データベースシステム
外部ソート

これまで考えてきたこと

- 効率よく検索するための物理的データ格納方式と索引について学んできた
- これまで想定した「検索」は「選択演算」のこと
- 思い出そう：（関係代数で）検索のための演算には何があった？

今回取り上げる演算

- 結合演算

- 射影演算→ソート処理が重要

ポイント

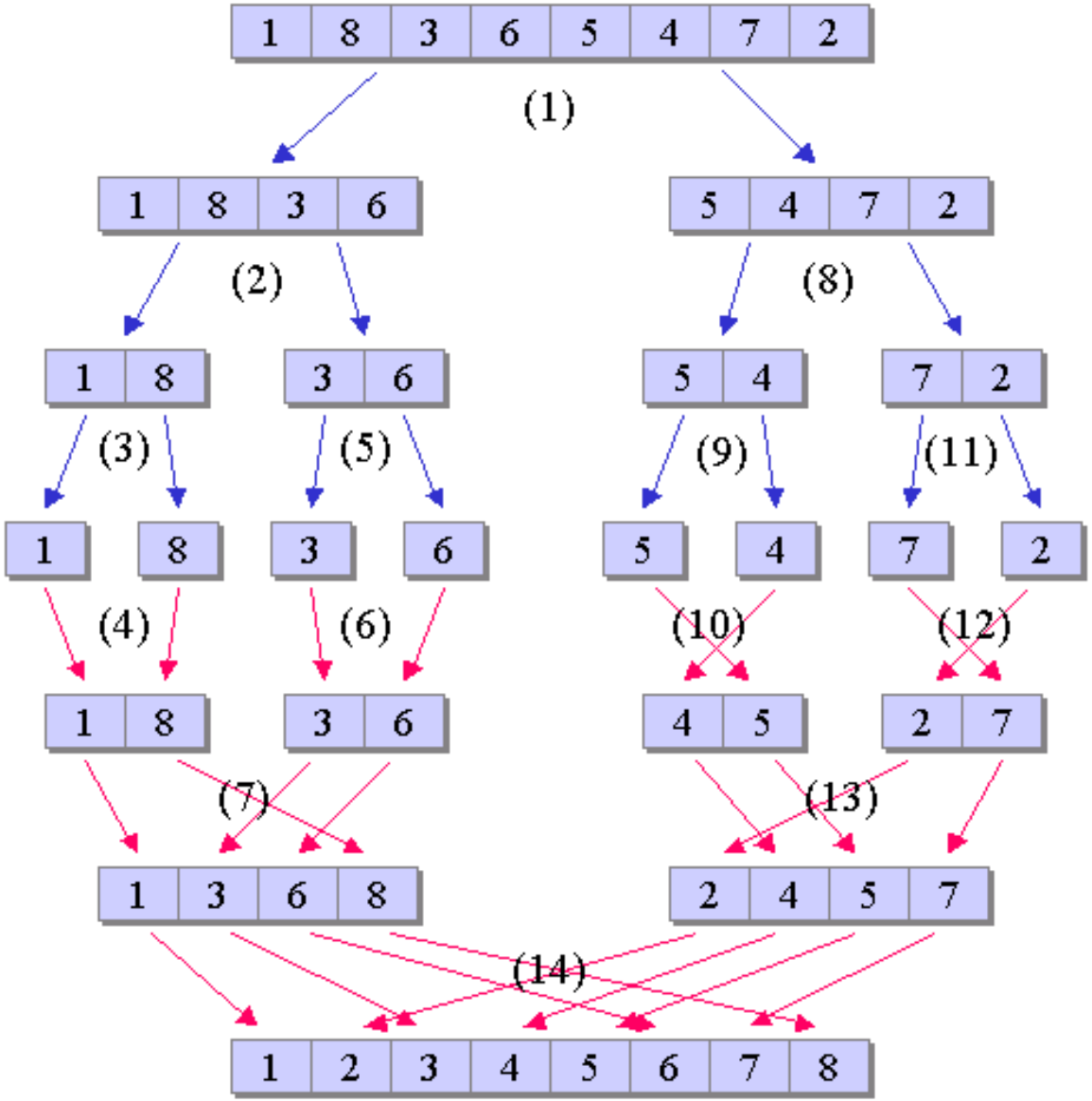
なぜ射影演算の
コストが重要なの？

今から考えること

- ここでも考えるべきは「IOコスト」，つまりメモリに呼び出すページ数を以下に抑えるか，です

- IOコストを踏まえながら「結合アルゴリズム」と「ソートアルゴリズム」をいくつか勉強します。

復習：マージソート


1 8 3 6 5 4 7 2
(1)
1 8 3 6
5 4 7 2
(2)
(8)
1 8
3 6
5 4
7 2
(3)
(5)
(9)
(11)
1
8
3
6
5
4
7
2
(4)
(6)
(10)
(12)
1 8
3 6
4 5
2 7
(7)
(13)
1 3 6 8
2 4 5 7
(14)
1 2 3 4 5 6 7 8

データベースの中身をソートする

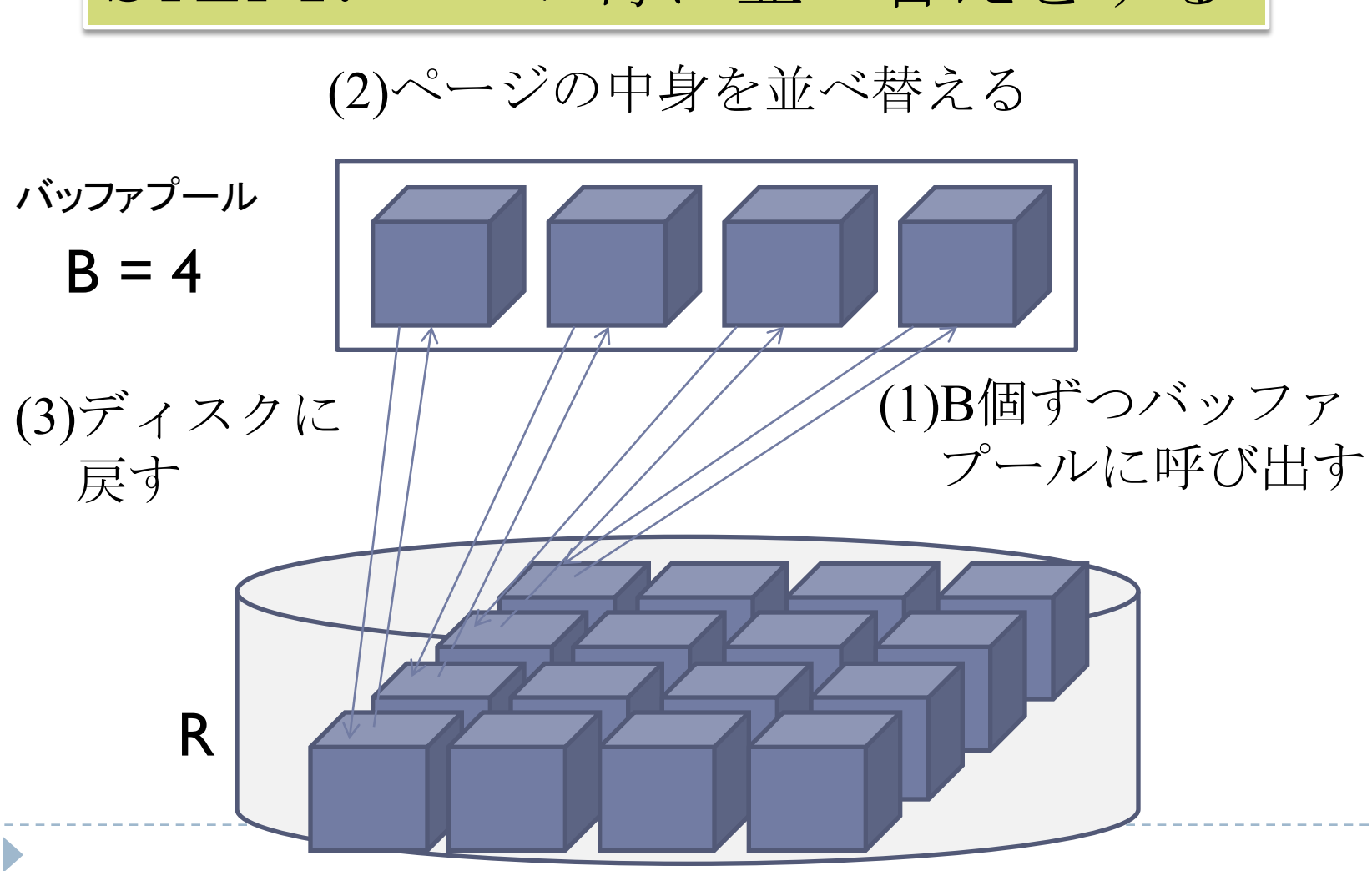
- 基本的にはマージソートを使う
- しかしソートするにはデータをメモリに持ってこなければならない

1GBもあるような巨大なテーブルはどうやってソートしよう？

外部ソート(K-wayマージソート)
ディスクとバッファプールを使ってメモリに収まりきらないデータのソートをしてしまう

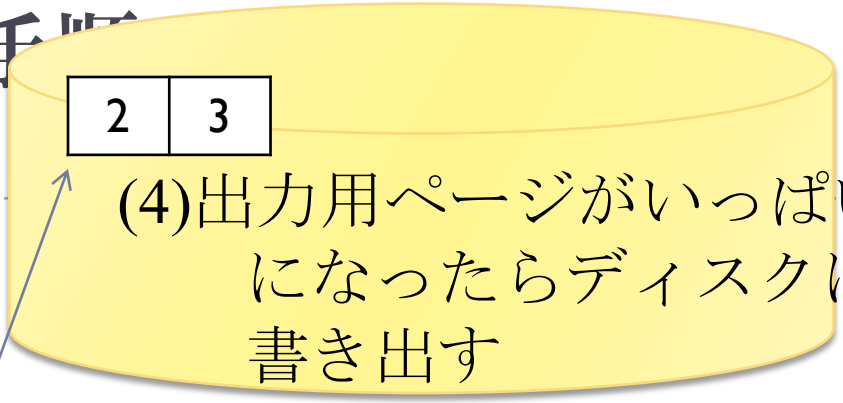
K-wayマージソートの手順

STEP1: ページ毎に並べ替えをする

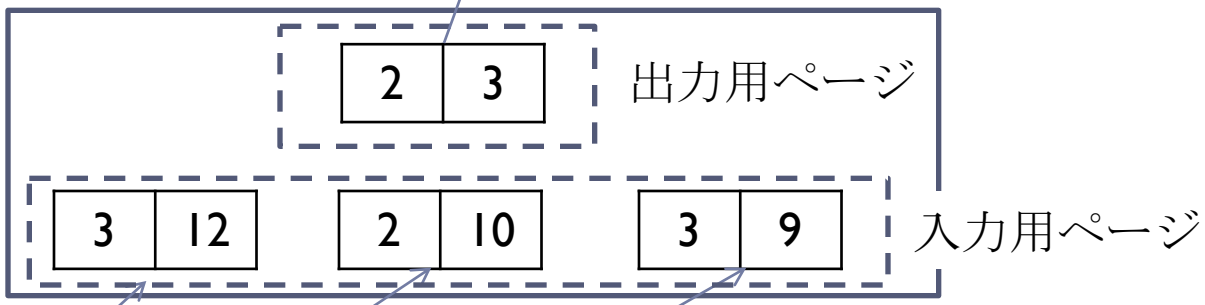


K-wayマージソートの手順

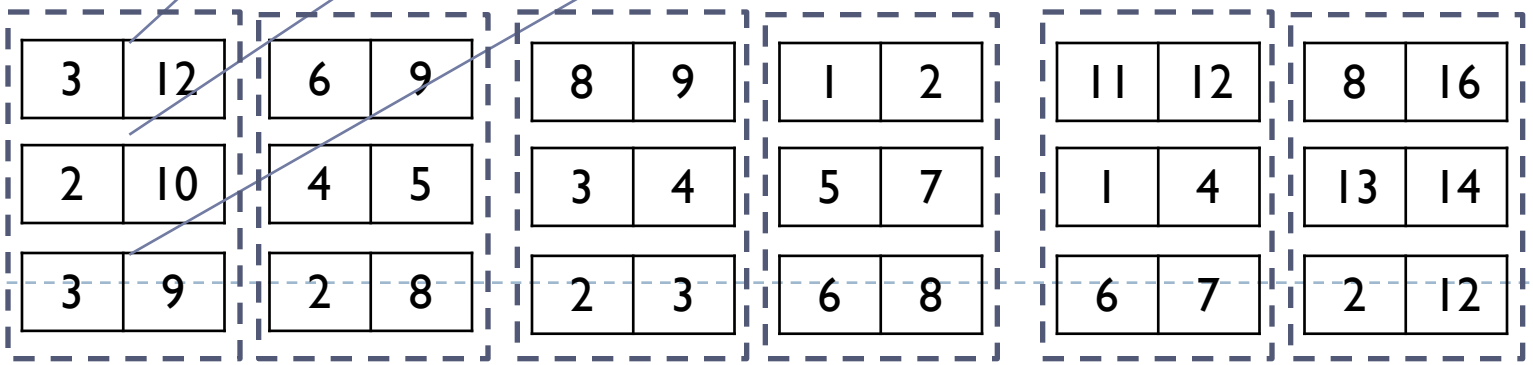
STEP2：「ラン」(Level1)内で並べ替え



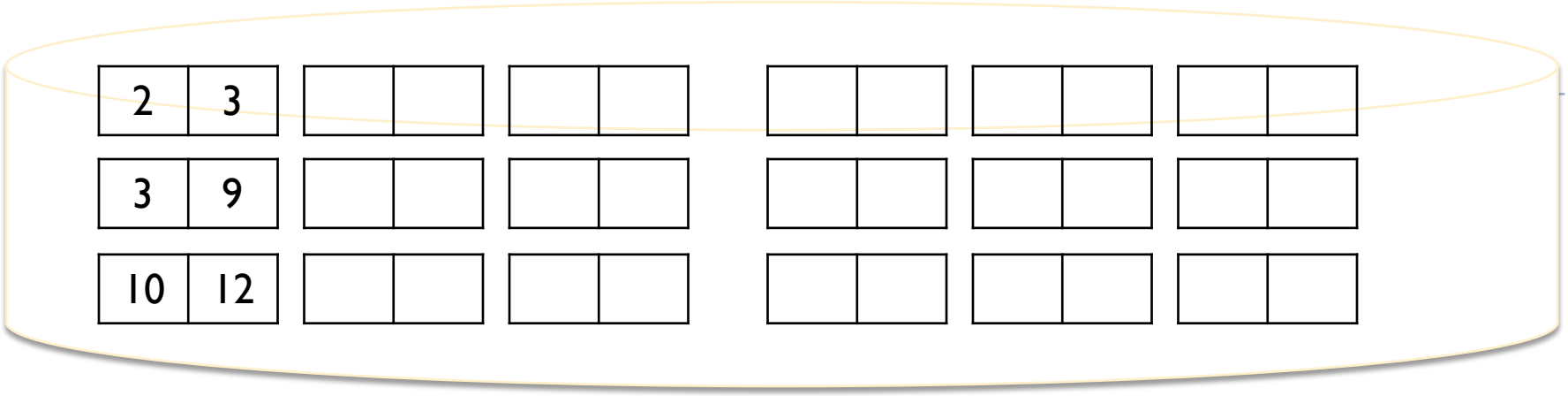
(3)小さい順に出力用ページに並べる

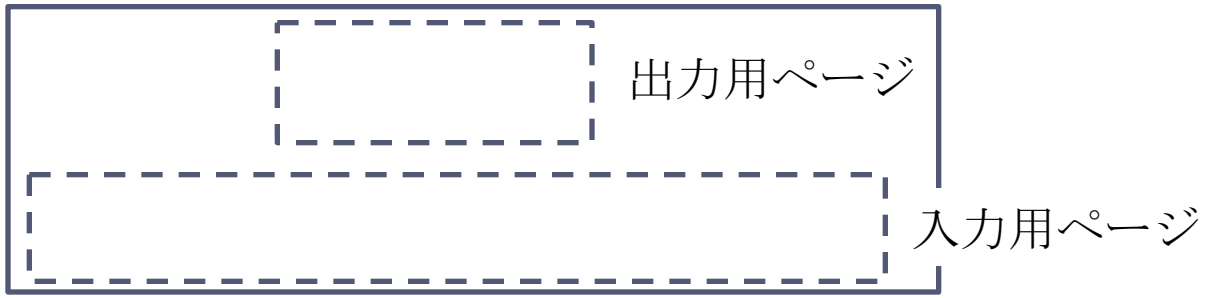


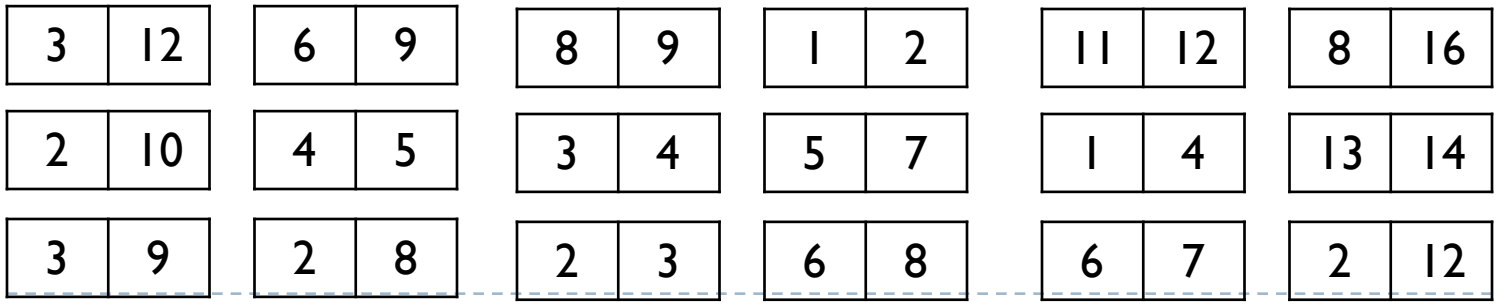
(1) K=B-1個分のページをレベル1の「ラン」とする
(2) 「ラン」毎にバッファプールに呼び出す



K-wayマージソートの手順（続きをやってみよう）



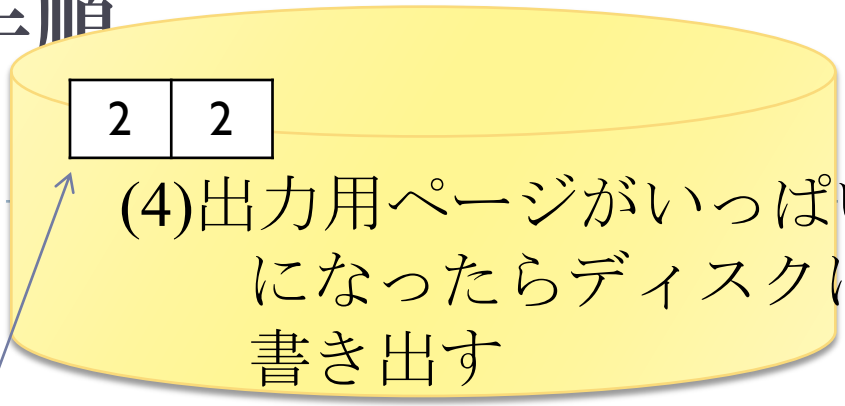




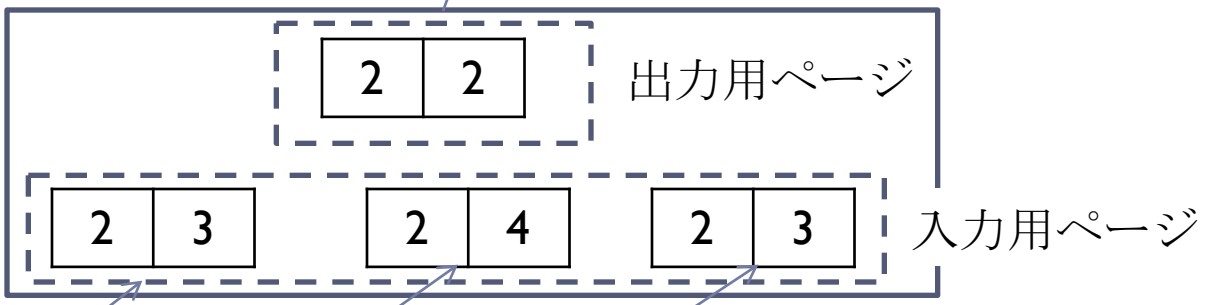
K-wayマージソートの手順

STEP3：「ラン」(Level2)内で並べ替え

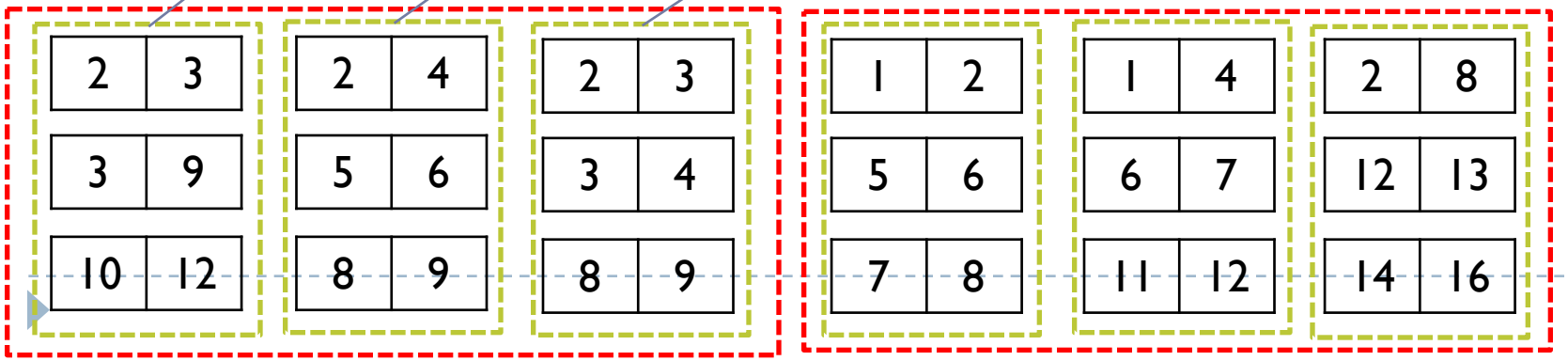
(4)出力用ページがいっぱいになったらディスクに書き出す



(3)小さい順に出力用ページに並べる

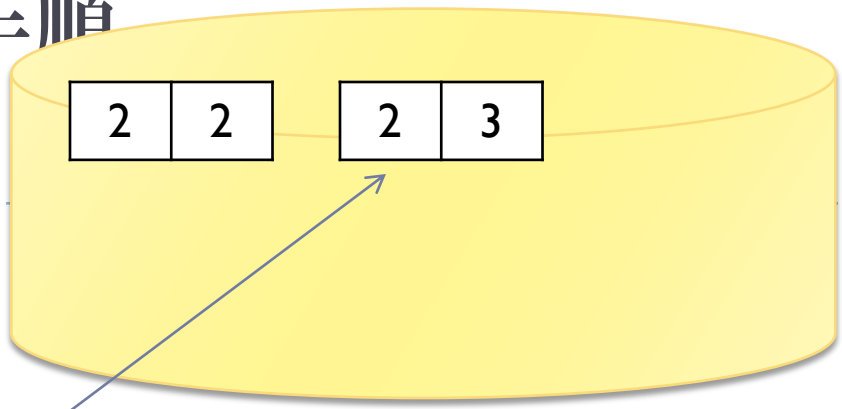


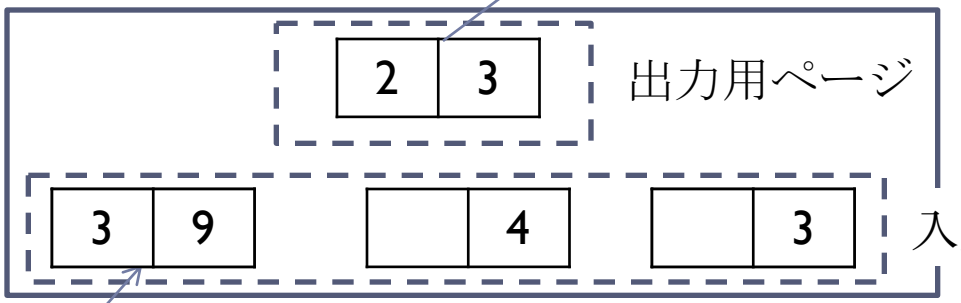
(1) M-1個分のレベル0のランをレベル1の「ラン」とする
(2) レベル1内の各レベル0ランから1ページずつ呼び出す



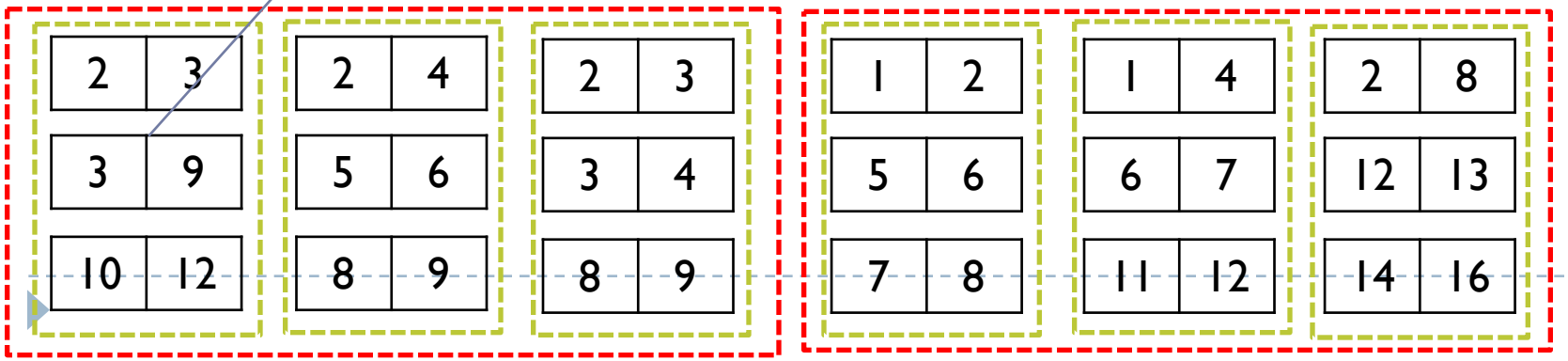
K-wayマージソートの手順

STEP3：「ラン」
(Level2)内で並べ替え

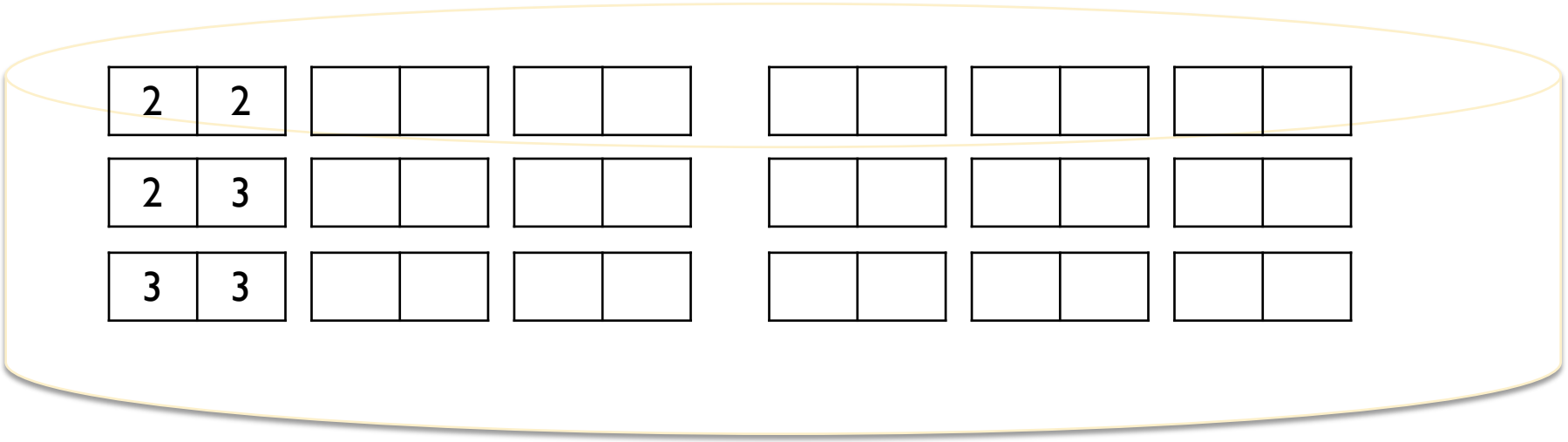


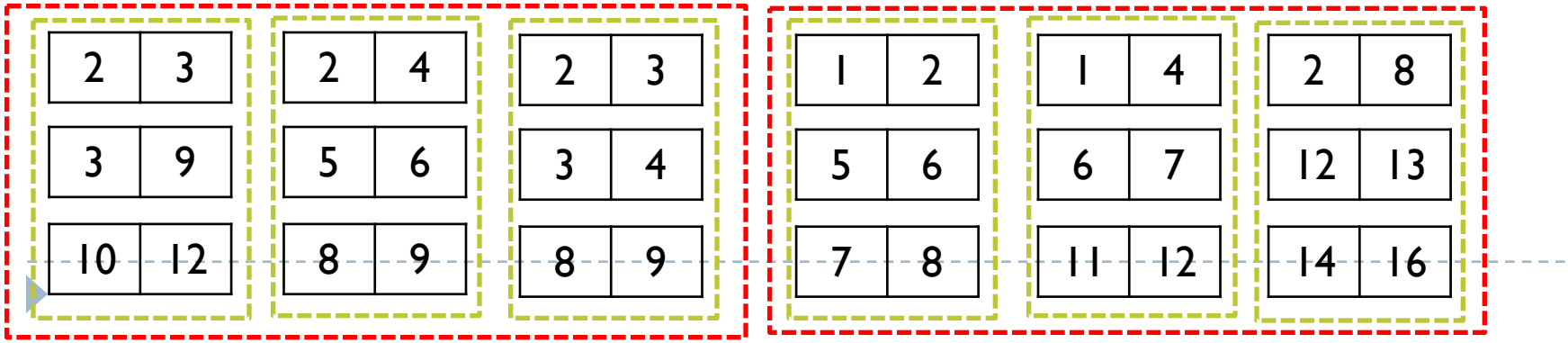


(5)ページが
空になったら次のページを呼び出す



K-wayマージソートの手順(続きをやろう)

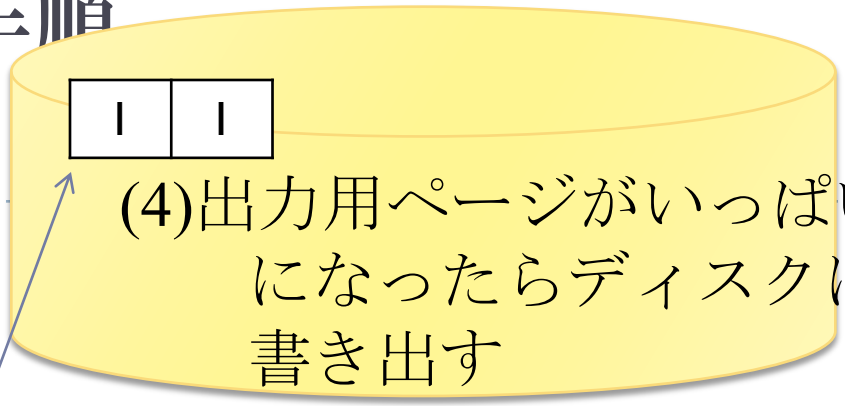




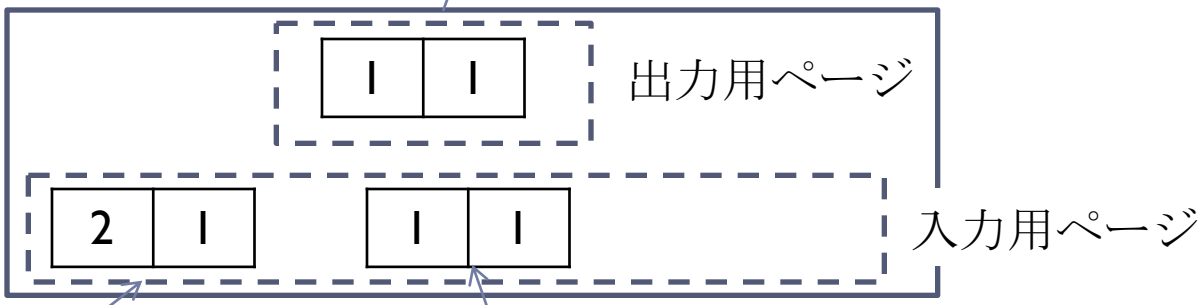
K-wayマージソートの手順

STEP4：「ラン」(Level)内で並べ替え

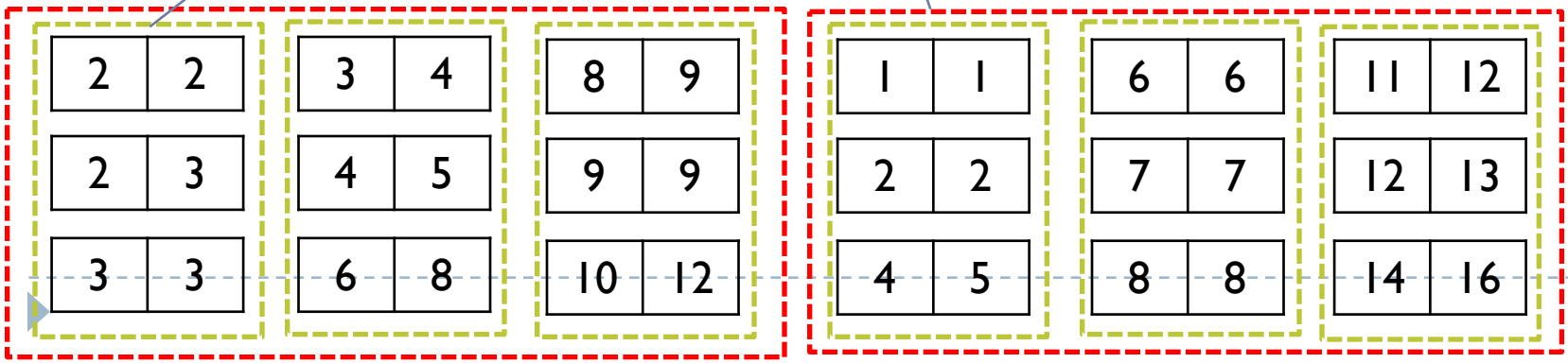
(4)出力用ページがいっぱいになったらディスクに書き出す



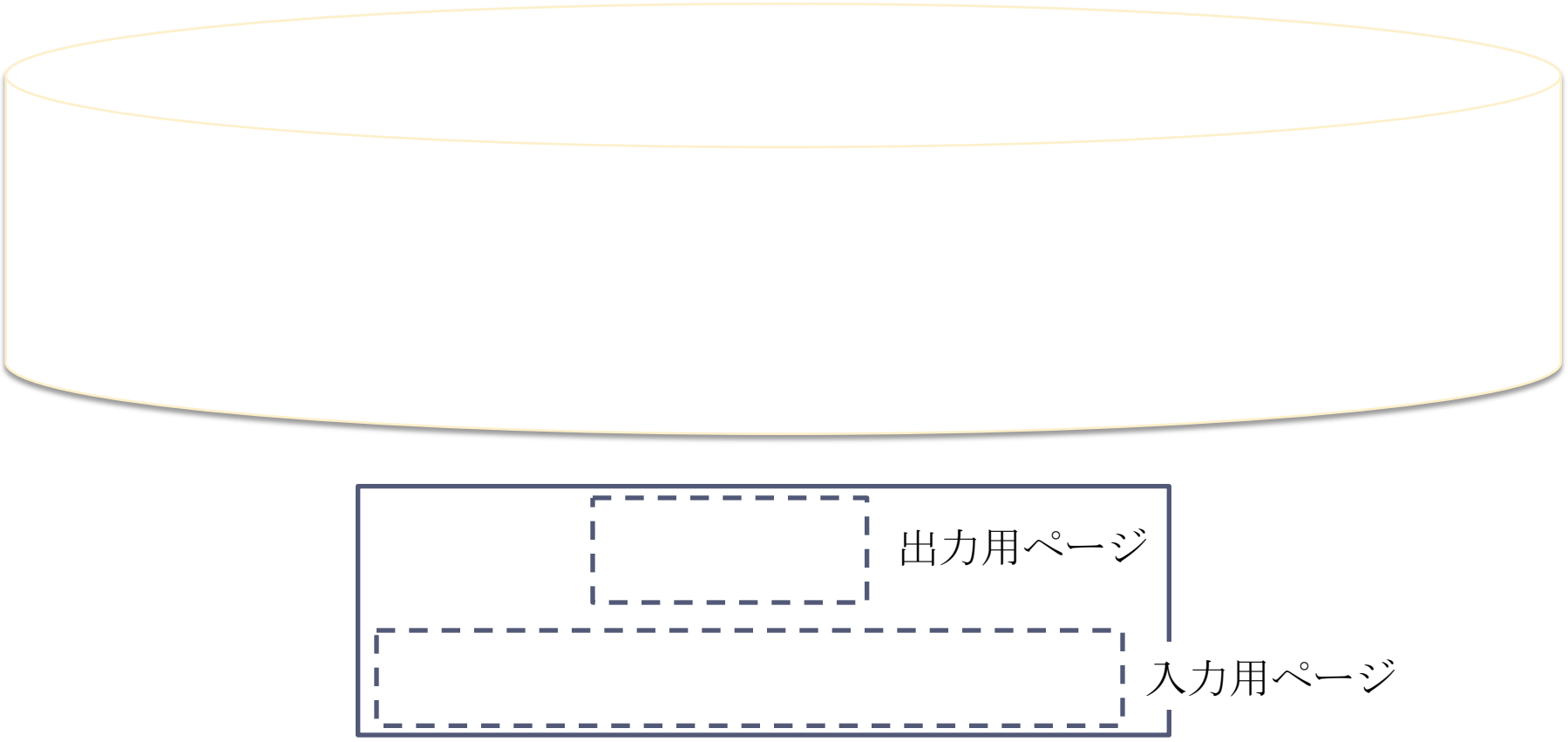
(3)小さい順に出力用ページに並べる



(1) K=M-1個分のレベル1のランをレベル2の「ラン」とする
(2) レベル2内の各レベル1ランから1ページずつ呼び出す



K-wayマージソートの手順(続きをやろう)



外部ソートのIOコスト

- 外部ソートのIOコスト

$$2N\log_B N$$

- 理由
  - 1回のソートにつきNページを読み込み，並べ替えたページをNページ書き込むのでIOコストは2N
  - ソートを繰り返す回数は $\log_B N$
  - なのでソートのIOコストは $2N\log_B N$