

連絡事項

- レポートの提出先
 - 西8号館3階(E棟とW棟の接続部付近)のレポートボックスに「コンピュータサイエンス入門(V-3)」という場所を作りました
- 授業の Web ページ
 - <http://www.is.titech.ac.jp/~etsuya/lecture/cs/2007/> からアクセスしてください
 - 配布資料の PDF 版などを順次置く予定です
- 次回(10月18日)は演習棟に集合してください

本日の話題: モデリングと表現

- モデリング事例 (1): 算数の文章題
- デジタルデータと計算
- モデリング事例 (2): Google の PageRank
- 画像のデジタル表現
- 音のデジタル表現
- 数値のデジタル表現

復習：算数の文章題 (1/2)

問題

鶴と亀があわせて9匹います。足の数は合わせて24本です。鶴と亀はそれぞれ何匹いますか？

本当の答

鶴が6匹
亀が3匹

モデリング

方程式

$$\begin{aligned}T + K &= 9 \\ 2T + 4K &= 24\end{aligned}$$

計算

解

$$\begin{aligned}T &= 6 \\ K &= 3\end{aligned}$$

復習：算数の文章題 (2/2)

- モデリングの過程では、問題を解くために必要な事項のみに注目し、他の情報は捨てる
 - 体長, 体重, 羽の色, 甲羅の模様などは無視
 - 「足の合計本数」と「全部で何匹か」のみに注目
 - 「足の合計本数」と「頭の合計数」に注目してもよい
- 問題の見方を変えた方がよい場合もある
 - そんなの数える暇があったら、最初から鶴と亀を別々に数えた方が簡単だろう！

データと計算

- 計算(computing)による問題解決
 - デジタルデータで問題を表現
 - デジタルデータに演算を適用することで、問題を解く

$$\begin{pmatrix} 1 & 1 \\ 2 & 4 \end{pmatrix} \begin{pmatrix} T \\ K \end{pmatrix} = \begin{pmatrix} 9 \\ 24 \end{pmatrix} \xrightarrow{\text{計算}} \begin{pmatrix} 6 \\ 3 \end{pmatrix}$$

PageRank (1/8)

- 問題

- Web ページをランク付けしたい

- 問題の背景

- サーチエンジンが検索結果をユーザに提示する時、表示の順番はどうすればよいか？

- 容易な側面

- Web ページ自体はもともとデジタル化されている

- 困難な側面

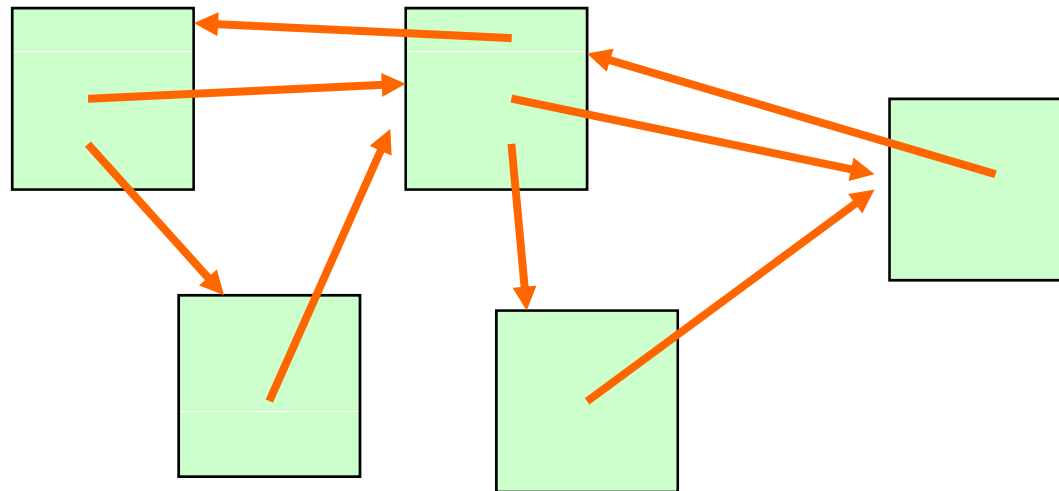
- ユーザから見たページの価値をいかに定義するか？
- 膨大な情報をいかに扱うか？

PageRank (2/8)

- PageRank アルゴリズム
 - Web ページのランク付けアルゴリズムの一つ
 - 論文: L. Page, S. Brin, R. Motwani, T. Winograd: The PageRank Citation Ranking: Bringing Order to the Web, 1998.
 - Google の基本アルゴリズムとして有名
 - 実際に使われているのはもっと複雑なもの

PageRank (3/8)

- Web のモデリング
 - ページとリンクの構造だけに注目
 - ページ内の文章等は無視
 - リンク (`...`) だけを見る



PageRank (4/8)

- Web のモデリング (cont.)
 - 時間的変化はとりあえず無視
 - ある瞬間のスナップショットを考える
 - 動的に生成されるページもとりあえず無視
 - 静的なページのみを考える

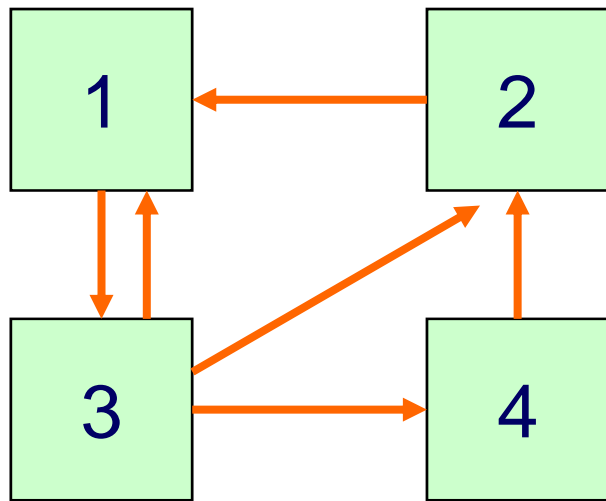
PageRank (5/8)

- ページのランク(価値)のモデリング
 - 直感的には以下のようなページはランクが高そう
 - 多くのページからリンクされている
 - ランクの高いページからリンクされている
 - この直感を以下のように定式化
 - $Rank(p)$: ページ p のランク
 - $Forward(p)$: p からリンクされたページの集合
 - $Back(p)$: p にリンクしているページの集合

$$Rank(p) = c \sum_{q \in Back(p)} \frac{Rank(q)}{|Forward(q)|}$$

PageRank (6/8)

- ページのランク(価値)のモデリング (cont.)



$$Rank(1) = c \left(Rank(2) + \frac{1}{3} Rank(3) \right)$$

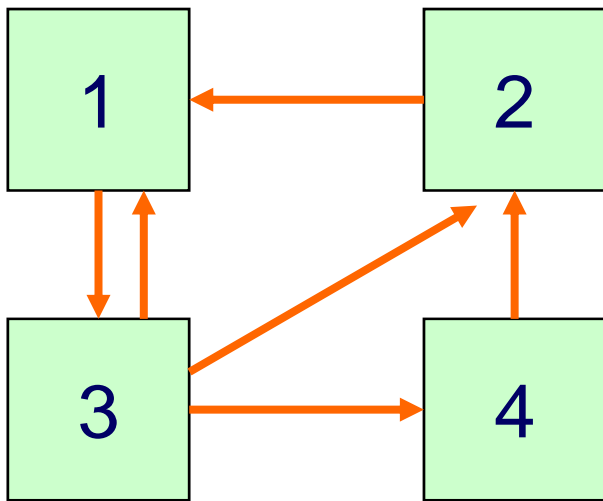
$$Rank(2) = c \left(\frac{1}{3} Rank(3) + Rank(4) \right)$$

$$Rank(3) = c Rank(1)$$

$$Rank(4) = \frac{c}{3} Rank(3)$$

PageRank (7/8)

- ページのランク(価値)のモデリング (cont.)
 - 数学的には, 固有ベクトルを求める問題
 - 見方を変えると, ランダムサーファアの訪問確率



$$\begin{pmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \end{pmatrix} = c \begin{pmatrix} 0 & 1 & 1/3 & 0 \\ 0 & 0 & 1/3 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1/3 & 0 \end{pmatrix} \begin{pmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \end{pmatrix}$$

$$A\vec{v} = \lambda\vec{v}$$

PageRank (8/8)

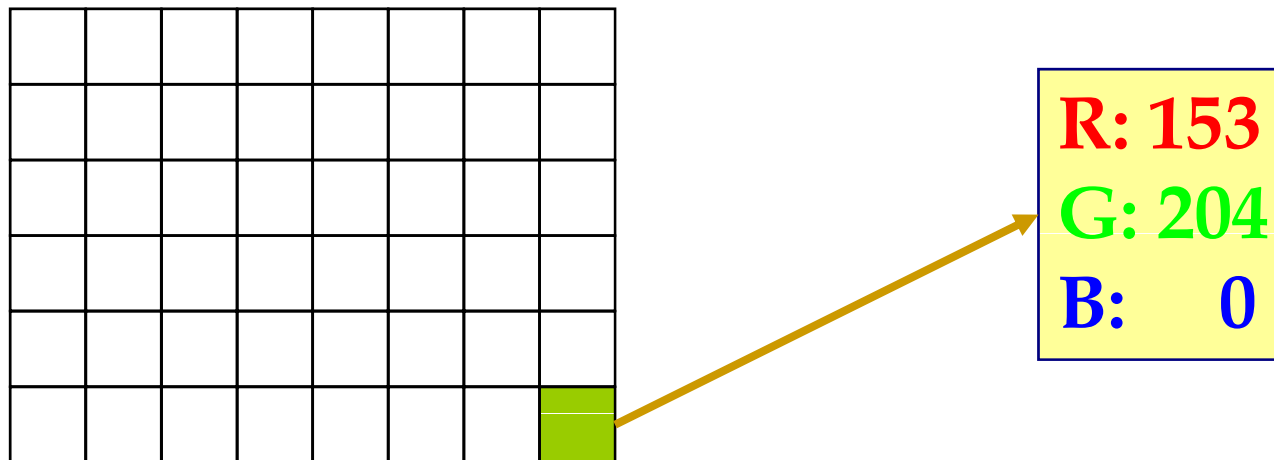
- 情報は最初からデジタル化されている
 - Web ページのクローリングは必要
- ランク(価値)の定義は難しい
 - 人間の主観にかかわる問題
- 思い切った情報の切捨て
 - 本文をすべて捨てるアプローチ
 - 「ページの価値は本文で決まる」とは考えない
- 集合知の利用
 - 個々人がリンクを張る行為を, そのページへの支持とみなし, 意見を集約
- 大量の情報を浅く処理し, ほどほどの精度を得る

モデリングの難しさ

- 厳密なモデリングは不可能な場合が多い
 - 表現すること自体が難しい
 - データ量を削減しないと、現在の技術では処理できない
- どこまで割り切るかの判断が難しい
 - 必要な近似のレベルや処理可能なデータ量等を考慮する必要がある

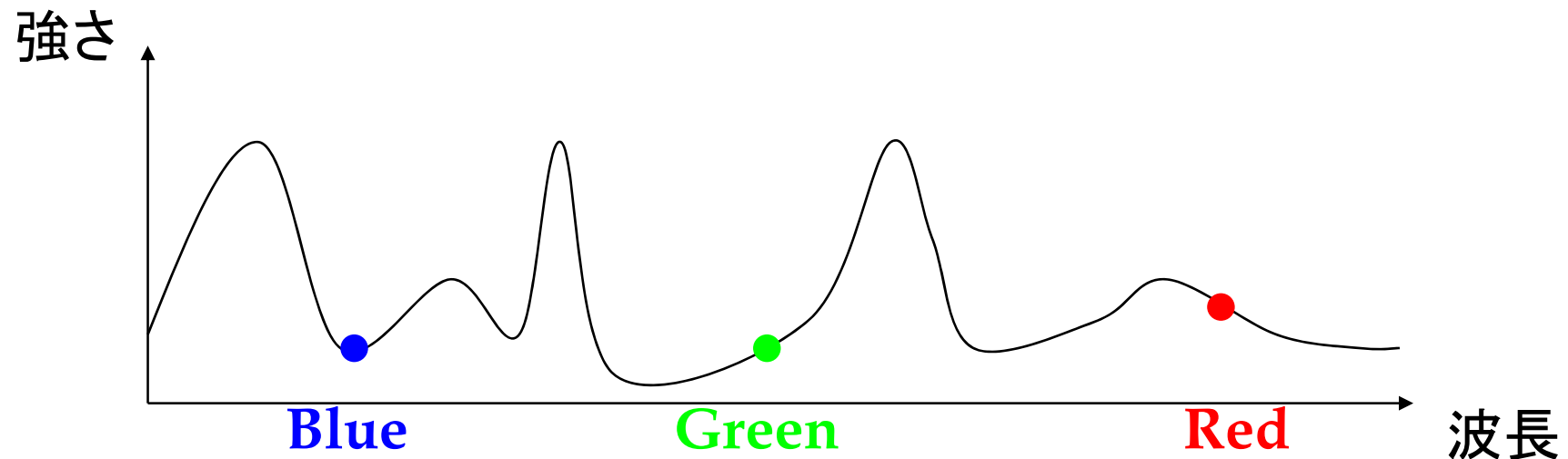
画像のデジタル表現 (1/4)

- コンピュータ画面のデジタル表現
 - 画面を画素(ピクセル)に分解
 - 1024×768, 1920×1200 など
 - 各画素の色を赤青緑(RGB)の三原色に分解
 - 各原色成分の強さを離散化し, 8ビット程度で表現



画像のデジタル表現 (2/4)

- 画素への分解
 - 空間的連続量の離散化手法として一般的なもの
 - cf. 流体や連続体のシミュレーション
- 三原色への分解
 - 可視光全体を3色に離散化してよいのか？



画像のデジタル表現 (3/4)

- 三原色への分解 (cont.)
 - これは, 人間の目の特性に合わせた表現方法

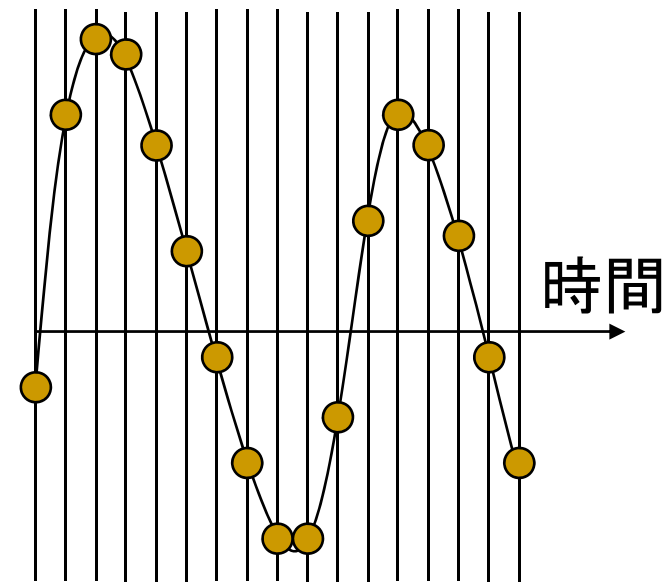
<http://en.wikipedia.org/wiki/Image:Cone-response.svg>
を参照してください.

画像のデジタル表現 (4/4)

- 各原色の強さの表現
 - 連続量を単純に離散化しているだけ
 - 8ビット(256階調)じゃ足りないという話はある
- 圧縮を行う場合が多い
 - 1画面分の静止画は、無圧縮なら数MB～十数MB
 - 1024×768, 24ビットカラーで2.25MB
 - 毎秒60フレームだと、毎秒100MBを超える
 - 圧縮時にも人間の目の特性を利用した方法が用いられる
 - コンピュータで解析するには適さないこともある

音のデジタル表現

- サンプリング + 離散化が基本
 - 細かい時間間隔で音のレベルを記録
 - 記録した音のレベルを離散化
 - e.g. CD の場合, 各チャンネルを, 44.1KHz でサンプリングし, 16ビットで離散化
- さらに圧縮することもある
 - e.g. MP3, AAC



画像 vs. 音

- 音も光も同じ波だが、デジタル化の方法は異なる
 - 音の方は素直に AD(Analog-to-Digital)変換
 - 光の方は素直でない
- 音と光では周波数がかなり違う
 - 可聴域は 20Hz~20KHz くらい
 - 可視光の周波数は、数百THz
- 実現可能性, コスト, 実用性などを考えて, 現在の
のような形になっている

数値のデジタル表現 (1/7)

- 実数値のデジタル表現
 - コンピュータ内部では、有限桁の2進小数を使う場合が多い
 - 一般には、離散化誤差が生じる
 - 2進数で小数点以下有限桁の数については、誤差が生じないこともある
 - 有理数なら、分数として表す方法もある

$$1.25 = 1 + \frac{1}{4} = 1.01_{(2)}$$

$$1.1 = 1 + \frac{1}{10} = 1.0001100110011\dots_{(2)}$$

数値のデジタル表現 (2/7)

- C言語のプログラムを実行して試してみる

```
#include <stdio.h>

int main() {
    printf("%.15f¥n", 1.1);
    printf("%.20f¥n", 1.1);
    printf("%.20f¥n", 1.25);
    return 0;
}
```

```
1.1000000000000000
1.100000000000000008882
1.2500000000000000000000
```

数値のデジタル表現 (3/7)

- 2進数と10進数は同等ではない
 - 整数値の表現力に差はないが、小数の表現力には差がある
- 金融業界では10進小数が必須
 - e.g. 0.01円(=1銭)を誤差なしで表現できないと外国為替の計算はできない
 - e.g. 利子の計算も誤差なしで行った後で丸める
- 物理学では、2進数と10進数の差をあまり気にしないことが多い
 - アナログな分野では、どうせ誤差は避けられない

数値のデジタル表現 (4/7)

- 自然数のデジタル表現
 - 単純に2進数で表すのが普通
 - 離散化誤差ないが、計算で桁あふれがありうる
 - ハードウェアは、8, 16, 32, 64ビットなどの(桁数に制限のある)整数の演算のみを実現することが多い

```
#include <stdio.h>
```

```
int main() {  
    printf("%d¥n", 256 * 256 * 256);  
    printf("%d¥n", 256 * 256 * 256 * 256);  
    return 0;  
}
```

```
16777216  
0
```

数値のデジタル表現 (5/7)

- ソフトウェアで対処して、桁数の多い整数や有理数を扱うことは可能
 - メモリは有限だが、64ビット(= 8バイト)よりはるかに多い

Mathematica による計算の例

```
In[1]:= 3^100
```

```
Out[1]= 515377520732011331036461129765621272702107522001
```

```
In[2]:= 3^100/6^50
```

```
717897987691852588770249
```

```
Out[2]=  $\frac{717897987691852588770249}{1125899906842624}$ 
```

数値のデジタル表現 (6/7)

- 整数の表現
 - 負の数の表現が必要
 - 2の補数表現が一般的
 - $-2^{n-1} \sim 2^{n-1}-1$ の範囲の整数値をnビットで表現可能
 - 稀ではあるが妙なことも起こる

10進	2進4桁	10進	2進4桁
0	0000	-1	1111
1	0001	-2	1110
2	0010	-3	1101
3	0011	-4	1100
4	0100	-5	1011
5	0101	-6	1010
6	0110	-7	1001
7	0111	-8	1000

数値のデジタル表現 (7/7)

```
#include <stdio.h>

int main() {
    int x = 256;
    printf("%d %d\n", x, -x);
    int y = x * x * x * 128;
    printf("%d %d\n", y, -y);
    return 0;
}
```

```
256 -256
-2147483648 -2147483648
```

情報のデジタル表現

- デジタル情報のデジタル表現は比較的簡単
 - e.g. 整数
 - それでも桁あふれや混合演算の問題はある
- アナログ情報のデジタル表現では誤差が生じる
 - 本当の実数 vs. コンピュータ内での小数表現
- 経済的表現 vs. 誤差の少ない表現
 - 少ないビット数ですめば経済的
 - 多くのビット数を使った方が誤差を減る
- 誤差が大きくても、実用上影響が少ないこともある
 - e.g. RGB