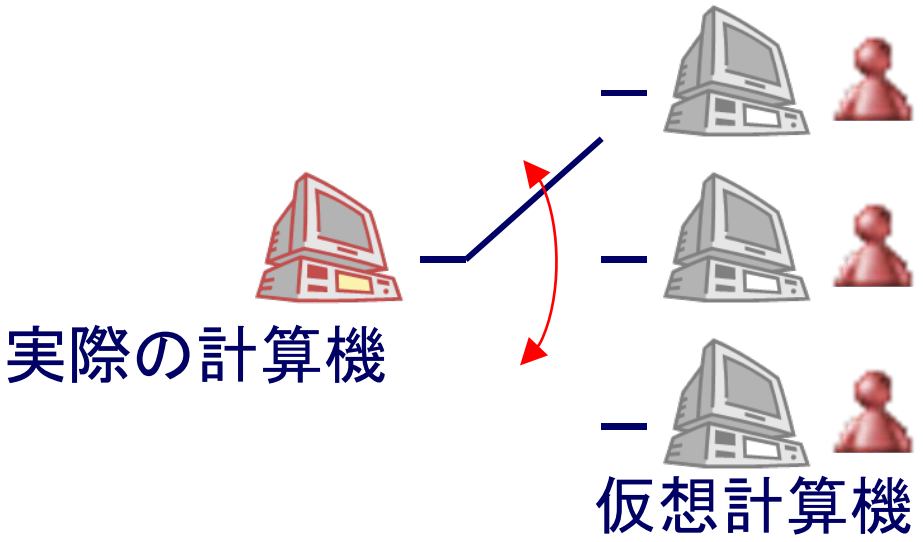
- プロセッサが，複数のプログラムを少しずつ実行する
- 短い時間（タイムスライス）で，プログラムを切替える

- トータルの実行時間は変わらない
  （切替オーバヘッドのため，効率は若干低下）
- ユーザの利便性は向上

⇒ 生産性の著しい向上

仮想計算機実現手段としてのTSS

- 時分割処理は，ユーザが計算機を独占しているように「錯覚」させるための仕組み
- 各ユーザに「独立した仮想計算機」を提供している
  - 他ユーザのプログラムの影響を受けないように．．．
  - ユーザのミスがシステム全体に影響しないように．．．



プロセスの概念の登場

- 独立した仮想計算機：
  - 個別のプログラム
  - 個別の計算環境（レジスタやメモリの内容）

  を保有している

- プロセッサは，時分割で，

  各仮想計算機の動作をシミュレートする

  と考えることもできる

計算環境まで含めた仮想計算機をプロセスと呼ぶ

第3世代のOS

- 第2世代OS:
  - TSSにより，「仮想計算機」をユーザに提供

- 第3世代のOS（1960年代後半～70年代前半）
  - 抽象性と汎用性の導入
  - 仮想計算機:
    - 論理的な存在で，比較的自由に設計可能
    - ハードウェアが違っても，同じ仮想計算機を！
    - ハードウェア上の制約からの脱却

      ⇒ 仮想記憶などの技術が確立

マイルストーン：IBM OS/360

- IBM System/360：
  - 1964年発売の汎用計算機ファミリー
  - 目的や用途に応じて，自由度の高い組合わせが可能

IBM System/360
IBMのWebページより

- IBM OS/360：
  - IBM System/360で採用されたOS
  - ハードウェアの差異を吸収し，高い汎用性を実現
  - 仮想記憶等，最新の技術を導入

第4世代のOS

- 第4世代のOS（1970年代後半～現在）
  - 基本的には，第3世代までの延長
  - プロセス間通信やネットワーク対応等，
    「協調作業」をサポートする仕組みが特徴的

現在のOS（1）

現在のOS：多くの役割を背負っている

1. 計算資源の有効利用
   - プロセッサを遊ばせない

2. ハードウェアの差異の吸収
   - プログラムの再利用が可能となる

3. 論理的で使いやすい計算環境の提供
   - プログラミングが楽になる
   - ユーザの負担を小さくする

現在のOS（2）

4. プロセスの独立性・安全性保障
   - 他のユーザに邪魔されない計算環境を実現
   - あるユーザのミスが，他のユーザやシステム全体に影響しないような仕組みを実現

5. 協調作業をサポートする仕組みの提供
   - 同一計算機内でのプロセス間通信
   - ネットワーク機能等，外部世界との架け橋

代表的なOS：UNIX（1）

- 元来はミニコン，ワークステーション用OS
- 1970年代初頭に誕生，他のOSに大きな影響を及ぼす
  - モジュール化，モジュールの単機能化
  - パイプ，ソケット等の概念
  - 豊富なネットワーク機能
  - TCP/IPの実装（BSD UNIX）⇒ インターネットの礎を築く
- 高い移植性，優れたパフォーマンス
- セキュリティ，安定性，堅牢性に優れた，完成度の高いOS

代表的なOS：UNIX（2）

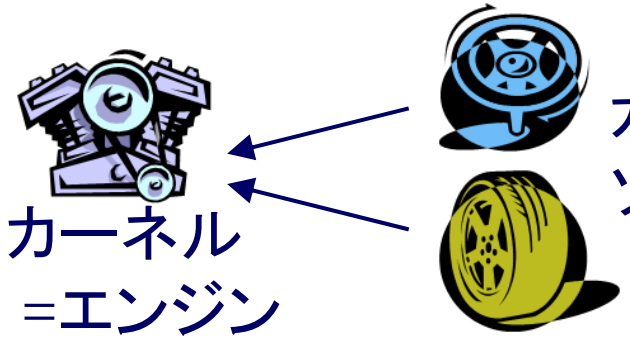
歴史的経緯

- 初期バージョンの開発は，AT&T（ベル研究所）が主体
  - 独占禁止法の関係で，当初は比較的自由に配布可能
    ⇒ 急速な普及，発展を遂げる
- BSD UNIX...カリフォルニア大バークレー校
  - AT&T版をベースに，大規模な機能改善を加えたもの
  - 豊富なネットワーク機能の実現
- AT&Tが分割され，UNIX の商用利用が可能に
  ⇒ 厳しいライセンス管理，不毛な法廷闘争
  ⇒ 開発は停滞，オープンソース UNIX へ

代表的なOS：UNIX（3）

オープンソースUNIX：

- UNIXと同様の機能，操作性を提供するOS
- ソースコードが公開されており，誰でも無料で利用可能
  - GNU/Linux...GNU のソフトウェア群と Linux カーネル



  - FreeBSD, OpenBSD, NetBSD etc.
  - Solaris (Sun Microsystems)

商標権の関係から，UNIXとは名乗っていない

代表的なOS：Mac OS

- アップルコンピュータ・マッキントッシュ搭載のOS
  - グラフィカルディスプレイ，マウスの利用を前提
  - 統一的操作性を持つグラフィカル・ユーザインタフェース
  - とくにユーザインタフェース面において，大きな影響

Macintosh 128K (1984)
http://www.tamabi.ac.jp/mc/collect/collect.htm

- 歴史的経緯
  - 元来は，ハードウェアに依存したOS (System 1～6)
  - System 7（96年）：ハードウェアとの分離，ネット対応
  - System 8, 9：システムの安定化
  - MacOS X：BSD UNIX ベース（NeXT社の技術を継承）

代表的なOS：Windows（1）

- Microsoft Windowsシリーズ
  - パーソナルコンピュータ，小型サーバ向けOS
  - PC用OSとして圧倒的なシェアを誇る
  - 他のOSの長所を吸収，巨大で複雑なOSに
  - 安定性，セキュリティの問題が指摘されることも
- 歴史的経緯
  - 出発点は MS-DOS：「ディスク」オペレーションシステム
    - 基本的なファイル管理機能のみ
    - 後に，UNIXライクな機能を拡張
  - 初期Windowsは，MS-DOSのユーザインタフェースを提供

代表的なOS：Windows（２）

歴史的経緯（続）

- Windows 3.1（1992，日本版は1993）
  - タスク管理，メモリ管理の充実，基本的なネット機能の装備
  - 日本語対応 ⇒ 日本のパソコン市場に大きな変化
- Windows 95 (1995)
  - ユーザインタフェースの改善
  - ネットワーク対応
- Windows 98, Windows ME...MS-DOSの技術制約が足枷に
- Windows NT, Windows 2000...MS-DOSに依存しないOS
- Windows XP...NT系列と95系列の融合
- Windows Vista...ユーザインタフェイスの刷新

前半部分のまとめ

- OSとは何か
  - 歴史
  - 役割
  - 機能
  - 代表的なOS

- パソコンで動いているOSは，生きた教材
  ⇒ あちこち探ってみると，新しい発見があるはず

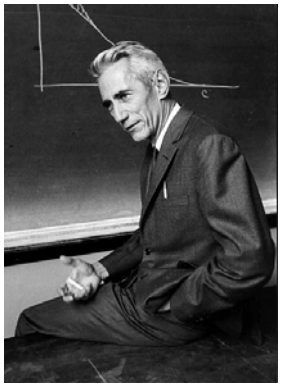
出欠確認 兼 ミニレポート

配布した用紙に以下を記入し，5限終了時に提出すること

- 所属，学生番号，氏名
- 5限開始時に着席している座席番号
  （背中部分に表示があるはず）
- 以下の課題への回答（数行程度でよい）

- 課題：
  - 携帯電話等，モバイル機器にもOSが搭載されている．
    これらのOSが，パソコン用OSと異なる点はどこか？

講義後半：情報理論入門



Claude E. Shannon (1916－2001)
情報理論の創始者

情報理論を支える3本の柱：

- 情報の定量化
  - 情報の量を，どのように量るか
- 情報源符号化
  - 情報源で発生する情報を，いかに効率よく表現するか
- 通信路符号化
  - 偶発的に発生する誤りから，いかに情報を保護するか

情報の定量化

基本的な考え方：

- あるニュースを知る…対象に関する不確かさが減少する
- ニュースの持つ情報量 ＝ 不確かさの減少量

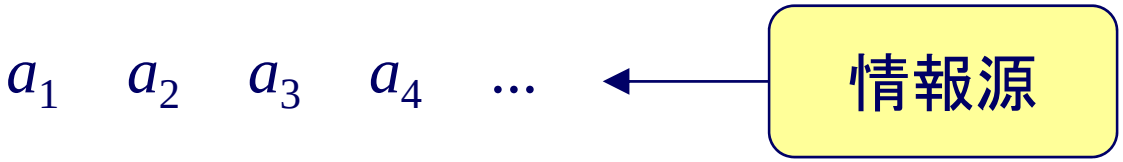
⇒「不確かさ」を定量化することが先決…エントロピーの導入

エントロピー

- もともとは，物理の専門用語
- 系の乱雑さを示す指標

情報源

- 情報源：
  - 情報を発生する存在
  - 発生する情報は，有限種類ある記号のどれか
  - 一単位時刻あたり一つの記号を発生する
  - 発生する記号の統計的性質（確率）はわかっている
  - 実際にどの記号が発生するかは，わからない



- 時刻 i に記号 a_i を発生する
- $a_i \in M$ (M は有限集合)

情報源と記憶

- サイコロを投げる，コインを投げる etc.
  - 記号の生起確率分布が，以前に発生した記号とは独立
    ⇒ 記憶の無い情報源
- 自然言語の文字，天気データ etc.
  - 記号の生起確率分布が，以前に発生した記号に依存
    ⇒記憶のある情報源
- 現実世界の多くの情報源は，記憶のある情報源
  - 記憶の無い情報源の組合せとして表現できる場合も多い
  - この講義では，記憶の無い情報源だけを考える

エントロピー

記憶の無い情報源 S が，以下の確率で各記号を発生するとき...

記号	a_1	a_2	...	a_n
確率	p_1	p_2	...	p_n

S のエントロピーは $H(S) = \sum_{i=1}^{n} - p_i \log_2 p_i$ （ビット）

- コイン投げの場合

記号	表	裏
確率	2^{-1}	2^{-1}

$2^{-1} = 1/2$

$$H(S) = -2^{-1}\log_2 2^{-1} - 2^{-1}\log_2 2^{-1} = 2^{-1} + 2^{-1} = 1 \text{ ビット}$$

エントロピーの例

- コインを2枚投げる場合

記号	表表	表裏	裏表	裏裏
確率	4^{-1}	4^{-1}	4^{-1}	4^{-1}

$$H(S) = 4(-4^{-1}\log_2 4^{-1}) = \log_2 4 = 2 \text{ ビット}$$

- サイコロ投げの場合

記号	1	2	3	4	5	6
確率	6^{-1}	6^{-1}	6^{-1}	6^{-1}	6^{-1}	6^{-1}

$$H(S) = 6(-6^{-1}\log_2 6^{-1}) = \log_2 6 = 2.585 \text{ ビット}$$

直感的には，「エントロピーが大きい＝予想が難しい」

「エントロピーが大きい＝発生情報に関する不確かさが大きい」

エントロピーの例（2）

- 奈良の天気

記号	晴	曇	雨
確率	0.4	0.5	0.1

$$H(S) = -0.4\log_2 0.4 - 0.5\log_2 0.5 - 0.1\log_2 0.1$$
$$-0.4(-1.322) - 0.5(-1.0) - 0.1(-3.322) = 1.361 \text{ ビット}$$

- サハラ砂漠の天気

記号	晴	曇	雨
確率	1.0	0	0

$H(S) = -1.0\log_2 1.0 = 0$ ビット

あいまいさなく，発生情報を予想可能

通報とエントロピー

- サッカー日本チームの成績

記号	勝	引分	負
確率	3^{-1}	3^{-1}	3^{-1}

$$H(S) = 3(-3^{-1}\log_2 3^{-1}) = \log_2 3 = 1.585 \text{ ビット}$$

- あるニュースを聞いた...「昨日の試合では，負けなかった」
  ⇒ 勝ったか，引き分けたか

記号	勝	引分	負
確率	2^{-1}	2^{-1}	0

$$H(S) = 2(-2^{-1}\log_2 2^{-1}) = \log_2 2 = 1 \text{ ビット}$$

- ニュースを聞いたことで，エントロピーが0.585ビット減少した

情報量

- ニュースを聞く前の試合結果...1.585ビット
- ニュースを聞いた後の試合結果...1.0ビット

ニュースを聞いたことで，試合結果に関する不確かさが減少した

⇒ ニュースが情報をもたらした

ニュースがもたらした情報の量 = 不確かさの減少量と定義する

「昨日の試合では，負けなかった」...0.585ビットの情報量を持つ

通報と情報量

- サッカーブラジルチームの成績

記号	勝	引分	負
確率	0.6	0.3	0.1

$$H(S) = -0.6\log_2 0.6 - 0.3\log_2 0.3 - 0.1\log_2 0.1$$
$$-0.6(-0.737) - 0.3(-1.737) - 0.1(-3.322) = 1.295 \text{ ビット}$$

- ニュース「昨日の試合では，負けなかった」

記号	勝	引分	負
確率	2/3	1/3	0

$$H(S) = -\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3} = 0.918 \text{ ビット}$$

情報量は1.295–0.918=0.377ビット...同じ通報でも，情報量は違う

通報の順序と情報量（1）

- トランプのカードを，52枚の中から1枚選ぶ

$$H(S) = 52(-\frac{1}{52}\log_2\frac{1}{52}) = 5.70 \text{ ビット}$$

- ヒント1：選んだ札は赤い札である

$$H(S) = 26(-\frac{1}{26}\log_2\frac{1}{26}) = 4.70 \text{ ビット}$$

  - ヒント1の持つ情報量 = 5.70 – 4.70 = 1.00 ビット

- ヒント2：選んだ札は絵札である

$$H(S) = 6(-\frac{1}{6}\log_2\frac{1}{6}) = 2.58 \text{ ビット}$$

  - ヒント2の持つ情報量 = 4.70 – 2.58 = 2.12ビット

通報の順序と情報量（2）

- ヒント3：選んだ札は8以上である
  - ヒント2の後にヒント3をもらった場合
    - エントロピーは変化無し…ヒント3の情報量は 0 ビット
  - ヒント2の前にヒント3をもらった場合

$$H(S) = 52(-\frac{1}{52}\log_2\frac{1}{52}) = 5.70 \text{ ビット}$$

⇩ ヒント1…1.00ビット

$$H(S) = 26(-\frac{1}{26}\log_2\frac{1}{26}) = 4.70 \text{ ビット}$$

⇩ ヒント3…1.12ビット

$$H(S) = 12(-\frac{1}{12}\log_2\frac{1}{12}) = 3.58 \text{ ビット}$$

⇩ ヒント2…1.00ビット

$$H(S) = 6(-\frac{1}{6}\log_2\frac{1}{6}) = 2.58 \text{ ビット}$$

天気予報

- 100日間の天気予報Aとその結果の記録:
  - 予報が晴で，天気も晴...45日
  - 予報が晴で，天気は雨...15日
  - 予報が雨で，天気は晴...12日
  - 予報が雨で，天気も雨...28日

		予報A	
		晴	雨
実況	晴	45	12
	雨	15	28

- 同じ期間の天気予報B:
  - 予報が晴で，天気も晴...0日
  - 予報が晴で，天気は雨...43日
  - 予報が雨で，天気は晴...57日
  - 予報が雨で，天気も雨...0日

		予報B	
		晴	雨
実況	晴	0	57
	雨	43	0

天気予報の当たる確率

天気予報は，必ずしも正確でない

- 予報Aの当たる確率

  (45 + 28) / 100 = 73%

		予報A	
		晴	雨
実況	晴	45	12
	雨	15	28

- 予報Bの当たる確率

  (0 + 0) / 100 = 0%

  ...絶対はずれる天気予報

		予報B	
		晴	雨
実況	晴	0	57
	雨	43	0

予報Aのほうが役に立つ（情報量が大きい）か？

- 晴れる日，雨の日は，トータル57日と43日

$$H(S) = -0.57\log_2 0.57 - 0.43\log_2 0.43 = 0.986 \text{ ビット}$$

⇒ 予報が，このエントロピーをどれだけ減らせるか？

予報Aの情報量

		予報A	
		晴	雨
実況	晴	45	12
	雨	15	28

- 予報Aが晴のとき…
  - 本当に晴れ: 45 / 60 = 0.75
  - 雨が降る: 15 / 60 = 0.25
  - 「晴」という予報を知った後のエントロピー

    $H(S) = -0.75\log_2 0.75 - 0.25\log_2 0.25 = 0.811$ ビット
  - 「晴」という予報の持つ情報量は0.986 – 0.811 = 0.175ビット
- 「雨」という予報の持つ情報量は0.986 – 0.811 = 0.105ビット

- 予報Aの平均情報量は, 0.6・0.175 + 0.4・0.105 = 0.147 ビット

予報Bの情報量

- 予報Aが晴のとき...
  - 本当に晴れ: 0 / 43 = 0
  - 雨が降る: 43 / 43 = 1.00
  - 「晴」という予報を知った後のエントロピー

$$H(S) = -1.0\log_2 1.0 = 0 \text{ ビット}$$

  - 「晴」という予報の持つ情報量は0.986 – 0 = 0.986ビット
- 「雨」という予報の持つ情報量は0.986 – 0 = 0.986ビット

		予報B	
		晴	雨
実況	晴	0	57
	雨	43	0

- 予報Bの平均情報量は 0.986 ビット

⇒ 当たる確率は低くても，情報量は予報Aより大きい

⇒ 我々が直感的に理解している情報量と，よくマッチしている

情報源符号化定理

シャノンの情報源符号化定理

- エントロピー h の情報源から発生する記号は，一記号あたり h ビットで表現（符号化）することが可能である

- コイン投げ：エントロピーは1ビット
  - 「表」を0に，「裏」を1 に符号化すれば良い
- コイン2枚投げ：エントロピーは2ビット
  - 「表表」「表裏」「裏表」「裏裏」をそれぞれ，00, 01, 10, 11に
- サイコロ：エントロピーは2.585ビット
  - 1, 2, 3, 4, 5, 6 を，001, 010, 011, 100, 101, 110 に...

    ⇒ 平均3ビット？...もっと効率よく符号化できるはず

情報源符号化

情報源符号化：

- 情報源から発生する記号を，効率よく表現するための手法
- 情報圧縮，と呼ばれることも

情報源記号	符号語
晴	00
曇	010
雨	011
雪	1101

- コンパクトに表現するだけでなく，取り扱いやすさも必要

  ⇒ 瞬時復号可能性

瞬時復号可能性

瞬時復号可能性:

符号語を区切らず書いても，後戻り無しで復号できるという性質

情報源記号	C_1	C_2
晴	0	0
曇	10	01
雨	110	011
雪	111	111

C_1 は瞬時復号可能
C_2 は瞬時復号可能でない

晴，雪，雪，晴を符号化して送る

- C_1:01111110 ⇒ 0 | 111 | 111 | 0...すぐに符号語に区切れる
- C_2:01111110...7文字目まで受け取った段階では
  - 0 | 111 | 111 | か 01 | 111 | 11 か 011 | 111| 1 か不明
    晴 雪 雪 曇 雪 （雪?） 雨 雪（雪?）

瞬時符号となる条件

符号 C が瞬時符号となる ⇔

C のどの符号語も，C の他の符号語の接頭語になっていない

情報源記号	C_1	C_2
晴	0	0
曇	10	01
雨	110	011
雪	111	111

0 は 01, 011 の接頭語
01 は 011 の接頭語

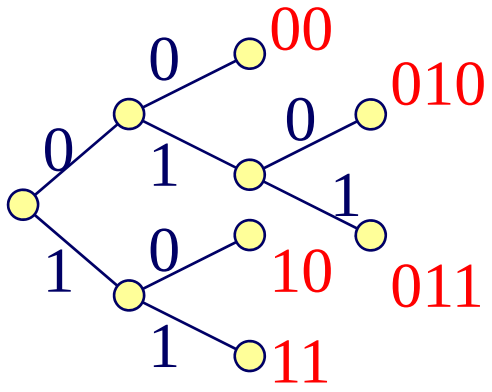
$C = \{w_1, w_2, ..., w_M\}$, $|w_i| = l_i$ とする．もし C が瞬時復号可能なら

$$2^{-l_1} + 2^{-l_2} + \cdots + 2^{-l_M} \leq 1$$（クラフトの不等式）

瞬時符号の作り方

- 符号木の葉を符号語に対応させる

- 符号木:
  - 二分木(子が最大2つの木構造)
  - 辺には 0 か 1 がラベル付け
  - 異なる子は異なるラベルで連結



平均符号長

- 確率 p_i で発生する記号に，符号語 w_i を対応付ける
- 符号語 w_i の長さを l_i とすると...
- 平均符号語長は... $\sum p_i l_i$

記号	確率	C_1	C_2
晴	0.4	0	111
曇	0.3	10	110
雨	0.2	110	10
雪	0.1	111	0

- C_1 の平均符号長は $0.4 \cdot 1 + 0.3 \cdot 2 + 0.2 \cdot 3 + 0.1 \cdot 3 = 1.9$
- C_2 の平均符号長は $0.4 \cdot 3 + 0.3 \cdot 3 + 0.2 \cdot 2 + 0.1 \cdot 1 = 2.6$

$\Rightarrow$ 平均符号長が短いほうが，効率よい（コンパクトな表現）

情報源符号化の目的

求められる情報源符号化法：

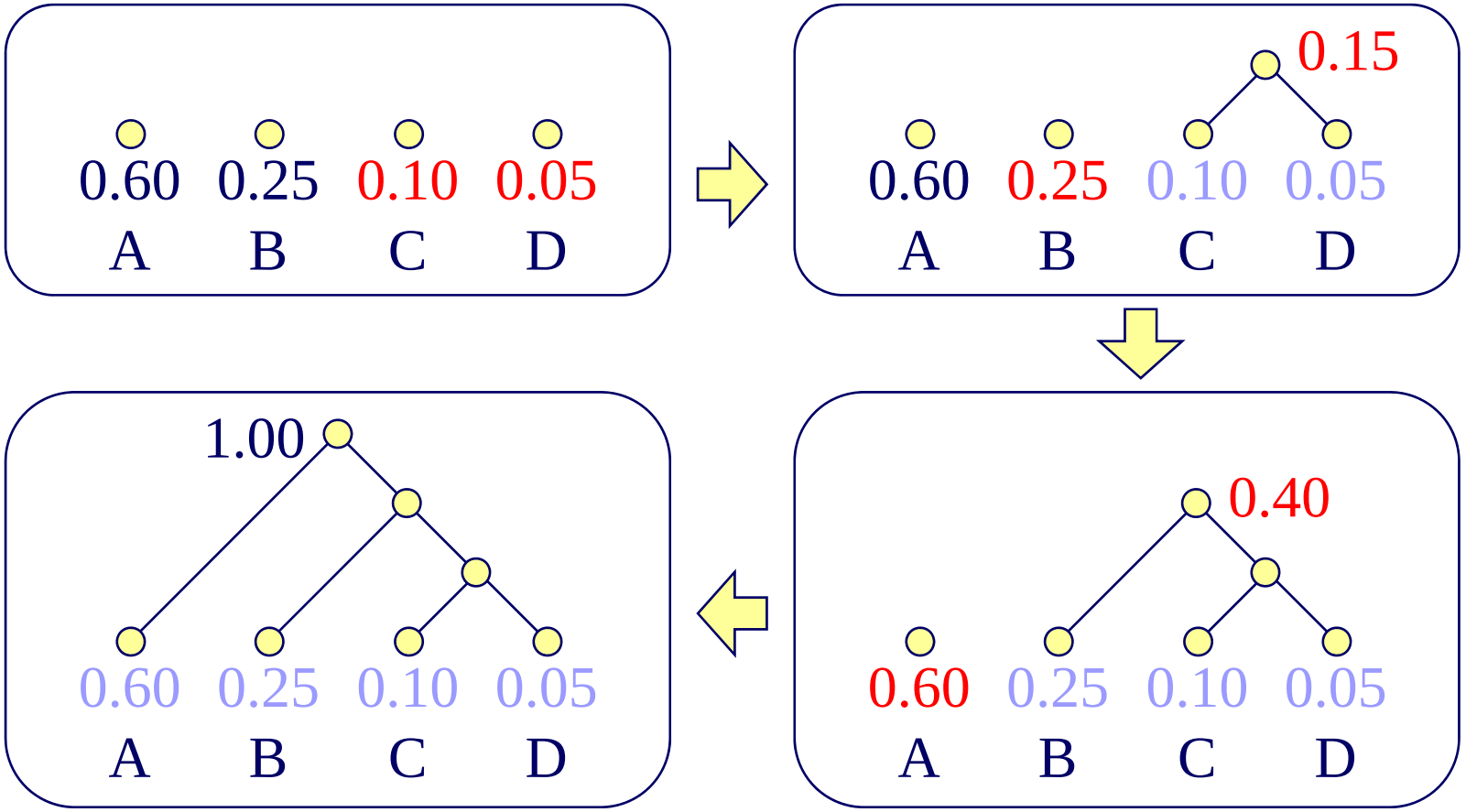
- 瞬時復号可能であること
- 平均符号長ができるだけ短いこと

⇒ ハフマン符号化法

- 符号木を，ボトムアップ的に構成していく
- 情報源記号と一対一に対応する葉節点を準備
- 葉節点には，対応する情報源記号の生起確率を付与
- 生起確率の小さな木（節点）を併合し，より大きな木を構成
- 併合された木の確率は，その部分木の確率の和
- 全体が一つの符号木になるまで，操作を繰り返す

ハフマン符号化法

記号	A	B	C	D
確率	0.60	0.25	0.10	0.05



ハフマン符号の平均符号長

記号	確率	C_1	C_2
A	0.60	00	0
B	0.25	01	10
C	0.10	10	110
D	0.05	11	111

- C_1 の平均符号長...2
- C_2 の平均符号長...0.60・1 + 0.25・2 + 0.10・3 + 0.05・3 = 1.60
- ハフマン符号 C_2 のほうが，効率がよい

- この情報源のエントロピー $H(S)$ = 1.49 ビット
- ハフマン符号にも，まだ改良の余地がある ⇒ ブロック化

ブロックハフマン符号

- 情報源記号を，いくつかまとめて（ブロック化して）符号化する

記号	確率	符号語
A	0.6	0
B	0.3	10
C	0.1	11

平均符号長 1.4 ビット

記号	確率	符号語
AA	0.36	0
AB	0.18	100
AC	0.06	1100
BA	0.18	101
BB	0.09	1110
BC	0.03	11110
CA	0.06	1101
CB	0.03	111110
CC	0.01	111111

平均符号長 2.67 ビット

⇒ 一記号あたり 1.335 ビット

ユニバーサル符号化，非可逆符号化

- ハフマン符号化...情報源記号の確率分布が，事前に必要
- 確率分布が（正確には）わからないケースも多い

⇒ ユニバーサル符号化法

- どのような情報源に対しても，そこそこ良い性能を発揮
- LZ77法 (lha, gzip, zoo, zip etc.)
- LZ78法 (compress, stuffit etc.)
- LZW法 (GIF, TIFF 等の画像フォーマット)

- 音声や画像等の情報...
  - 細部まで正確に再現できなくても，実害なし
  - 再現性を一部犠牲にして，高い圧縮率を⇒非可逆符号化

通信路符号化

偶発的に発生する誤りから，情報を保護したい

- 通信路において発生するノイズから情報を守る
- CD-ROM が傷ついても，中のデータは読めるようにする

- あくまでも偶発的な情報の損失だけを対象とする
  - 暗号，セキュリティ...意図的な妨害行為

- 基本的な考え方：
  - 誤りを検出し，訂正するための「余分な情報」を追加する

パリティ検査符号

パリティ記号の付加：

0と1で表現されたひとまとまりのデータの中に，

- 1が偶数個あれば0を
- 1が奇数個あれば1を

データの最後にくっつける操作

01001 ⇨ 010010

01011 ⇨ 010111

誤りを含む ⇦ 101001

- パリティ記号の付加されたデータは，偶数個の1を含む（偶パリティ符号）
- 1の個数が奇数個

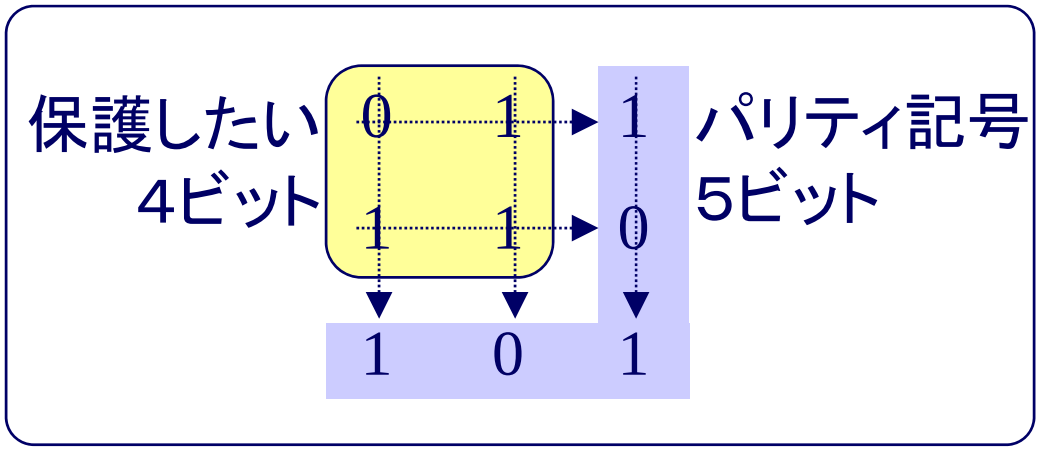
⇒ 誤りの影響を受けている

サーバ用メモリ等で利用されている

誤りの検出から訂正へ

- 偶パリティ符号そのものには，誤り訂正能力がない
- 偶パリティ符号を組み合わせれば，誤り訂正可能に

- 例：4ビットの情報を保護したい
  - 4ビットを長方形状に並べる
  - 水平方向，垂直方向にパリティ記号（5ビット）を付加

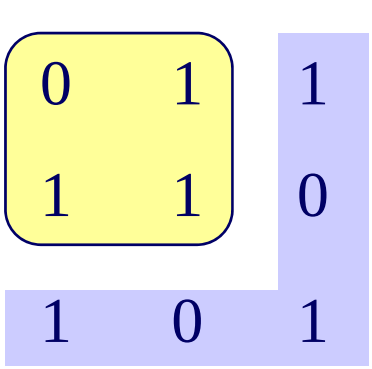


0111 ⇨ 011110101

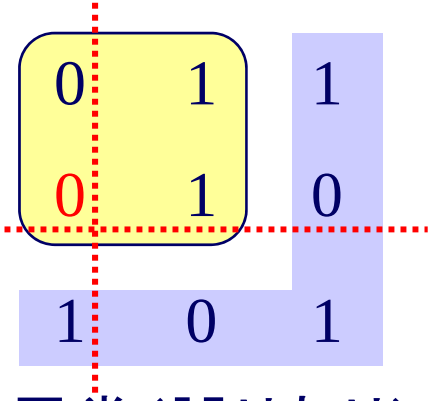
(9, 4) 符号
- 長さ9ビット
- 内4ビットが本来の情報
- 符号化率 4/9

誤り訂正の原理

- 前出の (9,4) 符号 ⇒ どの行，列とも，1は偶数個
- 1ビットの誤りが発生すると…
  - 誤りを含む行，列だけ，1が奇数個になる
  - 受信者は，誤りの発生位置を特定することが可能
    - 誤りは，異常のある行・列の交点に存在するはず



正常（誤り無し）

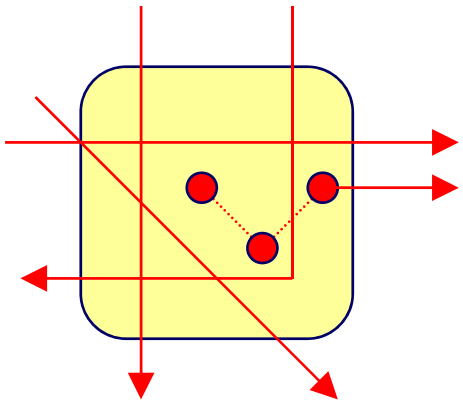


異常（誤りあり）

一般的な線形符号へ

前出の (9,4) 符号 ⇒ 行単位，列単位でパリティ記号を計算

- 二次元的な行，列にこだわる必要はない
- 斜めにたどる，途中で曲げる，スポット的に拾う etc…
- データビットの任意の組合せからパリティ記号を計算可能
- 組み合わせ方により，符号の性能が変化する



線形符号：

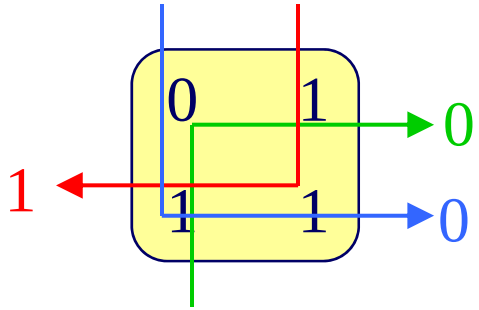
データビットの部分集合から
パリティ記号を定義する符号

いかにして良い組み合わせ方を探すか

⇒ 符号設計者の腕の見せ所

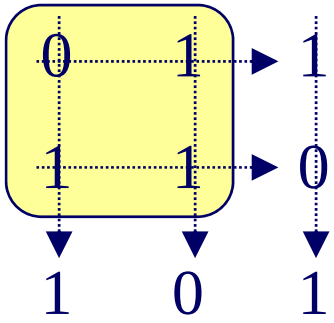
ハミング符号

- 代表的なブロック符号
- 符号語中に発生する1ビットの誤りを訂正可能
- 非常に効率が良い（完全符号）



0111 ⇨ 0111100

- (7,4)符号
- 効率 4/7



0111 ⇨ 0111110101

- (9,4)符号
- 効率 4/9

（どちらも，より大規模の符号も構成可能）

線形符号の特徴

- 符号化，誤りの検出，訂正とも，行列演算により実行可能
- 行列演算 ⇒ 単純な組合せ回路だけで実現可能
  ⇒ 高速に動作させるのに有利

- 規模の大きな線形符号
  - 巨大な行列に対する演算が必要となる
  - 組合せ回路の面積増大 ⇒ 遅延，消費電力，設計困難
  - スケーラビリティが悪い

もっと扱いやすい線形符号は？

巡回符号

- 符号化，復号を，シフトレジスタで実現できる線形符号

- 一般の線形符号に比べ，実装面で有利
  - シフトレジスタ利用による配線の簡単化
  - 符号化処理と復号処理で，回路の一部共有化が可能

代表的な巡回符号：

- BCH符号
- Reed-Solomon符号：
  - 多元符号（バイト単位での取り扱い等も容易）
  - CD, DVD等でも採用

畳込み符号

- 符号化装置を，有限状態機械として実現する方式
- 最尤復号が，比較的効率よく行える（ビタビアルゴリズム）
  - 統計的に，最も信頼度の高い復号法
- 軟判定復号の実現も容易
  - 復号精度を上げる技術
  - 受信値の信頼度を，0と1の二値でなく，より多段階で表現
- 多くの通信方式にて，畳込み符号が実際に用いられている

次世代誤り訂正符号

ターボ符号，LDPC符号

- 非常に規模の大きな符号
- 繰り返し計算を行うことにより，誤りの推測・訂正を行う
- サイバネティクス，機械学習的アプローチ，と理解することも

後半部分まとめ

- 情報の定量化
  - 情報の量を，どのように量るか
- 情報源符号化
  - 情報源で発生する情報を，いかに効率よく表現するか
- 通信路符号化
  - 偶発的に発生する誤りから，いかに情報を保護するか

現代の情報システムを支える基盤となる理論体系

出欠確認 兼 ミニレポート

配布用紙に以下を記入し，退室時，ドア付近の箱に提出

- 所属，学生番号，氏名
- 5限開始時に着席していた座席番号
  （背中部分に表示があるはず）
- 以下の課題への回答（数行程度でよい）

- 課題：
  - 携帯電話等，モバイル機器にもOSが搭載されている．
    これらのOSが，パソコン用OSと異なる点はどこか？

バイオの学生　　物質創成の学生