アルゴリズムとデータ構造

第8回　スタックとキュー

第8回のキーワード

アルゴリズム関係

- スタック（stack）
- LIFO: Last In, First Out / 後入れ先出し（FILO: First In, Last Out）
- プッシュ（push）
- ポップ（pop）
- トップ（top）
- スタックポインタ（stack pointer）
- キュー / 待ち行列（queue）
- FIFO: First In, First Out / 先入れ先出し
- エンキュー（enqueue）
- デキュー（dequeue）
- リングバッファ（ring buffer）
- $O(1)$

Java関係

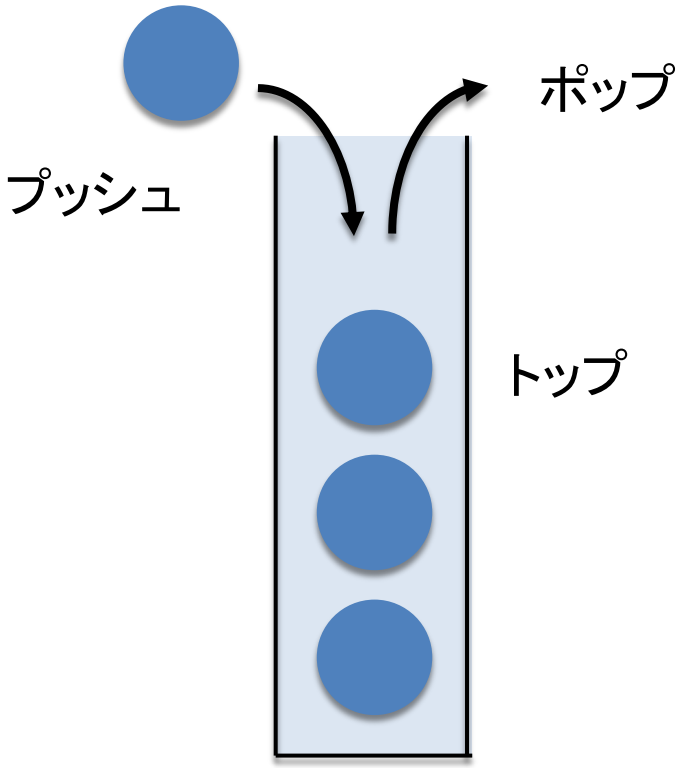
- i = (i + 1) % a.length
- throw, throws

スタックとキュー

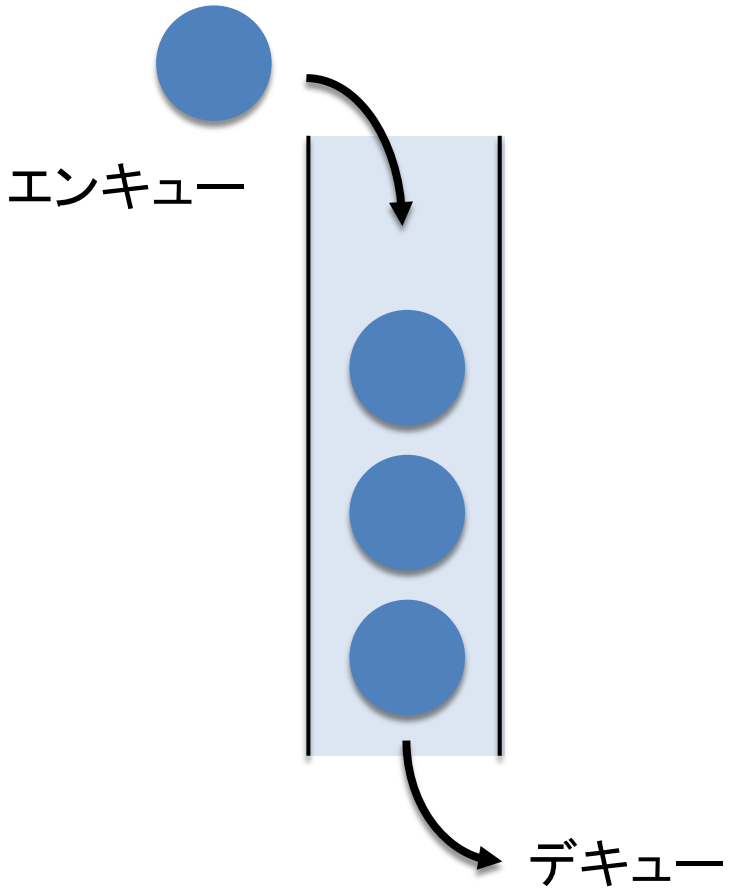
- スタック（stack）
  - 物を積むように，後から入れたデータから順に取り出せる「後入れ先だし」（LIFO）のデータ構造
  - スタックポインタを用いることで $O(1)$ での処理が可能
- キュー（queue）
  - 行列に並ぶように，先に入れたデータから順に取り出せる「先入れ先出し」（FIFO）のデータ構造
  - リングバッファによって実現すると $O(1)$ での処理が可能
- 両端キュー（double-ended deque）
  - データ列の両端から出し入れができるデータ構造
  - スタックとキューの機能をあわせ持つ

スタックとキュー

- スタック
  - LIFO：後入れ先出し

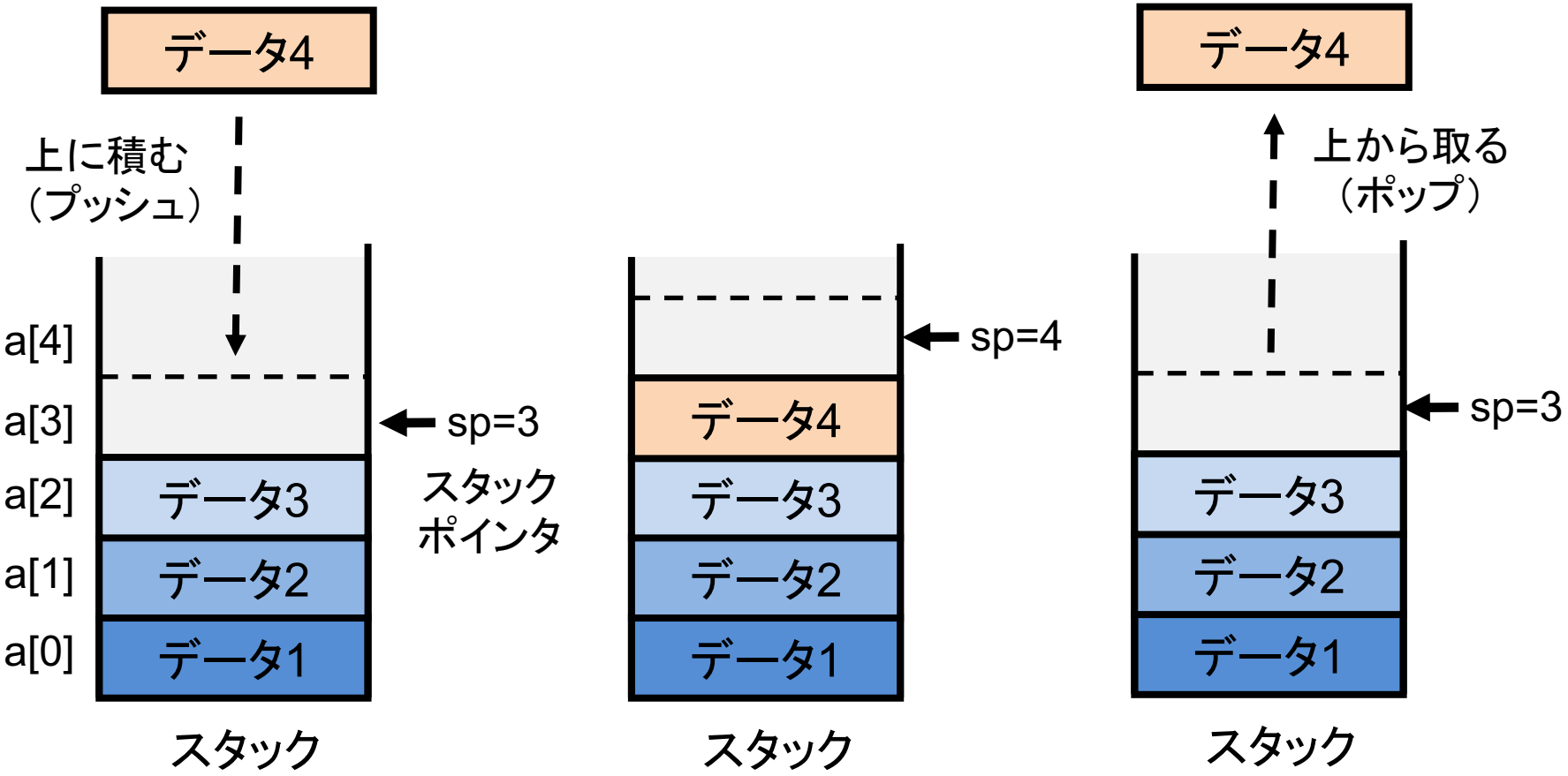


- キュー
  - FIFO：先入れ先出し



配列によるスタック

- スタックポインタ（整数）の増減で管理する
  - 処理時間はデータ数と無関係の一定値になる ⇒ $O(1)$



確認問題

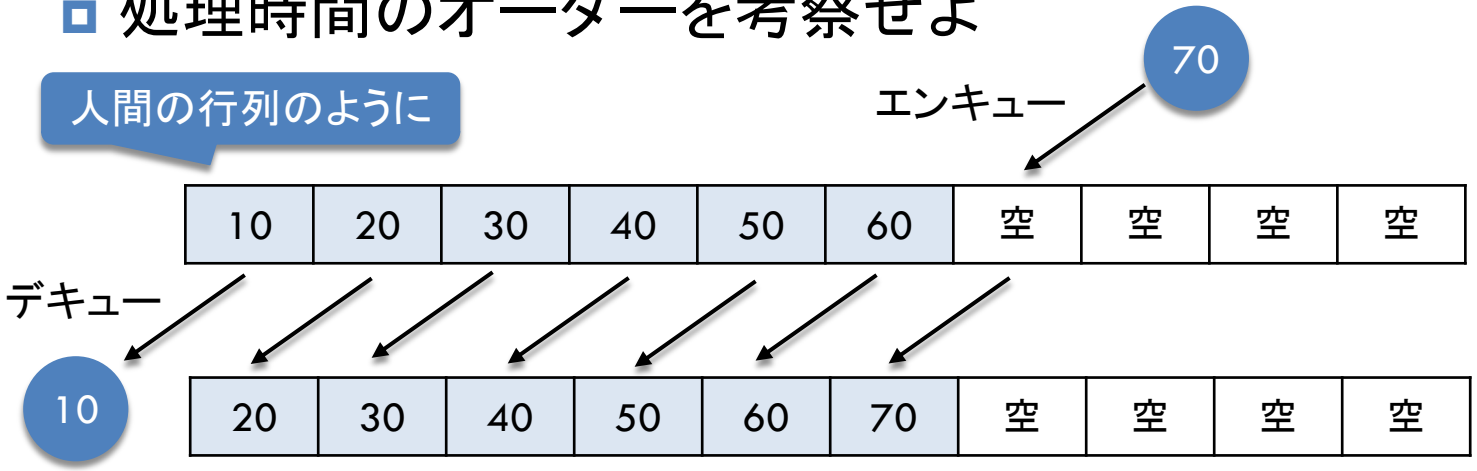
- 配列によるスタック
  - 下図は配列で実装されたスタックの使用中の状態である

  - これに対して，pop()→pop()→push(4)→push(3)→pop() と操作した場合の配列とスタックポインタの状態変化を示せ

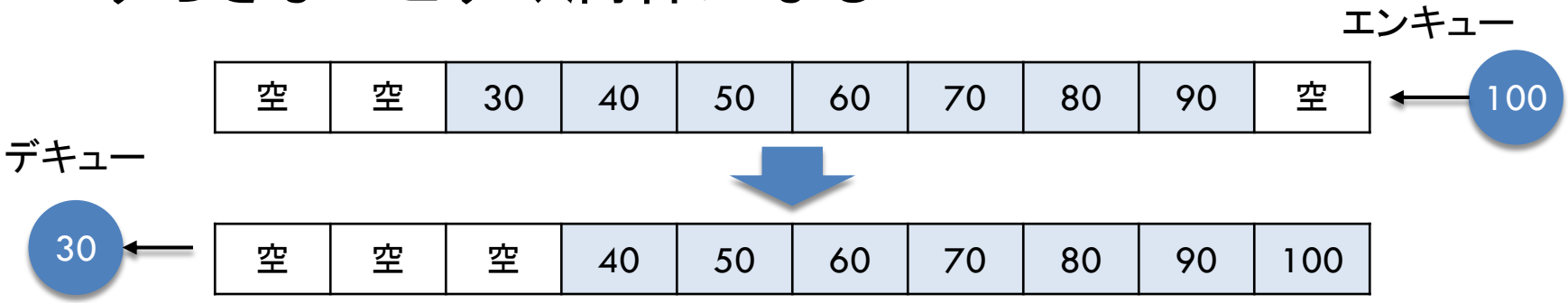
- スタックの実装
  - 配列を int [] array，スタックポインタを int sp，スタックに追加または削除するデータを int data とする
  - 上記を用いてJavaでスタックを実装するとき，プッシュ操作とポップ操作の要点をコードで示せ（演習課題の1）

キューを配列でどう実現するか

- 内容をずらすと時間がかかる
  - 処理時間のオーダーを考察せよ

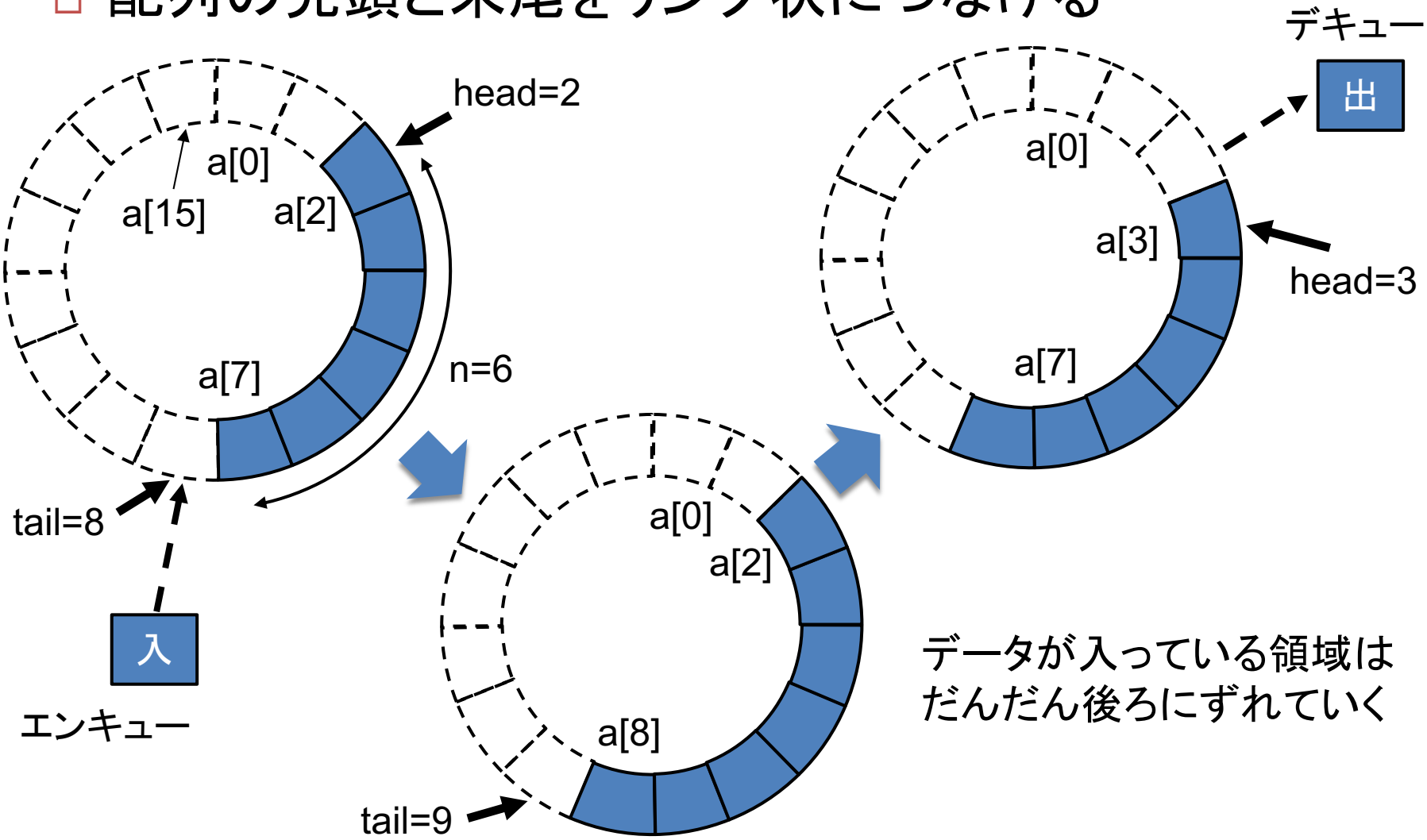


- ずらさないとすぐ満杯になる



リングバッファ

- 配列の先頭と末尾をリング状につなげる



データが入っている領域は
だんだん後ろにずれていく

確認問題

- 配列によるキュー
  - 下図は配列で実装されたリングバッファの状態である

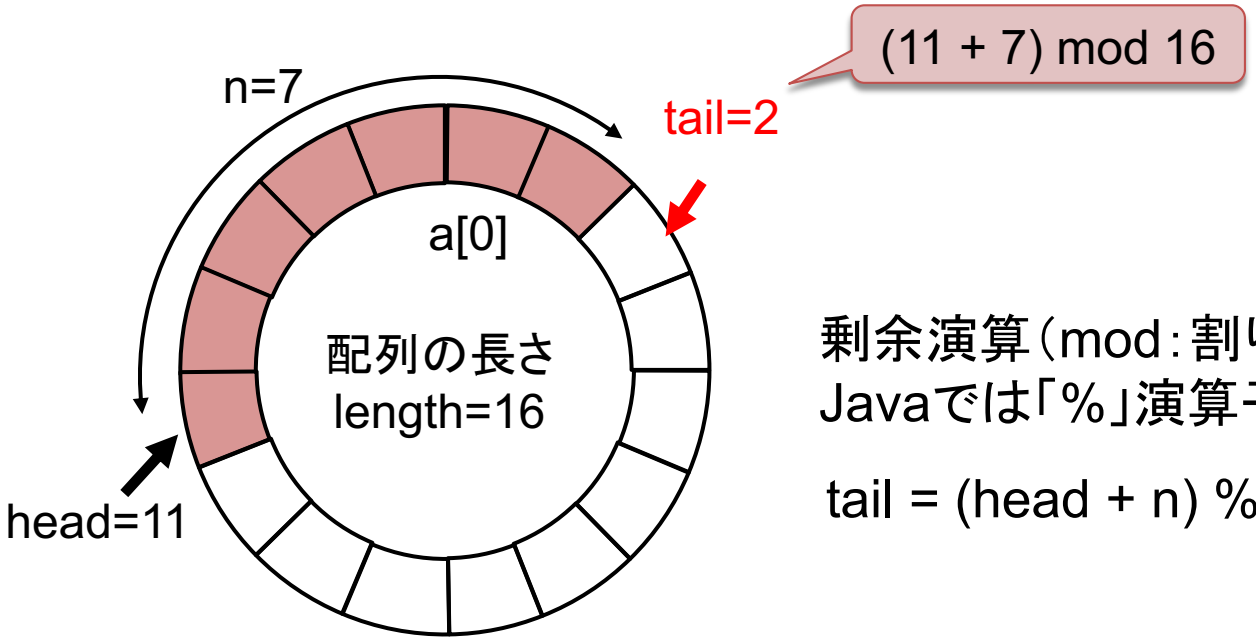
a	空	4	1	3	空

  - これに対して，dequeue()→dequeue()→enqueue(4)→enqueue(3)→dequeue() と操作した場合の状態変化を示せ

- リングバッファの実装
  - 前問の最終状態において，配列の長さ(5)，データの先頭の添字(4)，データ数(2)から，次にデータを追加すべき要素の添字(1)をどう求めればよいか考えよ
  - 配列の長さをlength，キューの先頭の添字をhead，データ数をnとするとき，次の追加位置を求める計算式を示せ

リングバッファ

- 配列内で使用要素が循環する
  - 配列の終端まで埋まると，また先頭要素から使用していく
  - 剰余演算を使えば，添字の値を循環させることができる
  - 例）5で割った余り：0, 1, 2, 3, 4, 0, 1, 2, 3, 4, 0, 1, 2, …



剰余演算（mod：割り算の余り）は
Javaでは「%」演算子

tail = (head + n) % length