データ構造とアルゴリズム
（第4回）

モバイルコンピューティング研究室
柴田史久

1

1

本日の講義内容

- 基本的なデータ構造（３）
  - 循環リスト
  - 双方向リスト

2

2

教科書 第5章（pp.135～142）

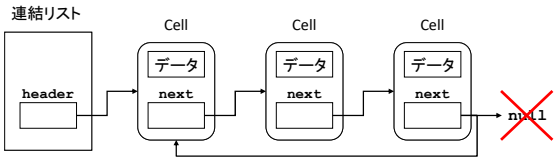
基本的なデータ構造(3)
循環リスト

3

3

循環リスト(circular list)

- 連結リストにおいて最後の要素の次の要素を先頭の要素にしたもの
- 厳密には「最初」「最後」がなくなる

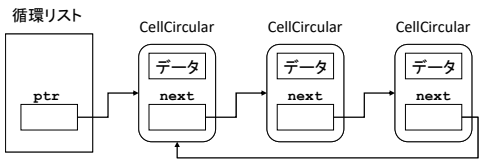


4

4

循環リストの操作

- 循環リストが空　＝　ptr が null
- 要素の挿入・削除　＝　連結リストと同様
- 要素をたどる場合は？　→　最後の判定が異なる



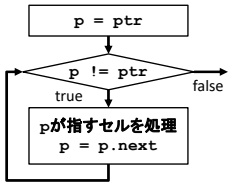

```
class CellCircular {
  CellCircular next ;  // 次の要素へのリンク
  MyData       value ; // この要素の値
}
```

5

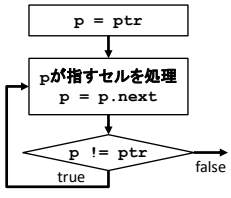
5

リストをたどる操作

```
CellCircular ptr ;
if (ptr != null) {
  CellCircular p = ptr ;
  while (p != ptr) {
    // pが指すセルを処理
    p = p.next ;
  };
}
```




```
CellCircular ptr ;
if (ptr != null) {
  CellCircular p = ptr ;
  do {
    // pが指すセルを処理
    p = p.next ;
  } while (p != ptr) ;
}
```

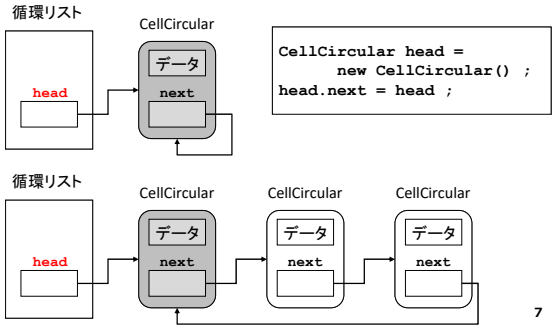


6

6

リストの頭を用いた循環リスト

- 境界条件（空と最後の判定）を統一
- 先頭を表す特別なセルを導入



7

7

リストをたどる操作(頭を利用する場合)

- リストの最後は head と同じところを指しているかどうかで判定可能
- 循環リストをたどる操作

```
CellCircular head ;

for (CellCircular p = head.next; p != head; p = p.next) {
  // p で指されるセルの処理
}
```

8

8

リストの頭を用いる場合の注意点

- リストの要素を順番に処理する際は，リストの頭をスキップする必要がある
- 循環リストを回転（rotation）できない
  - リストの頭がない場合，ptrの指す先を変更することで循環リストを回転できる

9

9

教科書 第4章（pp.142～156）

基本的なデータ構造(3)

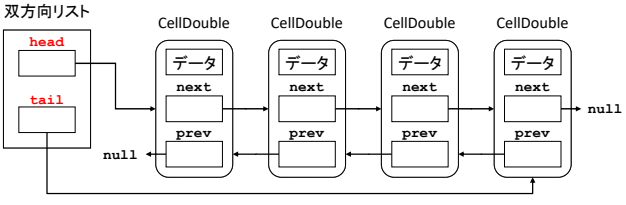
双方向リスト

10

10

双方向リスト(doubly-linked list)

- 双方向連結リスト，重連結リスト
- 各要素について前後両方のリンクを持たせたもの
- 前要素も忘れずに処理することが大事
- 基本的な双方向リスト

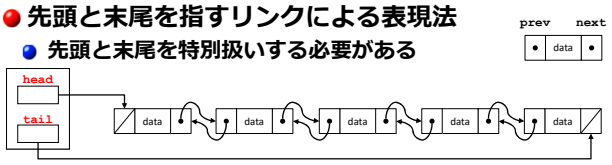


11

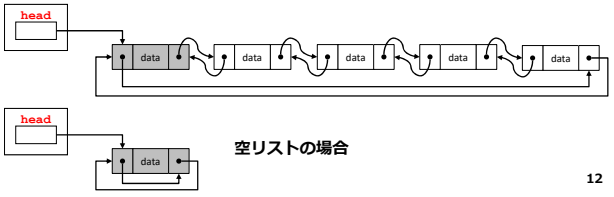
11

双方向リストの表現法

- 先頭と末尾を指すリンクによる表現法
  - 先頭と末尾を特別扱いする必要がある



- リストの頭を用いた表現法（循環リスト）



12

12

双方向リストの特徴

- 利点
  - リストの要素を前後に自由にたどれる
  - 要素の挿入・削除が連結リストより容易
- 欠点
  - 余分なリンクが必要

13

13

双方向リストの実装

- セルの定義（CellDoubleクラス）
  - 前のセルへのリンク：prev
  - 後のセルへのリンク：next
  - データ：data
    - 必要に応じてデータを扱うクラスを定義
- 双方向リストの定義（DoublyLinkedListクラス）
  - リストの頭へのリンク（循環リストの方法）

14

14

CellDoubleクラスの雛型

```
public class CellDouble {

  CellDouble  prev ;
  CellDouble  next ;
  Object      data ;

  public CellDouble(Object d) {
    this.data = d ;
    this.prev = this.next = null ;
  }

}
```

Cellクラスの雛型と比較してみよう

15

15

Cellクラスの雛形

参考

```
public class Cell {

  Cell    next ;
  MyData  data ;

  public Cell(MyData d) {
    this.data = d ;
    this.next = null ;
  }

}
```

16

16

DoublyLinkedListクラスの雛型

```
public class DoublyLinkedList{

  private CellDouble head ;

}
```

- リストの頭を用いた双方向リスト
- リストの頭へのリンクのみを持てばよい

17

17

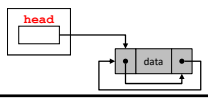
双方向リストの初期化

```
public class DoublyLinkedList{

  private CellDouble head ;

  public DoublyLinkedList() {
    // リストの頭を作成
    this.head = new CellDouble("**List Head**") ;
    // リストの頭のprevとnextが自分自身(this.head)を指すように
    this.head.prev = this.head.next = this.head ;
  }
}
```

- 初期化では，headが指す先（リストの頭）のprevとnextがリストの頭自身を指すようにする



空リストの場合

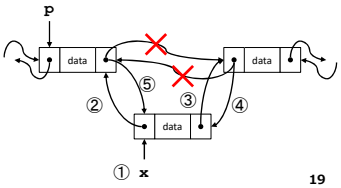
18

18

双方向リストへの挿入

- pの次に挿入
  - ① 新しいセルの作成（変数x）
  - ② 「xの前のセル」を「p」に設定
  - ③ 「xの次のセル」を「pの次のセル」に設定
  - ④ 「pの次のセルの前」を「x」に設定
  - ⑤ 「pの次のセル」を「x」に設定

```
x = new
  CellDouble(data); // ①
x.prev = p;         // ②
x.next = p.next;    // ③
p.next.prev = x;    // ④
p.next = x;         // ⑤
```

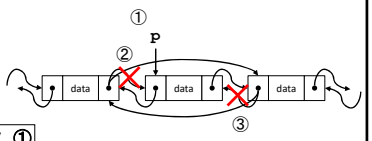


19

19

双方向リストからの削除

- pを削除
  - ① リストの頭でないことを確認
  - ② 「pの前のセルの次」を「pの次」に設定
  - ③ 「pの次のセルの前」を「pの前」に設定




```
if (p == head) {      // ①
  // エラー処理
}
p.prev.next = p.next; // ②
p.next.prev = p.prev; // ③
```

20

20

スタック･待ち行列の実現

- 双方向リストでもスタック・待ち行列が実現可能
- リストの頭を用いた表現法の使い勝手がよい
- 待ち行列では任意の要素を削除可能
  - 特定の要素を取り出して優先的に処理

21

21

まとめ

- 循環リスト
- 双方向リスト

22

22

参考文献

- 定本 Javaプログラマのための
  アルゴリズムとデータ構造（近藤嘉雪）
- 新・明解 Javaで学ぶ
  アルゴリズムとデータ構造（柴田望洋）
- 岩波講座ソフトウェア科学 3
  アルゴリズムとデータ構造（石畑清）
- Javaで学ぶアルゴリズムとデータ構造
  Robert Lafore（著）・岩谷 宏（翻訳）
- Java アルゴリズム+データ構造完全制覇
  オングス (著)・杉山 貴章・後藤 大地 (監修)

23

23