第6回のキーワード

アルゴリズム関係

- 文字列探索
- 力まかせ法
- $O(n) \sim O(nm)$
- ボイヤー・ムーア法
- $O(n/m) \sim O(nm)$
- 動的配列
- コレクション

Java関係

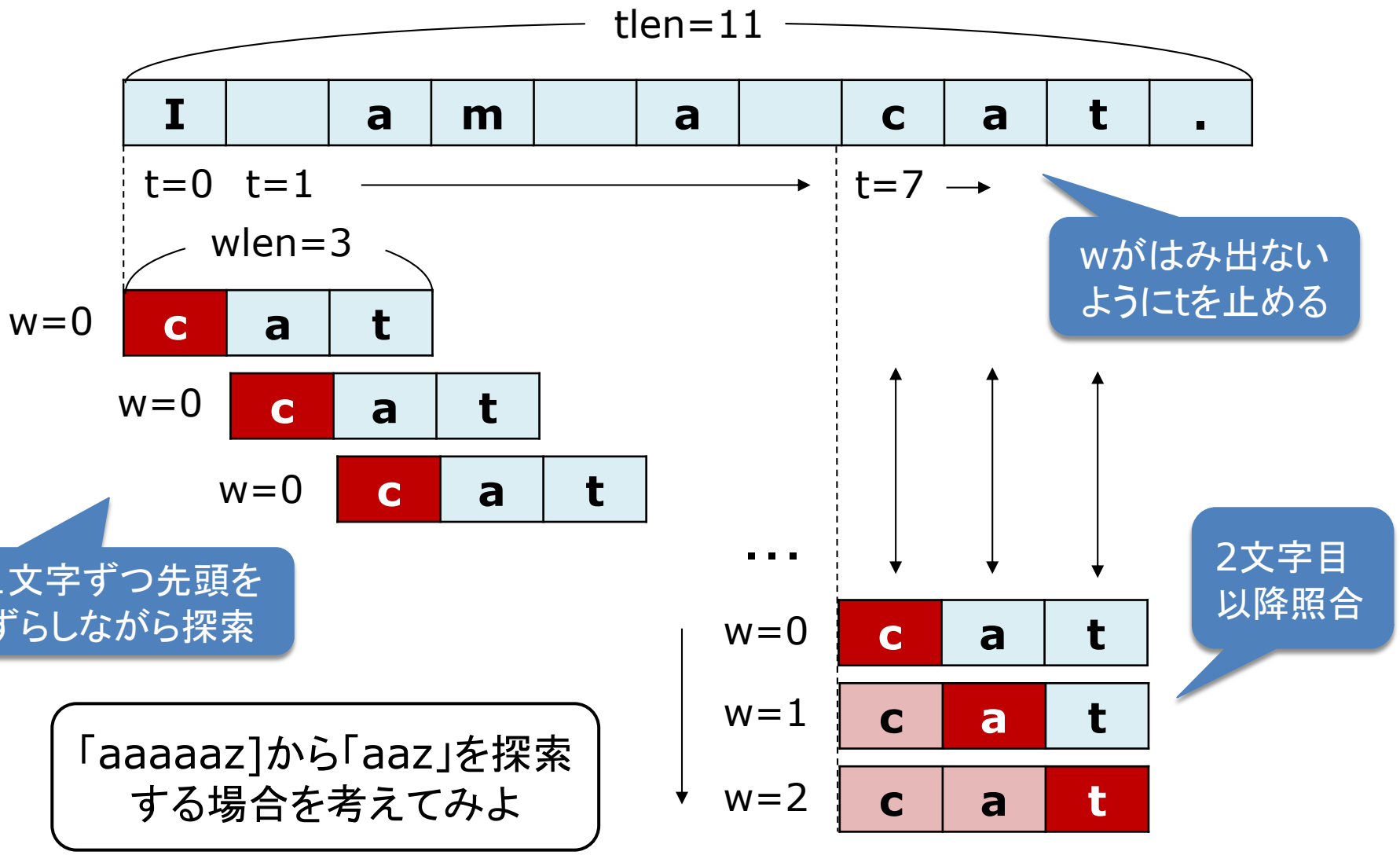
- charAt
- Arrays.sort
  Arrays.binarySearch
- ジェネリクス（総称型）
- 自然な順序
- Comparable<*E*>
- Comparator<*E*>
- ArrayList<*E*>
- ラッパークラス
- Collections.sort
  Collections.binarySearch

文字列探索

- 文字列探索
  - これまでは，データ列から1つの要素を探す問題を扱った
  - 今回は，文字「列」の中から文字「列」を探索する
  - 文字列だけでなく，DNA配列の探索などにも応用される

- 力まかせ法
  - 探索対象のテキストをn文字，探索文字列をm文字とする
  - 最初はt=0として，テキストのt文字目からt+m文字目まで，1文字ずつ順に探索文字列と照合する
  - もし，m文字すべてが一致したら，位置tで発見となる
  - そうでなければ，tを1だけ進めて同様の処理を繰り返す
  - ただし，テキストの残りがm文字未満になったら終了とする

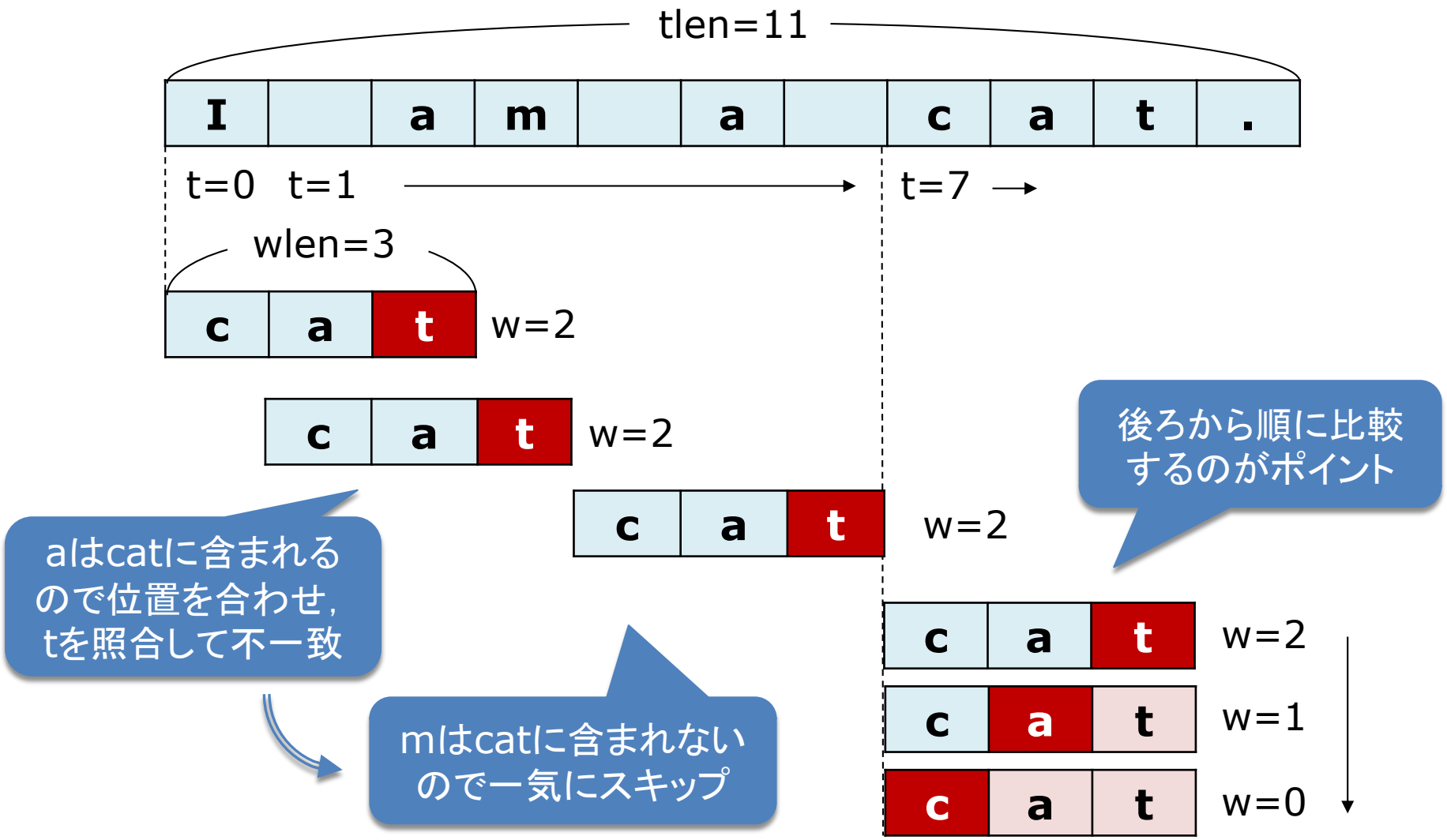
文字列探索(力まかせ法)


tlen=11
I a m a c a t .
t=0 t=1
t=7
wlen=3
w=0 c a t
w=0 c a t
w=0 c a t
wがはみ出ないようにtを止める
1文字ずつ先頭をずらしながら探索
...
w=0 c a t
w=1 c a t
w=2 c a t
2文字目以降照合
「aaaaaz]から「aaz」を探索する場合を考えてみよ

ボイヤー・ムーア法（簡易版）

- 基本的なアイデア
  - 文字列の先頭から照合するよりも，末尾から照合した方が，探索位置を大きくスキップできる
- アルゴリズムの概要
  - 対象のテキストをn文字，探索する文字列をm文字とする
  - 最初はt=0として，テキストのt+m文字目と探索文字列のm文字目から，1文字ずつ逆順に照合する
  - もし，m文字すべてが一致したら，位置tで発見となる
  - そうでなければ，テキスト側の不一致文字が探索文字列に含まれない場合は，tをmだけ進め，探索を続ける
  - テキスト側の不一致文字が探索文字列に含まれる場合は，その文字同士が合うようにtを適切に進め，探索を続ける

ボイヤー・ムーア法（簡易版）



クラス型の配列(復習)

- クラス型の配列の作成
  - class Item { int code; String name; }
  - Item [] data = new Item[10];
  - for (int i = 0; i < data.length; i++) data[i] = new Item();
- 配列要素のメンバのアクセス
  - if (data[i].code == code)
- 配列要素の(位置の)交換
  - Item t; t = data[i]; data[i] = data[j]; data[j] = t;
  - Javaの配列の構造やクラス型変数の代入について再確認

Javaによる探索とソート

- 配列の探索とソート
  - java.util.Arraysの静的メソッドが使用できる
  - 線形探索： Arrays.asList(*array*).indexOf(*key*)
    - ただし，これはクラスの配列でしかうまく動かない
    - intやdoubleなど基本型の配列では期待通りに動作しないので注意
  - 2分探索： Arrays.binarySearch(*array*, *key*)
  - ソート： Arrays.sort(*array*)

- Comarableインタフェース
  - 2分探索やソートでは，要素が比較できなければならない
  - そのためには，要素のクラスはComparableインタフェースを実装し，compareToメソッドを持つことが必要

動的配列とジェネリクス

- ArrayList<*E*>
  - 要素数を動的に変更できる配列（のようなクラス）
  - *E*に要素のクラス名を当てはめて使う（ジェネリクス）

    例）ArrayList<*String*> *alist* = new ArrayList<*String*>();
  - 要素の追加/取得/変更には，add/get/setメソッドを使う

    例）*alist*.add(*str*) / *alist*.get(*i*) / *alist*.set(*i*, *str*)

- ArrayListの探索とソート
  - java.util.Collectionsの静的メソッドが使用できる
  - 線形探索：*alist*.indexOf(*key*)
  - 2分探索：Collections.binarySearch(*alist*, *key*)
  - ソート：Collections.sort(*alist*)