大学共同利用機関法人 情報・システム研究機構
国立情報学研究所
National Institute of Informatics

積み木のようにソフトウエアを作るには？

プログラミングの科学

胡 振江 (Zhenjiang Hu)
国立情報学研究所
2010年8月5日

プログラミングとは？

大学共同利用機関法人 情報・システム研究機構
国立情報学研究所
National Institute of Informatics

- 問題の仕様から、プログラムを作成すること。

問題の仕様

プログラミング

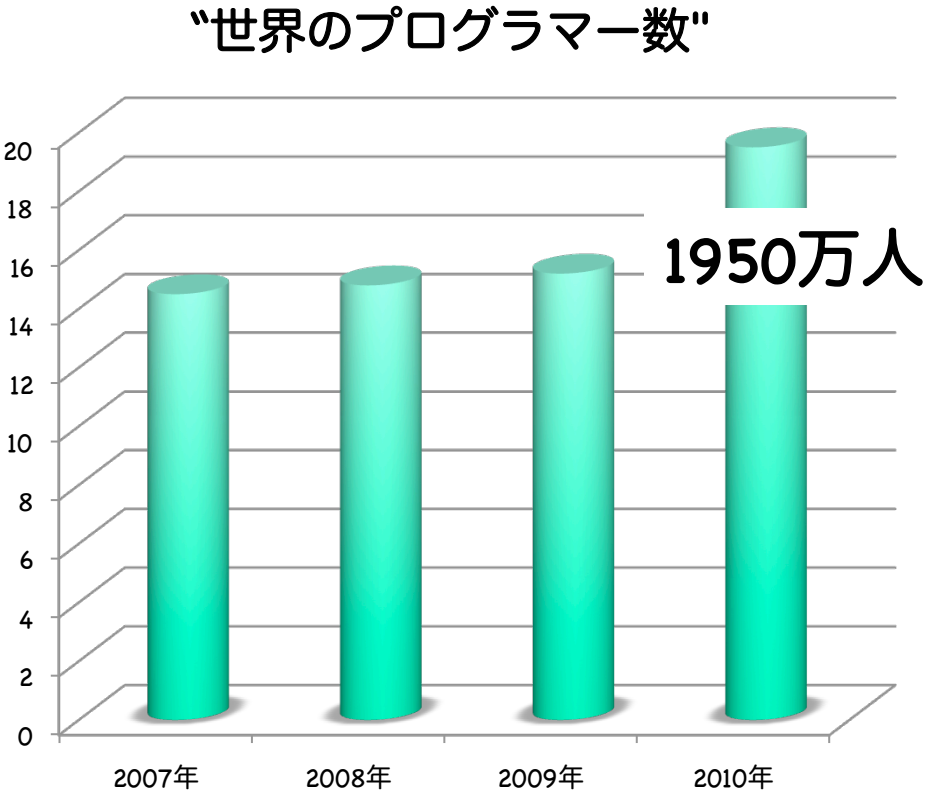
プログラム

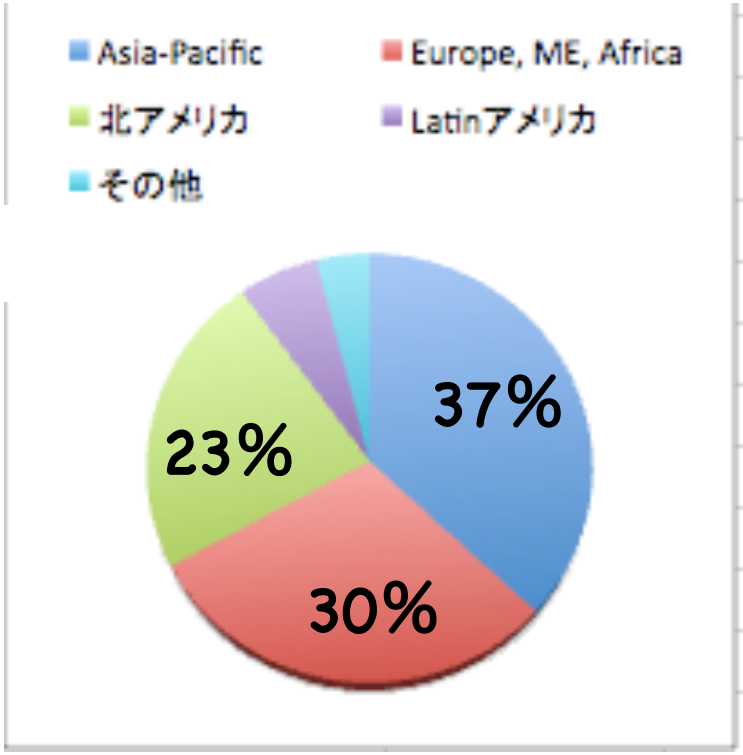
```
public Image grabFrameImage(){
    Image image;
    BufferToImage buffertoImage = null;
    FrameGrabbingControl fGrabbingControl=null;
    if(player!=null){
        fGrabbingControl=(FrameGrabbingControl)player.getControl("javax.media.control.FrameGrabbingControl");
    }
    Buffer buffer=fGrabbingControl.grabFrame();
    if(buffer!=null){
        buffertoImage=new BufferToImage((VideoFormat)buffer.getFormat());
    }
    image=buffertoImage.createImage(buffer);
    if(image!=null){
        return image;
    }
    return null;
}
```

プログラム実行

世界のプログラマー数

大学共同利用機関法人 情報・システム研究機構
国立情報学研究所
National Institute of Informatics





(From Infoworld)

プログラミングの例題

大学共同利用機関法人 情報・システム研究機構
国立情報学研究所
National Institute of Informatics

- 数列最大値問題

  数列の中で最大値を求める。

  [31,-41,59,26,-53,58,97,-93,-23,84] ➔ 97

- 最大部分列和問題

  数列の中で和が最大となる連続している部分列の和を求める。

  [31,-41,59,26,-53,58,97,-93,-23,84] ➔ 187

プログラミングは難しい？

大学共同利用機関法人 情報・システム研究機構
国立情報学研究所
National Institute of Informatics

(1) 正しいプログラムを作ること。

- **米国では年間595億ドル（国内総生産の0.6%）**
  （米国商務省国立標準技術研究所（NIST）調査，2002年）
- **日本にあてはめると社会的損失約3兆円**

航空管制システム障害
欠航215便，遅延1500便以上，足止め客30万人以上
2003年3月1日

東証システムダウン
東証1部，2部，マザーズなど全2520銘柄が停止
2005年11月1日

三菱東京UFJ―セブン銀ATM取引障害2万件
2008年5月12日

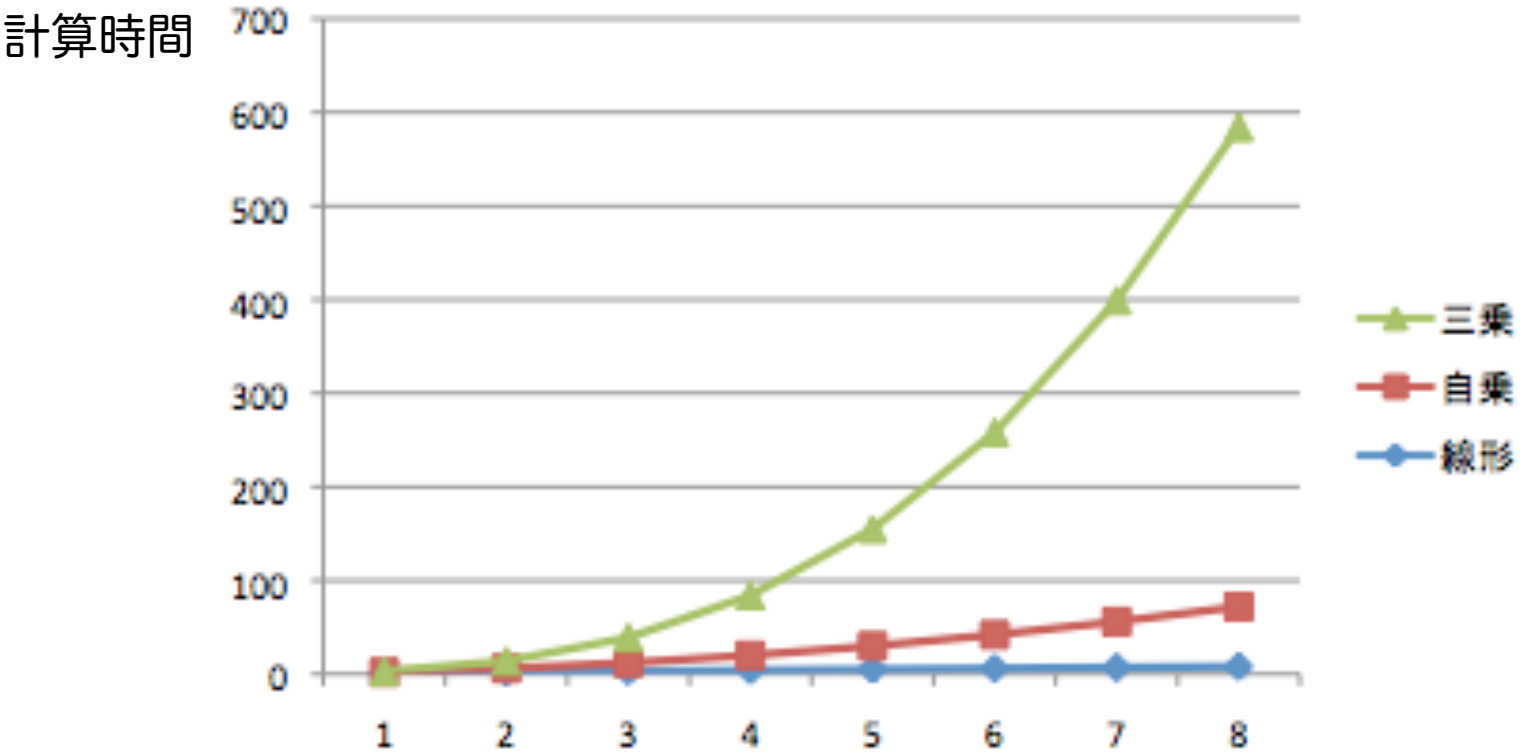
ATCのプログラムミス
300系新幹線計100編成で機能停止
2005年3月22日

（奈良先端大松本氏の講義資料より）

プログラミングは難しい？

大学共同利用機関法人 情報・システム研究機構
国立情報学研究所
National Institute of Informatics

(2) 計算資源を効率的に利用するプログラムを作ること。



プログラミングは難しい？

大学共同利用機関法人 情報・システム研究機構
国立情報学研究所
National Institute of Informatics

(3) 維持・変更しやすいプログラムを作ること。

- 読みやすいプログラムへ

  プログラムの美学！（東大名誉教授和田先生より）

- プログラムの部品化（モジュール化）

  部品の再利用、部品の交換によるシステムの更新

プログラミングは創造性の高い活動

大学共同利用機関法人 情報・システム研究機構
国立情報学研究所
National Institute of Informatics

プログラミングは人間の智慧が欠かせない。

- 誤解 1
  - 計算機が自動的にプログラムを生成する。

- 誤解 2

計算物理学者は1986年に・・・!

計算機プログラミングは、おおよそ頭を使わない退屈な仕事である。取るに足りないエラーに対してつまらない変更を加えたり、結果の表示を格好よくするために、何時間も何日も何週間もかかってやっている。 (NRC, Physics Through the 1990s, 1986)!

本講座では

大学共同利用機関法人 情報・システム研究機構
国立情報学研究所
National Institute of Informatics

正しくて、効率よくて、美しいプログラムの作成
人間の創造的な活動への支援

プログラミングの科学: プログラムの運算
(calculational programming)

正しさ：基本部品の組み合わせ
高効率：運算ルール（理論）
美しさ：関数によるプログラムの表現

鶴亀算問題：どう解決するか？

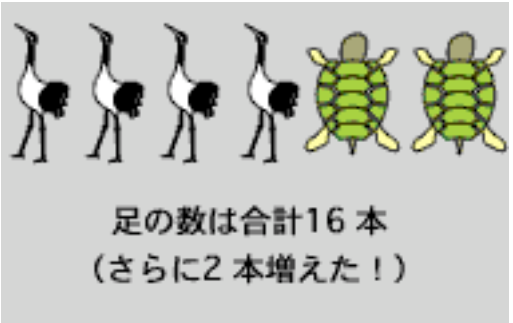
大学共同利用機関法人 情報・システム研究機構
国立情報学研究所
National Institute of Informatics

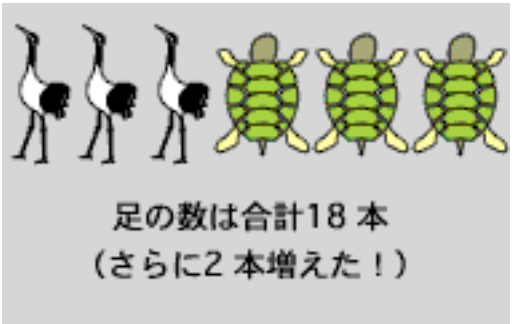
- 鶴と亀が合せて6 匹います。足の合計が2 0 本であった。 鶴と亀はそれぞれ何匹いますか？

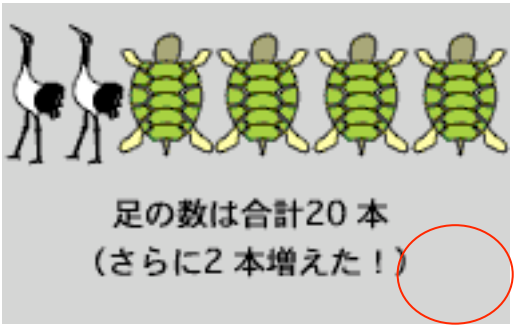
幼稚園の子供の解答

大学共同利用機関法人 情報・システム研究機構
国立情報学研究所
National Institute of Informatics

- 図を書いて全列挙する







（図は「学びの場」より）

小学生の解答

- 算術規則を応用する。

6 匹すべてが鶴とすると、足の数は6 ×2 ＝12 本。
足の数の和は20 だから、20−12 ＝8 で、
8 本足りない。
亀を1 匹増やすと2 本ずつ足が増えていくから、
8 ÷2 ＝4 匹の亀がいることになる。
鶴は6−4 ＝2匹。

中学生の解答

大学共同利用機関法人 情報・システム研究機構
国立情報学研究所
National Institute of Informatics

- 方程式で簡単に解ける。

亀が k 匹、鶴が t 匹とする。
次の方程式が簡単に作れる。

$$k + t = 6$$

$$4k + 2t = 20$$

方程式を解くと

$$k = 4$$

$$t = 2$$

がすぐわかる。

鶴亀算からプログラミング問題へ

- 同じ問題でも、使う「道具」によって、問題の難易度が変わる。
  - 方程式を使えば、算術問題を簡単に解ける。

- プログラミング問題を解くときの道具は？
  - プログラミングのための「方程式」は

次の問題はどうときますでしょうか？　（正しくて効率的なもの）

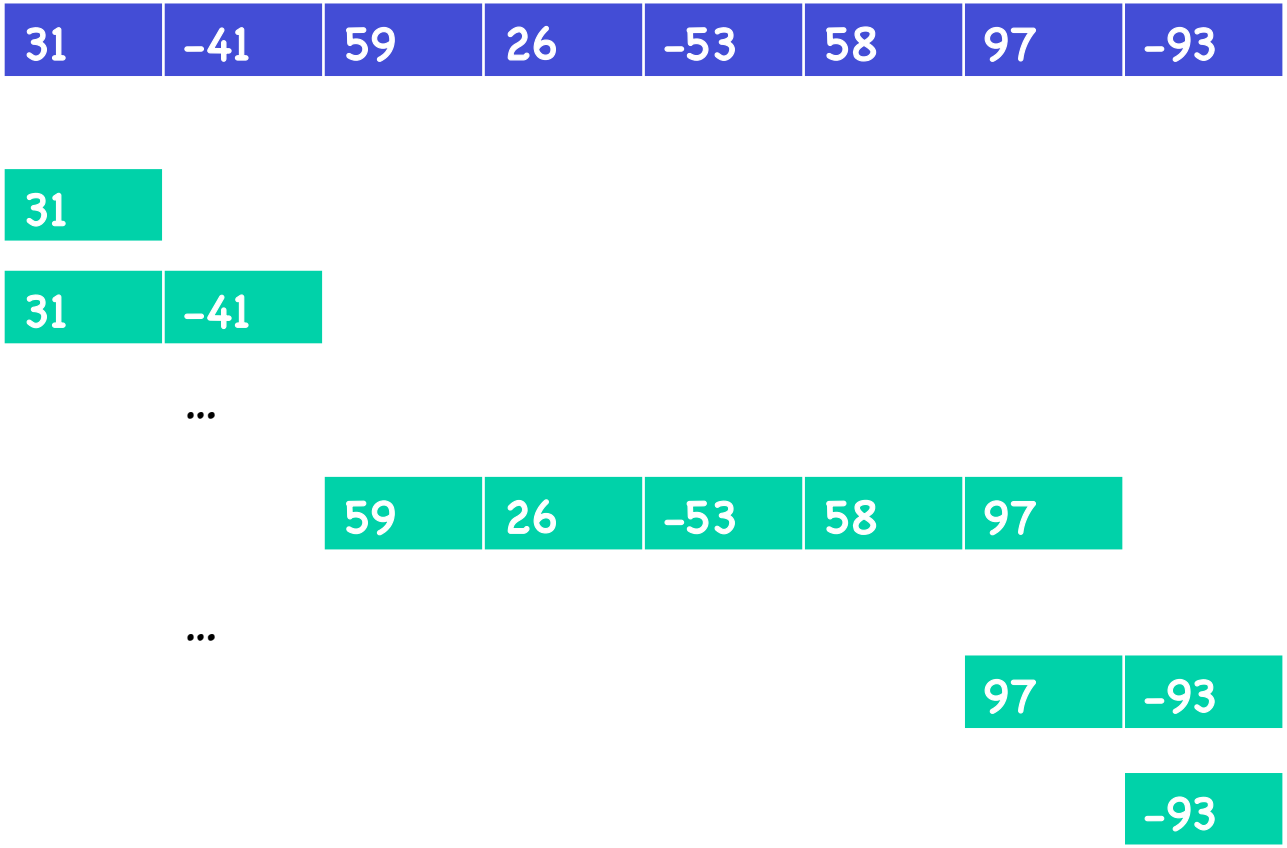
最大部分列和の問題
　数列の中で和が最大となる連続している部分列の和をもとめる。

[31,-41,59,26,-53,58,97,-93]　➔ 187

ナイブな解法

大学共同利用機関法人 情報・システム研究機構
国立情報学研究所
National Institute of Informatics

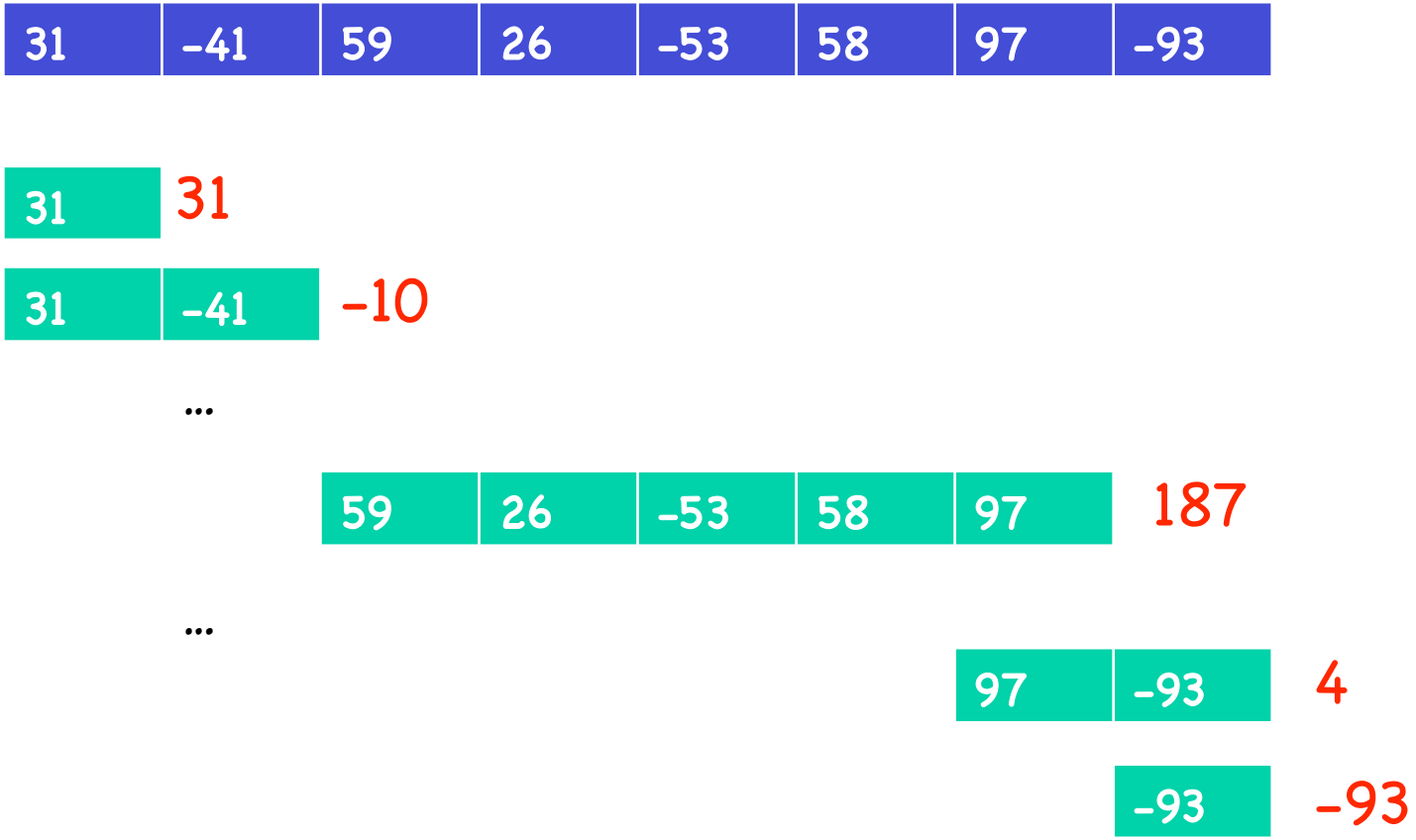
(1) すべての部分列を求める



ナイブな解法

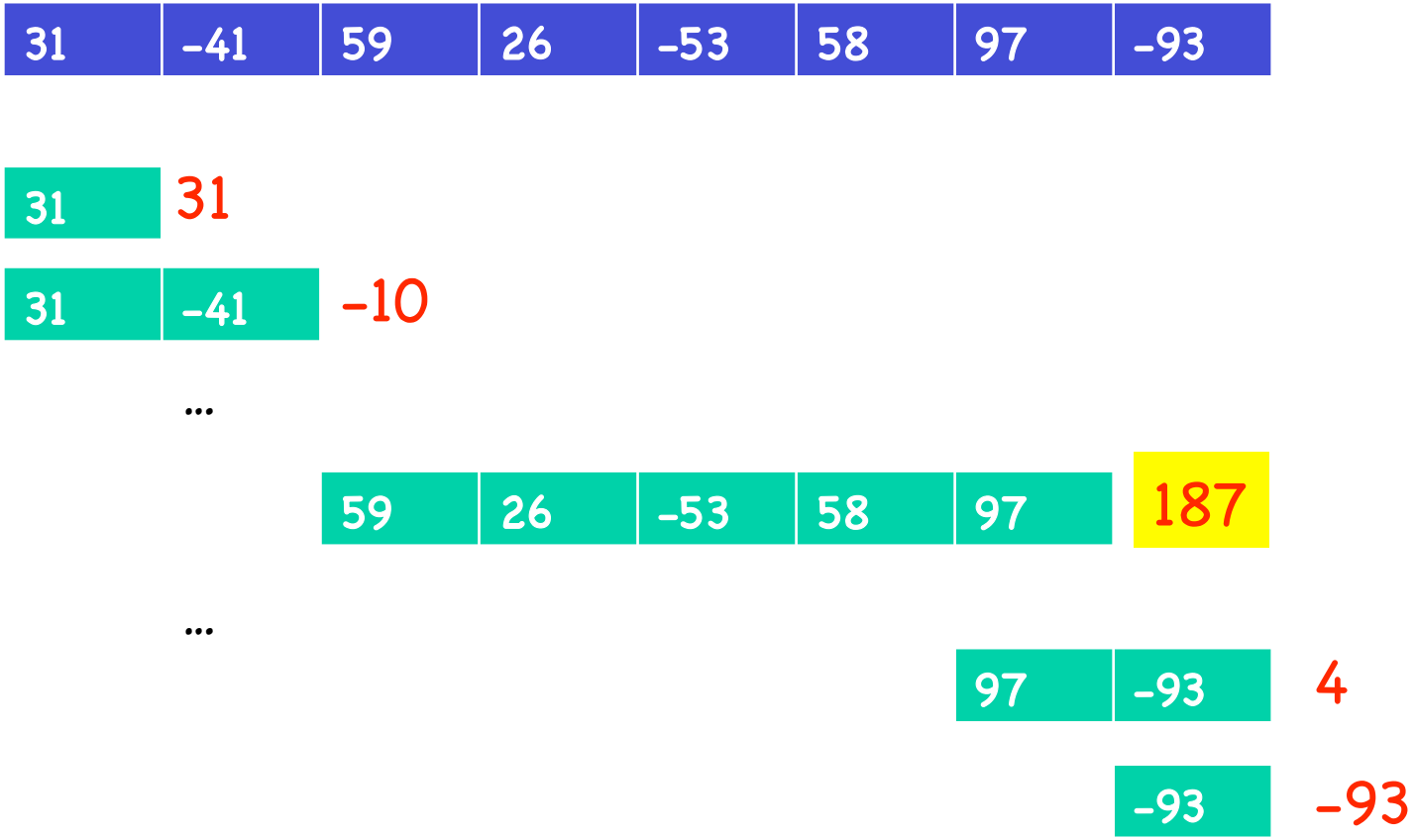
大学共同利用機関法人 情報・システム研究機構
国立情報学研究所
National Institute of Informatics

(2) 部分列ごとに和を求める



ナイブな解法

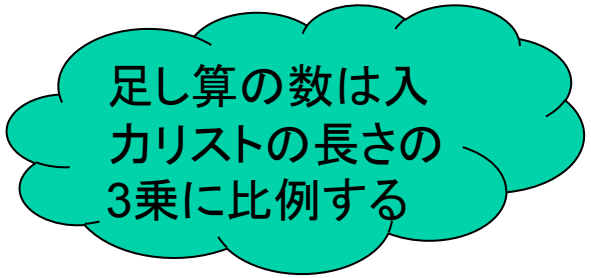
(3) 部分列ごとに和の最大値を求める



ナイブな解法

大学共同利用機関法人 情報・システム研究機構
国立情報学研究所
National Institute of Informatics

(1) すべての部分列を求める
(2) 部分列ごとの和を求める
(3) 部分列ごとの和の最大値を求める



全ての候補を生成する → それぞれの候補をテストし、結果を求める

正しそうであるが、効率が悪い。

最大部分列和問題の賢い解法

大学共同利用機関法人 情報・システム研究機構
国立情報学研究所
National Institute of Informatics

x		31	-41	59	26	-53	58	97	-93
s	0	31	0	59	85	32	90	187	94
mss	0	31	31	59	85	85	90	187	187

左から右へ順に
　s = （左のsとxの和）と０の大きい方の値
　mss = 左のmssとsの大きい方の値

この計算法は正しい？

正しくて効率のよいプログラムを作るには？

大学共同利用機関法人 情報・システム研究機構
国立情報学研究所
National Institute of Informatics

魚　和　熊掌　不可同時兼得　《孟子》

魚　和　熊掌　可不同時兼得

- 手法１
  (1) 効率のいいプログラムを作る
  (2) プログラムの正しさを検証する

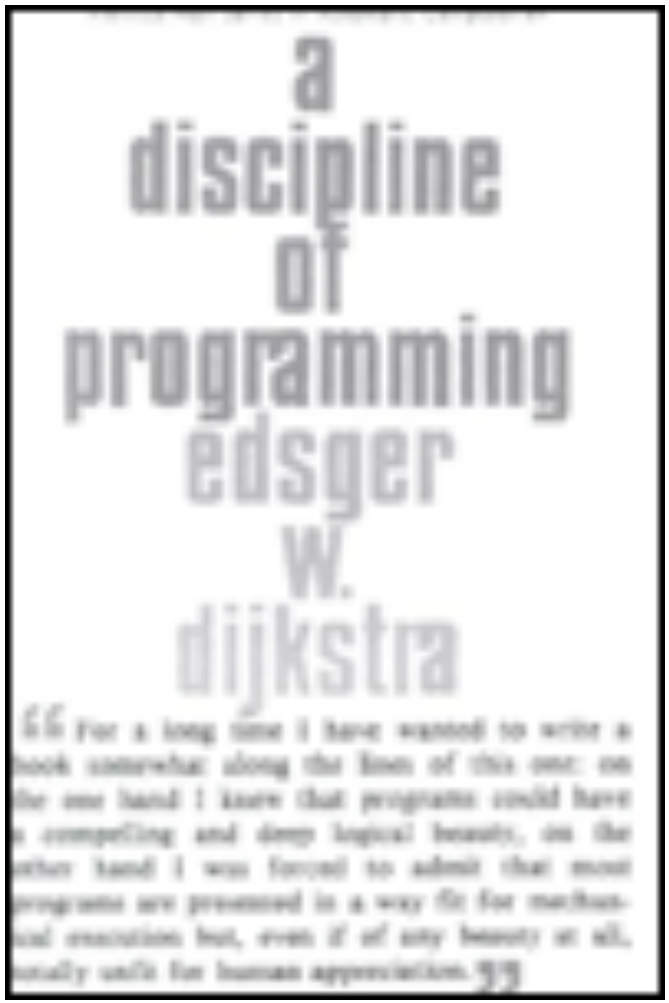
- 方法２
  (1)　正しいプログラムを作る
  (2)　プログラムの意味を保持しながら効率のよいものに変換する

プログラムングを科学する (1/2)

大学共同利用機関法人 情報・システム研究機構
国立情報学研究所
National Institute of Informatics

- E. W. Dijkstra
  (チューリング賞受賞)
  A discipline of Programming,
  Prentice-Hall, 1976

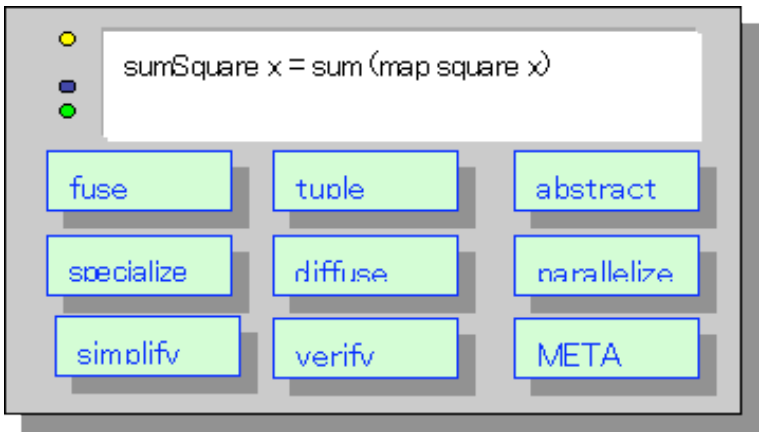
  プログラミングは単に
  経験によるものではなく、
  数学的な活動



プログラミングを科学する (2/2)

大学共同利用機関法人 情報・システム研究機構
国立情報学研究所
National Institute of Informatics

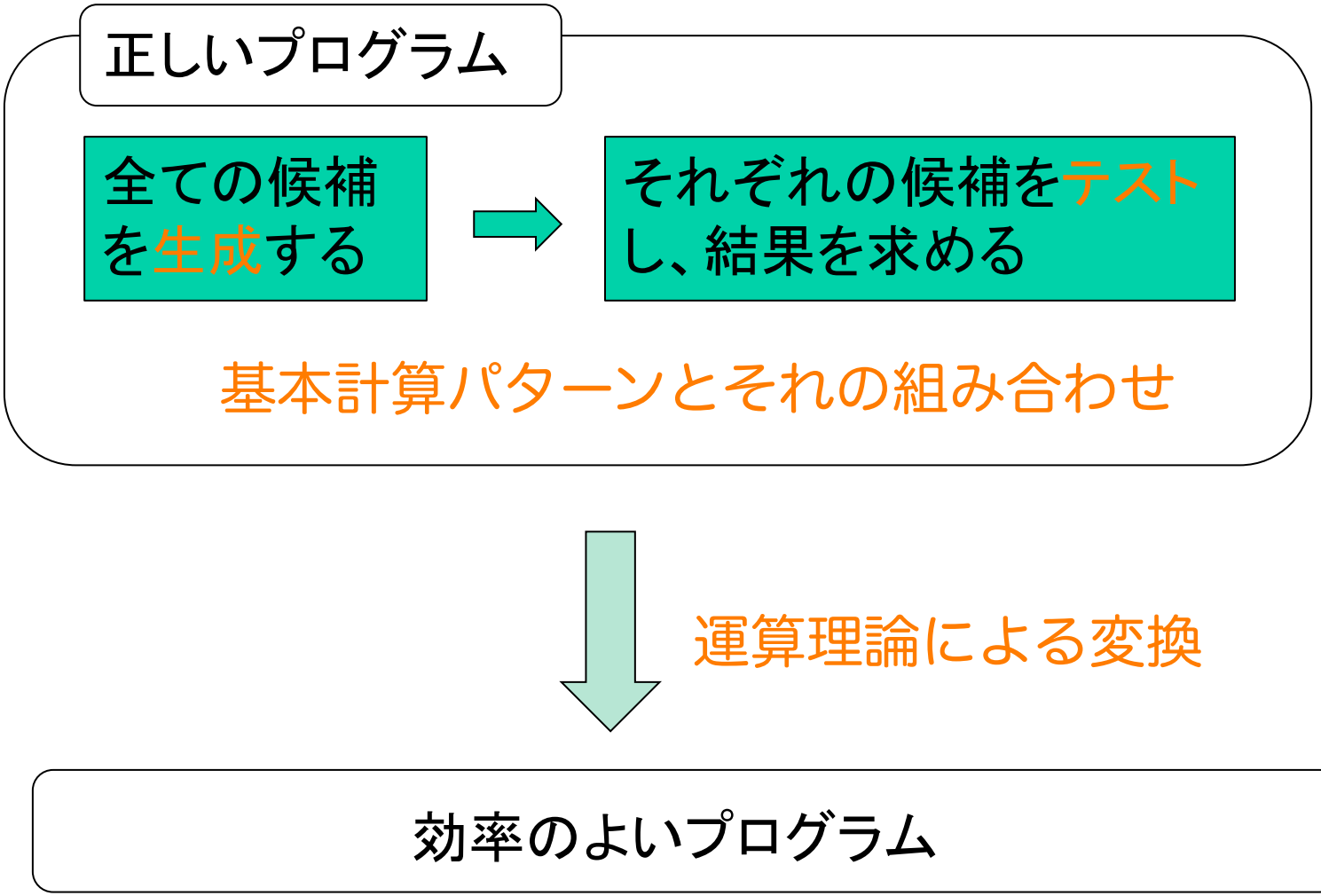
- 構成的アルゴリズム論
  - 80年代に、IFIP WG 2.1のメンバー (Bird, Meertens等)が中心に提案したプログラムcalculus
  - プログラムを運算することにより、プログラムを積み木のように基本ブロックから構成し、そのプログラムを効率的なものに変換する。



東京大学
武市研究室

運算によるプログラミング

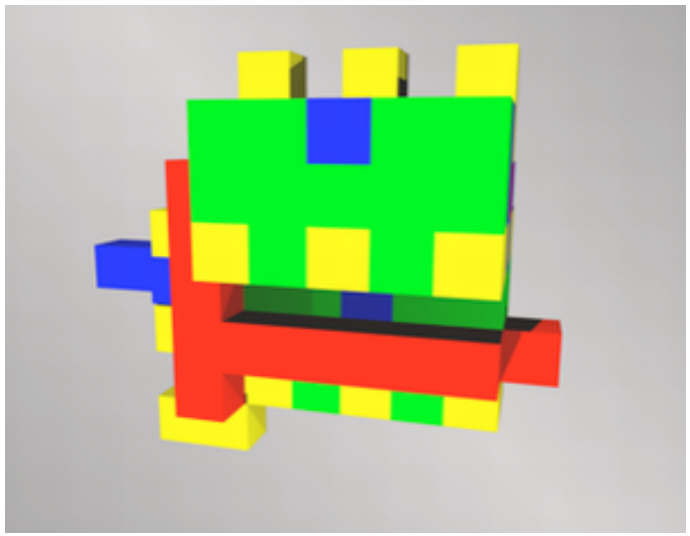
大学共同利用機関法人 情報・システム研究機構
国立情報学研究所
National Institute of Informatics

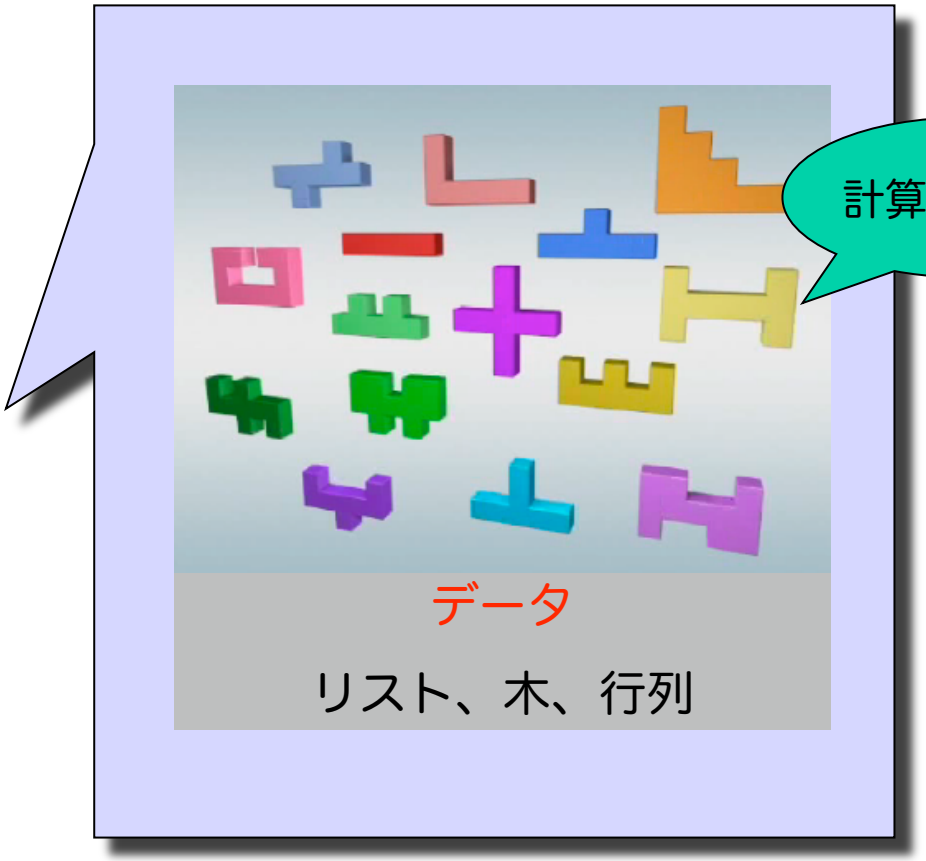


積み木のように …

大学共同利用機関法人 情報・システム研究機構
国立情報学研究所
National Institute of Informatics

プログラム





チャレンジ

- 基本スケルトンの選択
  - 多い：どれを選べばよいか分からなくなる
  - 少ない：複雑な問題に対処できない

- 組み合わせ手法
  - 系統的：
    - 基本スケルトンの組み合わせ手法
  - 効率的：
    - 融合変換：冗長なデータ生成の削除
    - 組化変換：同じデータの重複辿りの削除

リスト上の基本計算パターン

大学共同利用機関法人 情報・システム研究機構
国立情報学研究所
National Institute of Informatics

- リスト

  順序のある要素の並び

  [ ]

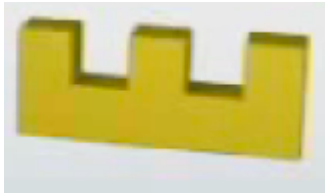
  [1,2,3,4]

  [[1,2], [], [3,4]]

- リスト上の計算パターン

map

fold

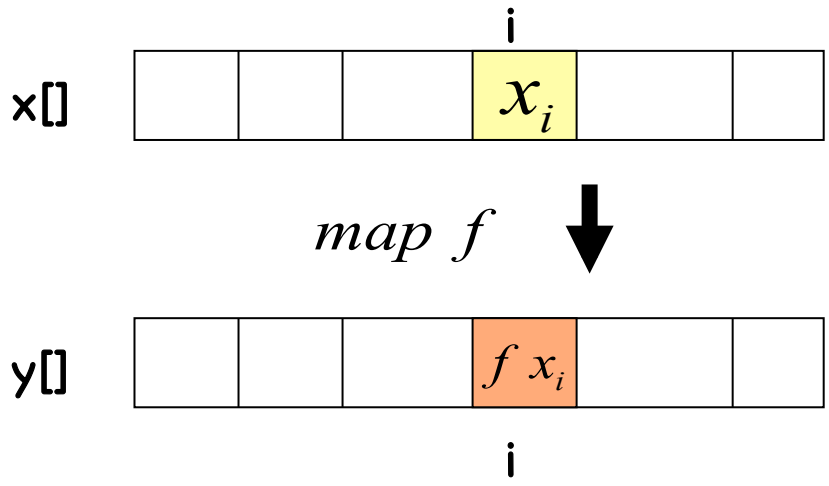


scan

zip

リスト上の基本計算パターン: map

大学共同利用機関法人 情報・システム研究機構
国立情報学研究所
National Institute of Informatics



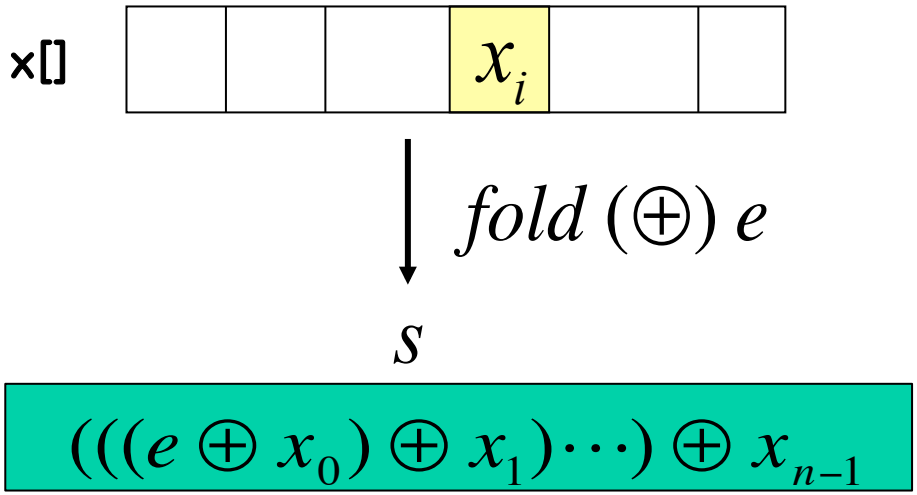

```
for (i=0;i<n;i++)
    y[i] = f(x[i]);
```

例：すべての要素を倍にする
　　すべての要素の逆数をもとめる

```
map double
map f where f(x) = 1/x
```

リスト上の基本計算パターン: fold

大学共同利用機関法人 情報・システム研究機構
国立情報学研究所
National Institute of Informatics

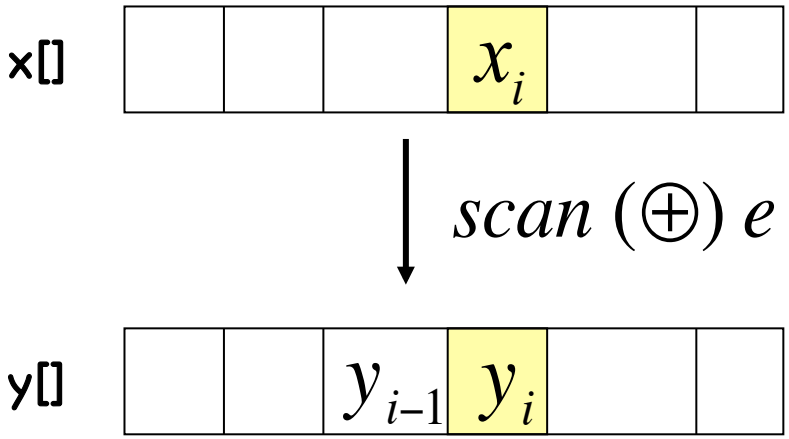



```
s=e;
for(i=0;i<n;i++)
    s = s ⊕ x[i];
```

例：要素の和をもとめる　　fold (+) 0

リスト上の基本計算パターン: scan

大学共同利用機関法人 情報・システム研究機構
国立情報学研究所
National Institute of Informatics

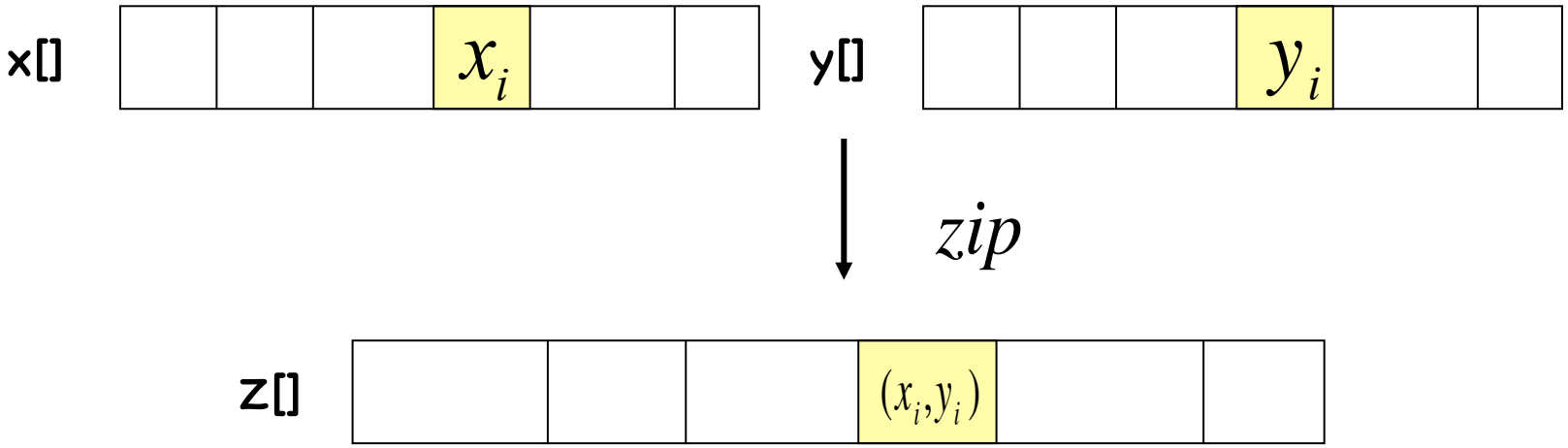



```
s=e; y[0]=e
for(i=1;i<n-1;i++)
    y[i] = y[i-1] ⊕ x[i];
```

例：接頭部分列和 (prefix-sum)を求める　scan (+) 0

リスト上の基本計算パターン: zip

大学共同利用機関法人 情報・システム研究機構
国立情報学研究所
National Institute of Informatics

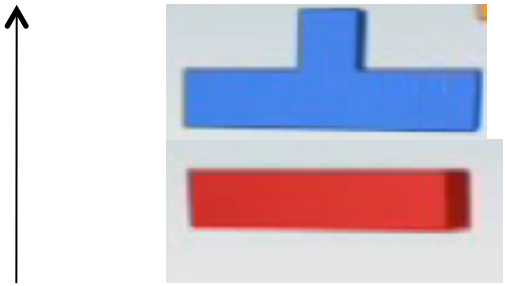



```
for(i=0;i<n;i++)
    z[i] = (x[i], y[i-1]);
```

プログラミング例

大学共同利用機関法人 情報・システム研究機構
国立情報学研究所
National Institute of Informatics

- 列の要素の自乗の和を求める



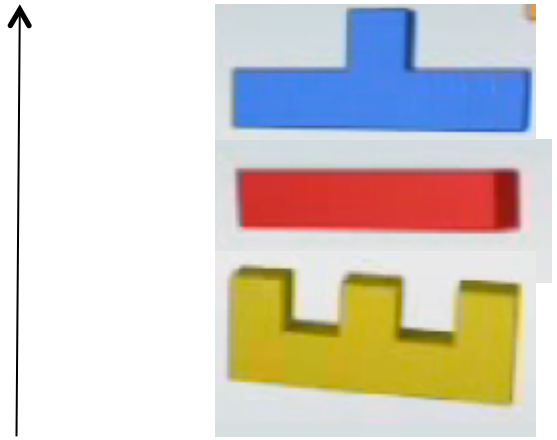
要素の和の計算

各要素の自乗の計算

プログラミング例

大学共同利用機関法人 情報・システム研究機構
国立情報学研究所
National Institute of Informatics

- 接頭部分列の和の最大値を求める



要素の最大値の計算

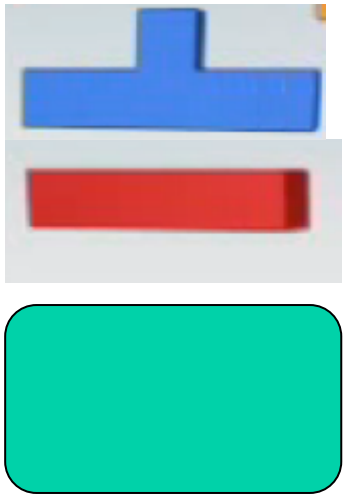
各部分列の和の計算

接頭部分列の列挙

プログラミング例

大学共同利用機関法人 情報・システム研究機構
国立情報学研究所
National Institute of Informatics

- 連続の部分列の和の最大値を求める



要素の最大値の計算

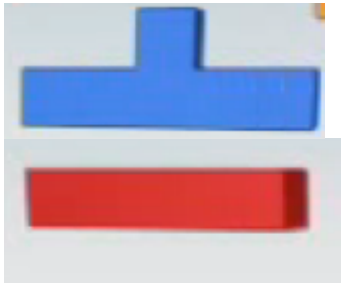
各部分列の和の計算

連続の部分列の列挙

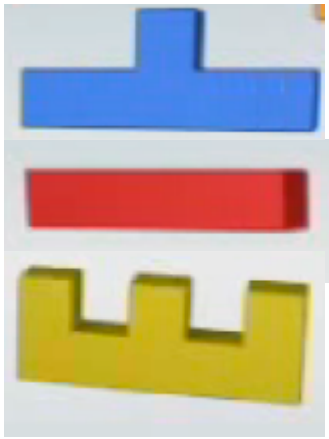
プログラミング例

大学共同利用機関法人 情報・システム研究機構
国立情報学研究所
National Institute of Informatics

- 連続の部分列の和の最大値を求める



連続の部分列の列挙



接尾リストの結合

各部分列の接尾リストの列挙

接頭部分列の列挙

その他の例

- 科学計算
  - 行列計算
  - QR分解
  - 多項式計算
  - N体問題

- 構造化文書処理
  - XML文書のタッグマチング問題
  - XQueryの並列化処理

- 最適化問題
  - 最大重み和問題
  - 動的計画法問題

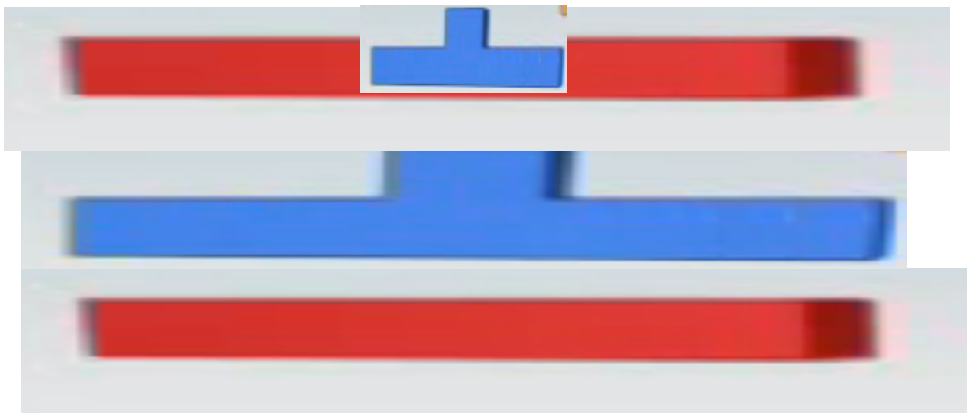
- 画像処理
  - 可視地点判定問題
  - レイトレーシング

基本パターンの表現力

大学共同利用機関法人 情報・システム研究機構
国立情報学研究所
National Institute of Informatics

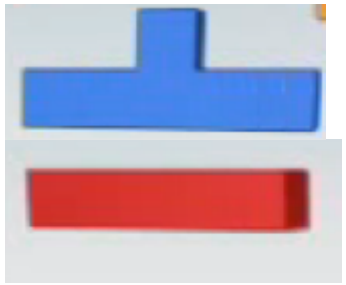
- リスト上の計算可能な関数はこれらの計算パターンの組み合わせで記述可能 (分離定理)

- Googleの計算：mapReduce



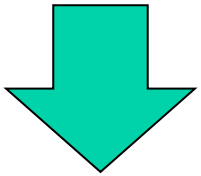
効率化：融合変換

大学共同利用機関法人 情報・システム研究機構
国立情報学研究所
National Institute of Informatics



要素の和の計算

各要素の自乗の計算



各要素の自乗の和の計算

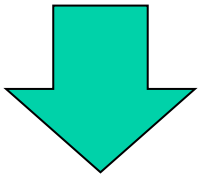
中間不必要なデータの生成を避ける

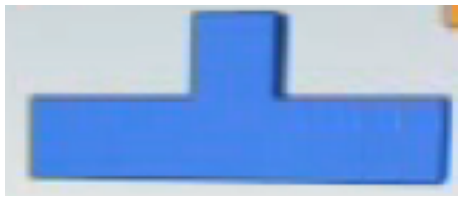
効率化：組化変換

大学共同利用機関法人 情報・システム研究機構
国立情報学研究所
National Institute of Informatics

要素の和の計算

要素の最大値の計算



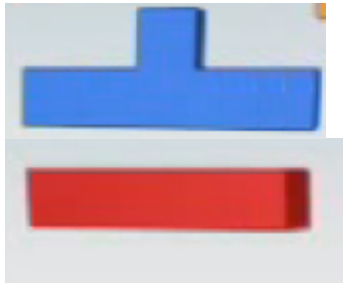


要素の和と最大値の計算

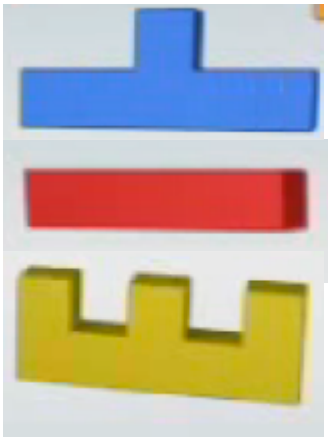
不必要なデータの辿りを避ける

連続の部分列和の問題

大学共同利用機関法人 情報・システム研究機構
国立情報学研究所
National Institute of Informatics



連続の部分列の列挙



接尾リストの結合

各部分列の接尾リストの列挙

接頭部分列の列挙

融合運算、組化運算

プログラムの運算

大学共同利用機関法人 情報・システム研究機構
国立情報学研究所
National Institute of Informatics

$$
\begin{aligned}
mss &= \{\text{定義}\}\\
&\quad max \cdot map\ sum \cdot \underline{segs}\\
&= \{segs \text{の定義}\}\\
&\quad max \cdot \underline{map\ sum \cdot concat} \cdot map\ tails \cdot inits\\
&= \{\text{map promotion}\}\\
&\quad \underline{max} \cdot concat \cdot map\ (map\ sum) \cdot map\ tails \cdot inits\\
&= \{max \text{の定義}\}\\
&\quad \underline{foldl\ (\sqcup)\ (-\infty) \cdot concat} \cdot map\ (map\ sum) \cdot map\ tails \cdot inits\\
&= \{\text{fold promotion}\}\\
&\quad max \cdot \underline{map\ max \cdot map\ (map\ sum) \cdot map\ tails} \cdot inits\\
&= \{\text{map 分配則}\}\\
&\quad max \cdot map\ (\underline{max} \cdot map\ \underline{sum} \cdot tails) \cdot inits\\
&= \{max,\ sum \text{の定義}\}\\
&\quad max \cdot map\ (\underline{foldl\ (\sqcup)\ (-\infty) \cdot map\ (foldl\ (+)\ 0) \cdot tails}) \cdot inits\\
&= \{\text{Horner 則: } x \odot y = (x + y) \sqcup 0\}\\
&\quad max \cdot \underline{map\ (foldl\ (\odot)\ 0) \cdot inits}\\
&= \{\text{scan lemma}\}\\
&\quad \underline{max} \cdot scanl\ (\odot)\ 0\\
&= \{max \text{の定義}\}\\
&\quad \underline{foldl\ (\sqcup)\ (-\infty) \cdot scanl\ (\odot)\ 0}\\
&= \{\text{fold-scan fusion: } (u, v) \oslash x = (u \sqcup w, w),\ w = (v + x) \sqcup 0\}\\
&\quad fst \cdot foldl\ (\oslash)\ (\underline{-\infty \sqcup 0}, 0)\\
&= \{-\infty \sqcup 0 = 0\}\\
&\quad fst \cdot foldl\ (\oslash)\ (0, 0)
\end{aligned}
$$

最大部分列和問題の賢い解法

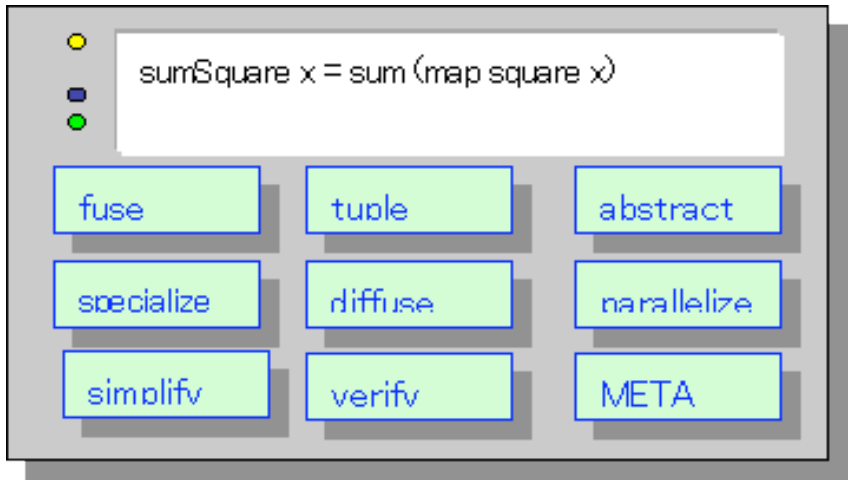
入力：数列 x[1..n]
出力：mss

```
mss = 0;
sum = 0;
for (i=1;i<n;i++) {
    sum = sum + x[i];
    if (sum < 0) sum = 0;
    if (mss < sum) mss = sum;
}
```

最適化のための運算ルールの開発

大学共同利用機関法人 情報・システム研究機構
国立情報学研究所
National Institute of Informatics

- 基本運算：融合運算、組化運算、蓄積運算
- 動的計画法の運算
- 貪欲アルゴリズムの運算
- 逆計算プログラムの運算
- …



Yicho運算システム
HYLOの融合運算器

東大武市・胡研究室
(1998-2007)

積み木のように並列プログラミング

大学共同利用機関法人 情報・システム研究機構
国立情報学研究所
National Institute of Informatics

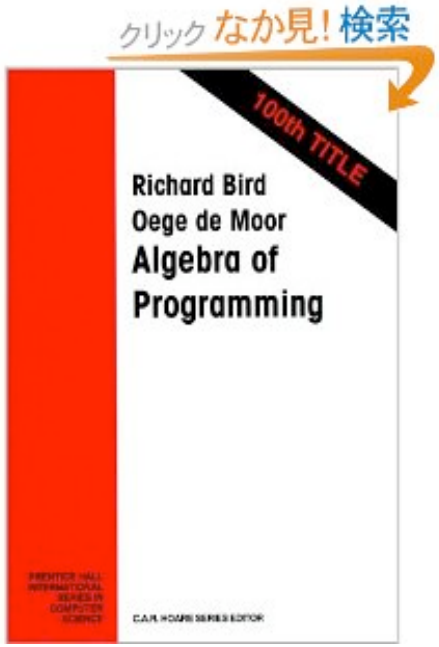
- JST さきがけ21 「機能と構成」 (2002-2005)

http://www.jst.go.jp/kisoken/presto/complete/kinou/kenkyu/hu/index.html

もっと勉強したいなら

大学共同利用機関法人 情報・システム研究機構
国立情報学研究所
National Institute of Informatics

- おすすめの本



- 講義資料
  - 東京大学「プログラムの数理 2008」
    http://research.nii.ac.jp/~hu/pub/teach/pm08/

まとめ

大学共同利用機関法人 情報・システム研究機構
国立情報学研究所
National Institute of Informatics

- プログラミングは科学的な活動！

Calculational Programming
- 基本計算パターンの合成によってプログラムを作る
- 運算理論(ルール)を用いて、プログラムを効率化する

- プログラマー：プログラミングのKUMON教育
  - 基本計算パターンの勉強
  - 代表的な運算ルールの勉強
- コンピュータ科学者：プログラミング理論の構築
  - 多くの運算ルールを開発する
  - 運算ルールの体系化

ご清聴ありがとうございました。