データ構造とアルゴリズム

第5回　自由課題作成

メディア情報コース
平居　悠（ひらい　ゆたか）

到達目標

- C言語を通じてデータ構造とアルゴリズムを理解する。
- 目的に応じて効率的なプログラムを自由に設計できる。

前回

第１回　構造体、共用体と線形リスト
第２回　探索とソート
第３回　メモリの動的確保
第４回　文字列の類似度計算、再帰、動的計画法
第５回　自由課題作成
第６回　自由課題作成
第７回　自由課題発表

前回の目標

2つの文字列の類似度を評価する
効率的なプログラムを作成できる
ようになる。

前回学んだこと

1. 編集距離
2. レーベンシュタイン距離を求めるアルゴリズム
3. レーベンシュタイン距離を計算するプログラム

編集距離 (edit distance)

2つの文字列がどれくらい異なっているかを評価するための計量。自然言語処理（人間の言葉をコンピュータで分析する技術）で頻繁に使われる。

最も広く使われる編集距離の定義は、情報理論学者のウラジミール・レーベンシュタインが提案した「**レーベンシュタイン距離**」である。

レーベンシュタイン距離

2つの文字列の間のレーベンシュタイン距離を計算するには、1つの文字列をもう1つの文字列に変換するのに必要な1文字の最小回数を求める。

レーベンシュタイン距離

実行できる編集の種類：

- 1文字の**挿入** (例：cat→cast)
- 1文字の**削除** (例：bait→bat)
- 1文字の**置換** (例：cut→cat)

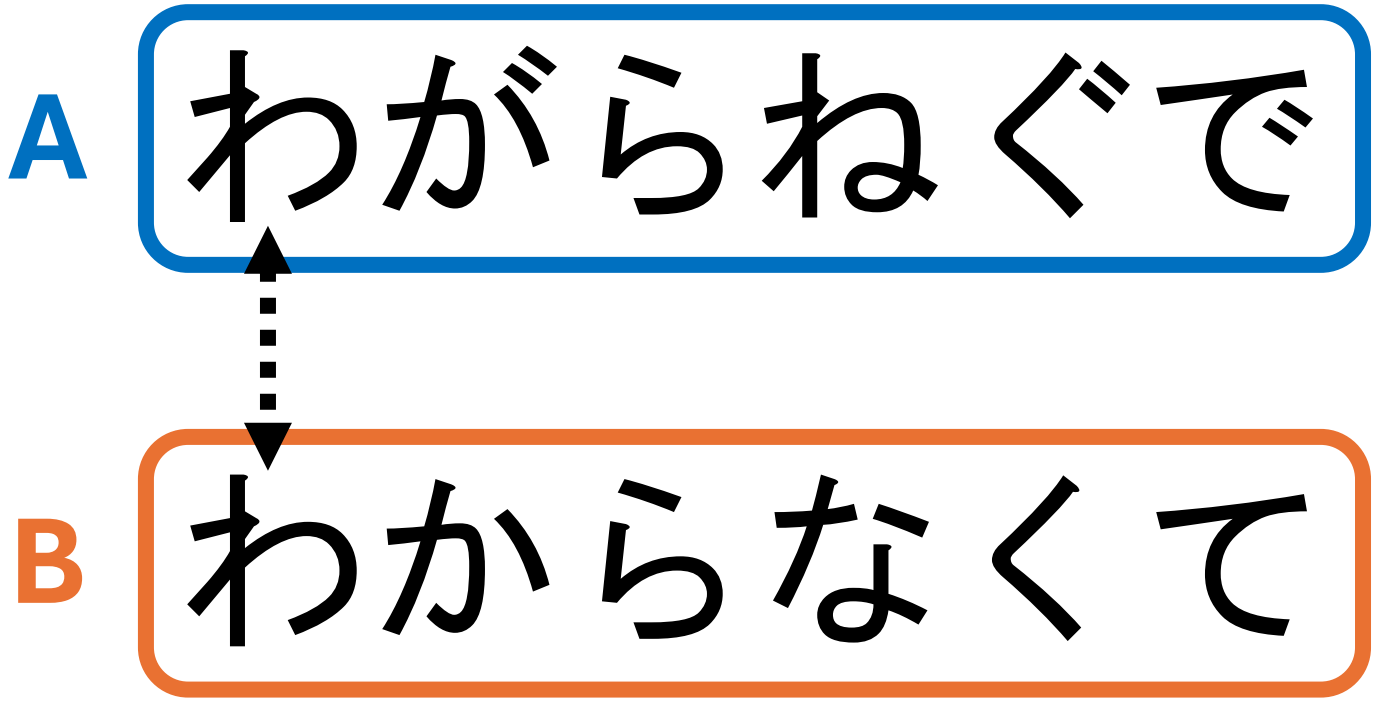
編集距離のアルゴリズム

①文字列AとBのどちらかが空なら、もう一方の長さを編集距離に足して終了



編集距離のアルゴリズム

②文字列AとBの1文字目を比べる



編集距離のアルゴリズム

③同じなら次の文字に進んで①へ戻る

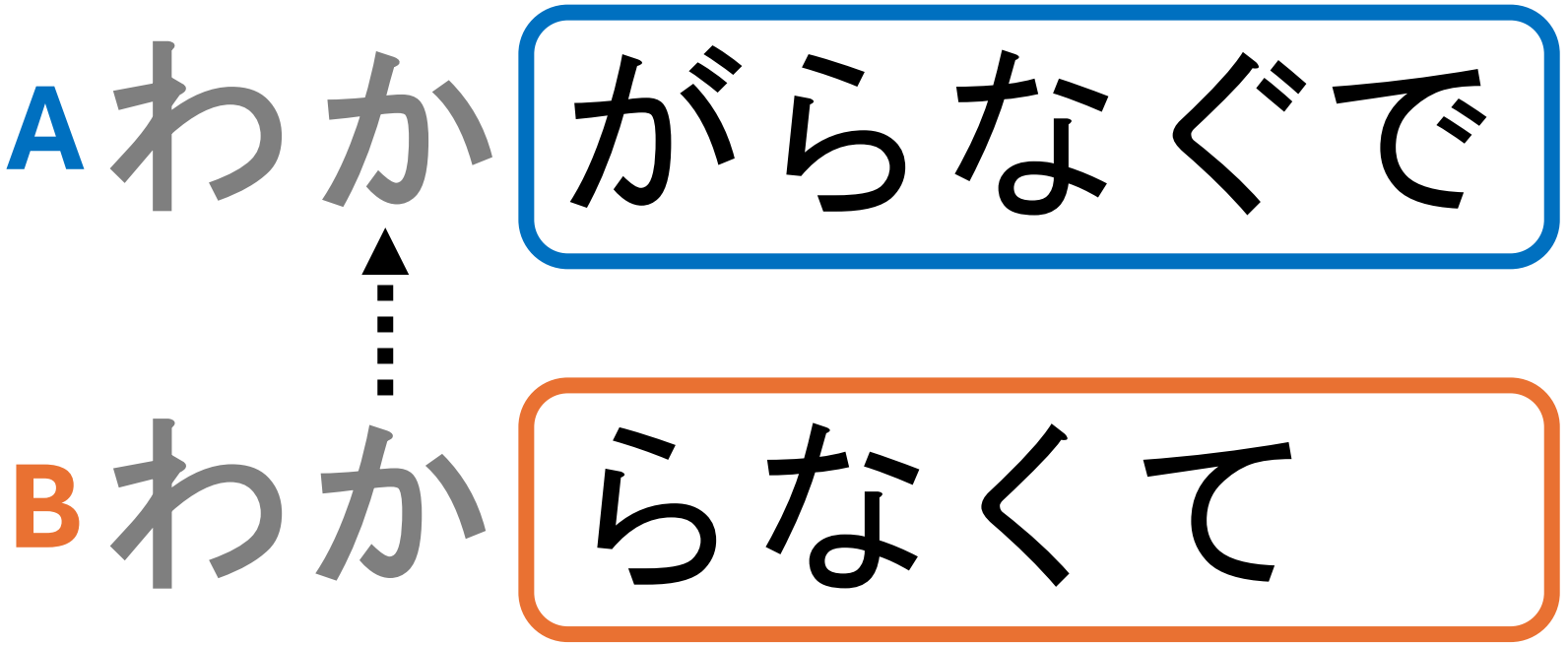
A わ がらねぐで

B わ からなくて

編集距離のアルゴリズム

⑤3つの編集距離を計算する。

5-1 Bから1文字を削除 (=挿入)し①へ戻る



編集距離のアルゴリズム

⑤3つの編集距離を計算する。

5-2 Aから1文字を削除し①へ戻る

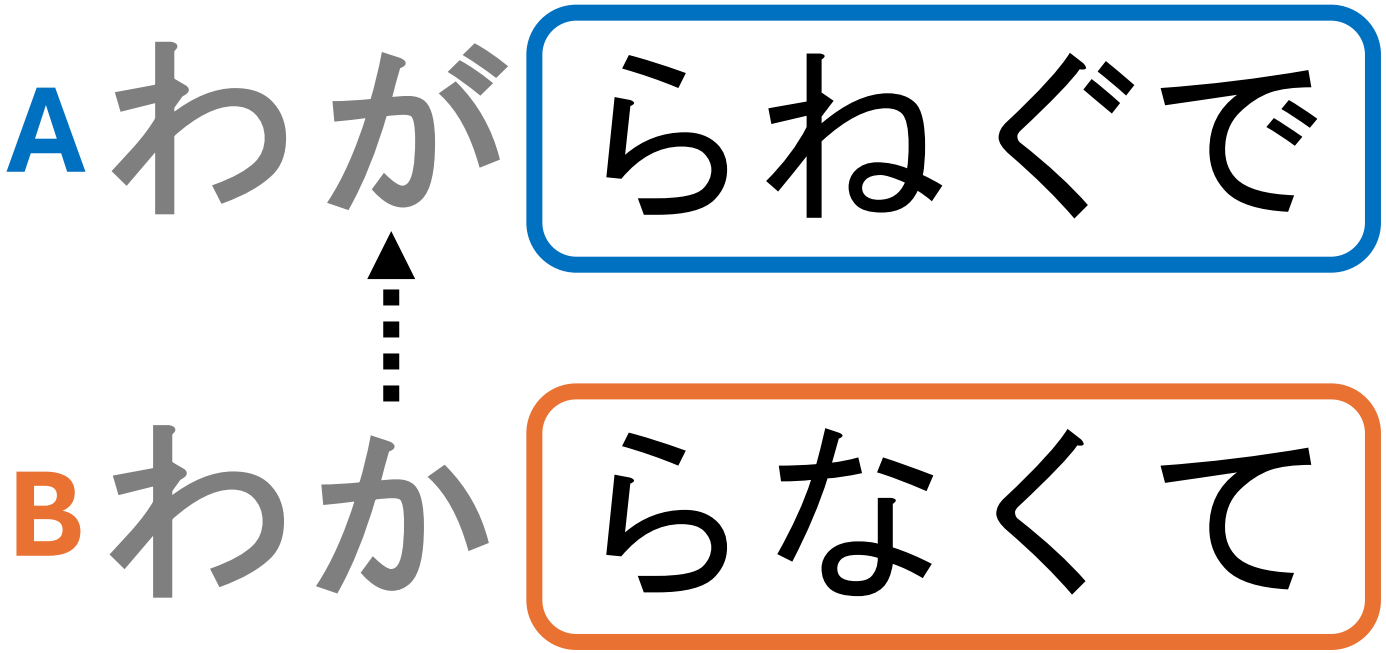
A わ~~が~~ らねぐで

B わ からなくて

編集距離のアルゴリズム

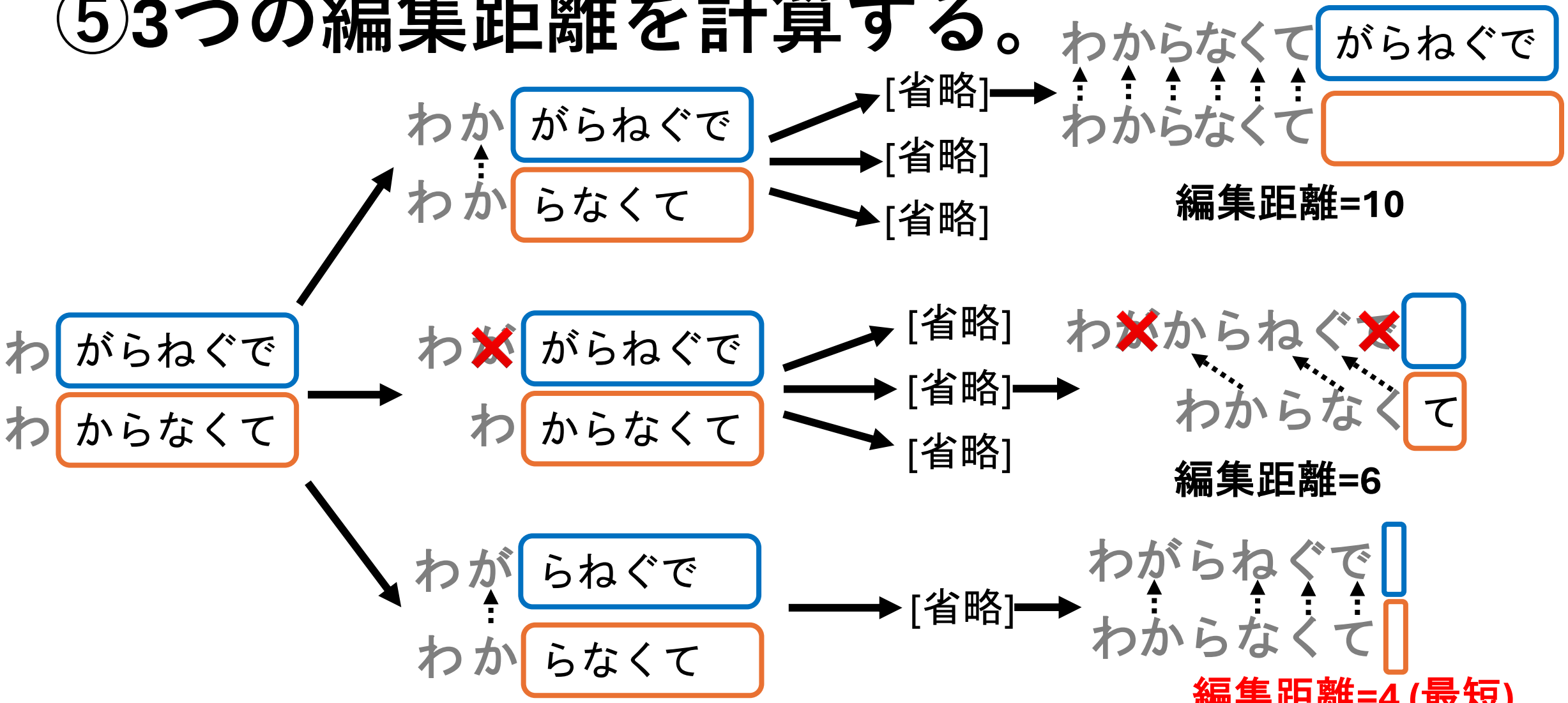
⑤3つの編集距離を計算する。

5-3 AとBから1文字を削除(=置換)し①へ戻る



編集距離のアルゴリズム

⑤3つの編集距離を計算する。



全ての計算が終わったら、最短の距離を結果とする。

再帰的関数を使ったアプローチ

編集距離の計算のように、簡単な作業を何回も繰り返しその結果を組み合わせることによって元の複雑な問題を解決できる場合、再帰処理を利用することが多い。

再帰的関数を使ったアプローチの問題点

再帰呼び出しを行う度に処理が分岐する木構造になっているため、対象の文字列が長くなるに伴い必要な計算量が爆発的に増大してしまう。

動的計画法を使ったアプローチ

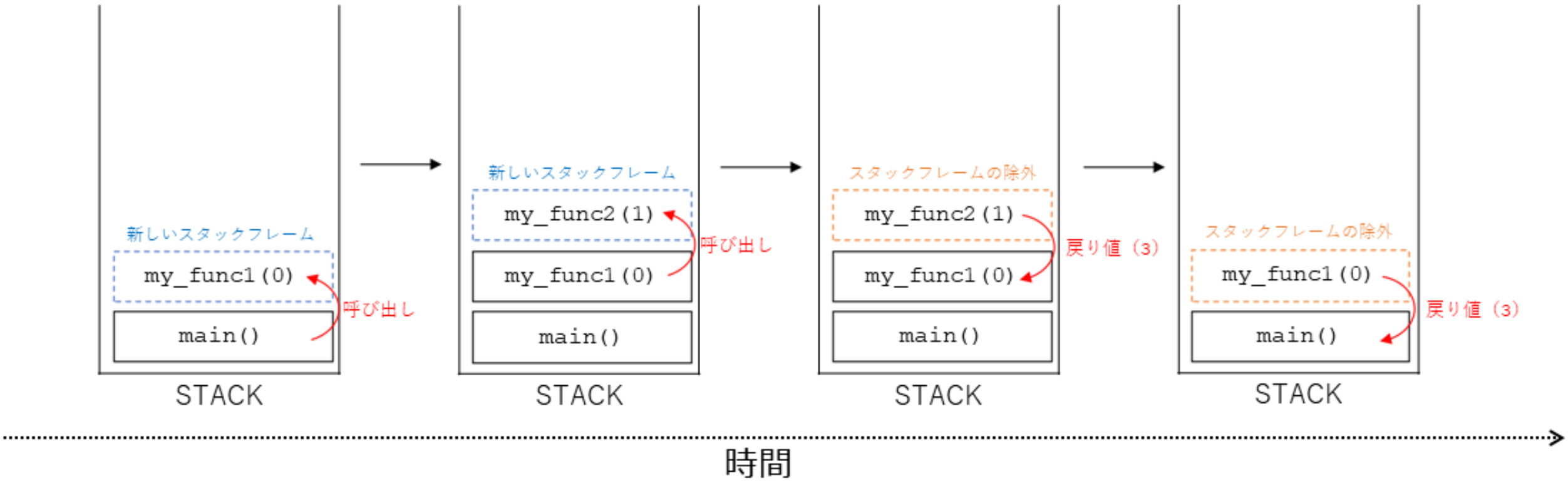
このような無駄を省くためには、一度計算した結果を保存し後で再び同じ処理を行う必要があれば記録された結果を再利用すれば良い。

このような方法は「**動的計画法**」または、「**ダイナミック・プログラミング**」と呼ばれる。

スタックフレーム

プログラムの中で関数が呼び出されると、メモリのスタック領域で「**スタックフレーム**」と呼ばれるブロックが割り当てられ、そこで関数の処理に必要な情報（関数の中で使われる変数など）が格納される。

スタックフレーム



関数の実行が終了した後はスタックフレームが除外され解放されたメモリを再利用できる。

スタックオーバーフロー

再帰が深くなるとスタックが溢れてしまい（**スタックオーバーフロー**）プログラムの実行が中断される恐れがある。

ループを使ったアプローチ

for文やwhile文などによる反復処理（ループ）の場合は、処理を繰り返す度に同じスタックフレームが再利用されるため、オーバーフローする心配がない。

メモリ消費の改善

levenshtein_loop.cでは、実際に参照されるのは現在の添字 (i, j)要素か、その直前 (i−1, j−1)の要素だけである。

この性質を利用すれば、2行だけの表で同じ処理を行うことができる。

前回の問い

- 2つの文字列の類似度を調べるにはどのようにすれば良いか？
  - 編集距離を用いる。
- 2つの文字列の類似度を調べるプログラムを設計するにはどのような方法があるか？
  - 再帰的関数や動的計画法を使った方法がある。

今回

第１回　構造体、共用体と線形リスト
第２回　探索とソート
第３回　メモリの動的確保
第４回　文字列の類似度計算、再帰、動的計画法
第５回　自由課題作成
第６回　自由課題作成
第７回　自由課題発表

今回の目標

チームで作成するプログラムを
決める。

今回の内容

1. 自由課題の説明
2. チーム分け（乱数の生成）
3. チームごとのグループワーク

今回の内容

1. 自由課題の説明
2. チーム分け（乱数の生成）
3. チームごとのグループワーク

自由課題テーマ

授業で習ったデータ構造とアルゴリズム（構造体、探索・ソート、メモリ動的確保、編集距離）やその他自分で調べたデータ構造・アルゴリズムの1つ以上を用いたCプログラムを考案し、開発する。

作成するもの

1. Cプログラム
2. 説明書（PDF）
   内容：
   - プログラムのタイトル
   - プログラムの概要
   - 使い方
   - 利用したアルゴリズムに関する説明
   - 利用したデータ構造に関する説明
   - 工夫した点
   - アピールしたい点
   - 参考文献
3. プレゼンスライド

今後のスケジュール

第5回（今回）

- チーム名の決定
- 役割分担の決定
- 本講義のs4グループでチーム固有の掲示板の作成
- ブレーンストーミングをして、作成するプログラムの決定
- 仕様書の作成
- チームでの作業スケジュールの作成
- プログラムの作成開始

第6回

- プログラムの完成
- 説明書の作成
- プレゼン資料の作成
- 発表の準備

第7回

- 発表会（発表時間15分＋質疑応答5分）

評価基準

- **対象の処理に適切なアルゴリズム・データ構造を利用しているか**
- **プログラムのアイディアとオリジナリティ**
- **プログラムの出来**
  - 説明通り動作するか
  - 技術レベル（これまで学んできたことを活用しているか）
  - ソースコードのスタイルと分かりやすさ（インデント、変数名、無駄な繰り返しの有無、関数定義を活用しているか）
- **利用した資料を参考文献で記述しているか**
- **著作権侵害行為（他人が作ったプログラムなどの盗用）をしていないか**

今回の内容

1. 自由課題の説明
2. チーム分け（乱数の生成）
3. チームごとのグループワーク

乱数の生成

rand関数:

0以上RAND_MAX以下の乱数を生成する。

stdlib.hで定義されるRAND_MAXは処理系に依存するが、少なくとも32767はある。

乱数の種

rand関数は**種**を利用した計算によって乱数を生成している。rand関数の中に種が埋め込まれているため、rand関数をそのまま利用すると毎回同じ順で乱数が表示されてしまう。

srand関数

種の値を変更する。

srand(10), srand(32)などとすることで、種の値を変更できる。ただし、このような定数を渡してもrand関数が生成する乱数は決まった順にしかならない。

そのため、乱数の種をランダムにする必要がある。

現在の時刻による乱数の種の設定

```
srand(time(NULL));
```

乱数の種を現在時刻によってランダムにする。

現在の時刻による乱数の種の設定

```
srand(time(NULL));
```

現在の暦時刻をtime_t型として取得する。

得られる暦時刻の実態は、int型やlong型などの加減乗除ができる算術型の値（単一の数値）。

現在の時刻による乱数の種の設定

```
srand(time(NULL));
```

取得した値をsrand関数に渡す。このとき、現在の時刻を示すtime_t型の暦時刻の値が、srand関数が受け取るunsigned int型に型変換される。

チーム分け

右のプログラムは0から3の乱数を生成するプログラムrand03.cである。このプログラムを書いて実行し、最初に表示されたチームを自由課題で所属するチームとする。

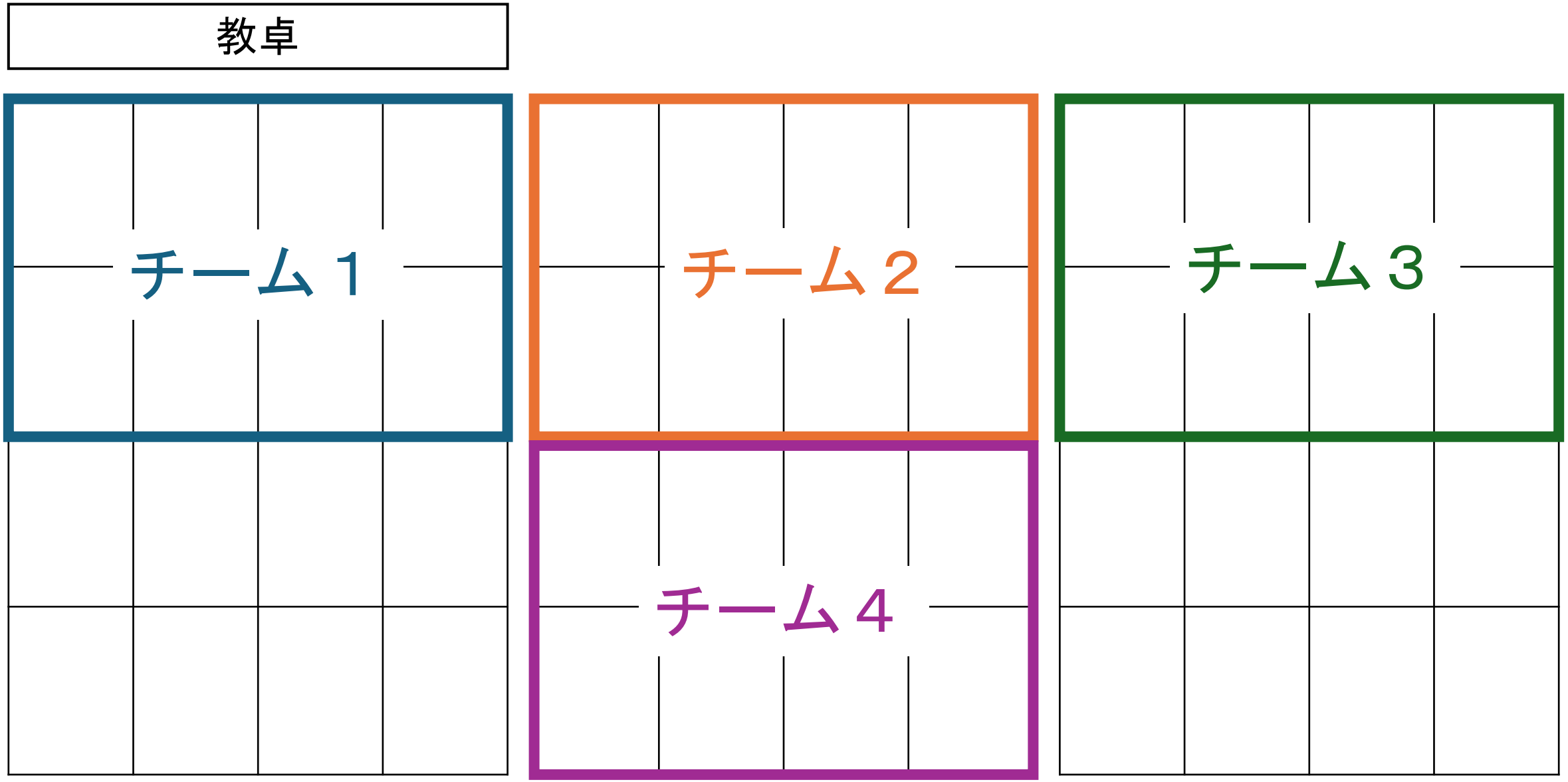
```
#include <time.h>
#include <stdio.h>
#include <stdlib.h>

int rand_0_3(void)
{
  int num = rand() % 4;
  return num;
}

int main(void)
{
  srand(time(NULL));
  printf("あなたはチーム%dです。\n", rand_0_3() + 1);

  return 0;
}
```

チーム座席



今回の内容

1. 自由課題の説明
2. チーム分け（乱数の生成）
3. **チームごとのグループワーク**

複数人でコード開発を行う上でのポイント

- **作業分担しやすいプログラム設計：**
  1. プログラムの全体的な設計を作る（仕様書）。
  2. 必要な処理とその順番を考えて記述する（疑似コード）。
  3. 疑似コードで特定された処理ごとに必要な入力情報および処理の出力を具体的に指定する。
  4. メンバーに分けて必要な関数を実装し、最後に一つのプログラムにまとめる。
- **ソースファイルをチームで共有し更新できるバージョン管理システム(gitなど)を活用する。**

今回の内容

1. 自由課題の説明
2. チーム分け（乱数の生成）
3. チームごとのグループワーク

今回の目標

チームで作成するプログラムを
決める。

課題

進捗状況を各自報告する。ただし、以下の要素を含めること。

1. チーム名
2. チームが作るプログラムの概要・仕様（代表者のみ）
3. 各自の作業スケジュールと現時点での進捗状況
4. 残っている作業・課題
5. ソースコード（添付ファイル; 未完成でOK; コード開発担当者のみ）

提出先：s4 「第5回課題（進捗状況）」

締め切り：1月12日 (火)

次回

第１回　構造体、共用体と線形リスト
第２回　探索とソート
第３回　メモリの動的確保
第４回　文字列の類似度計算、再帰、動的計画法
第５回　自由課題作成
第６回　自由課題作成
第７回　自由課題発表