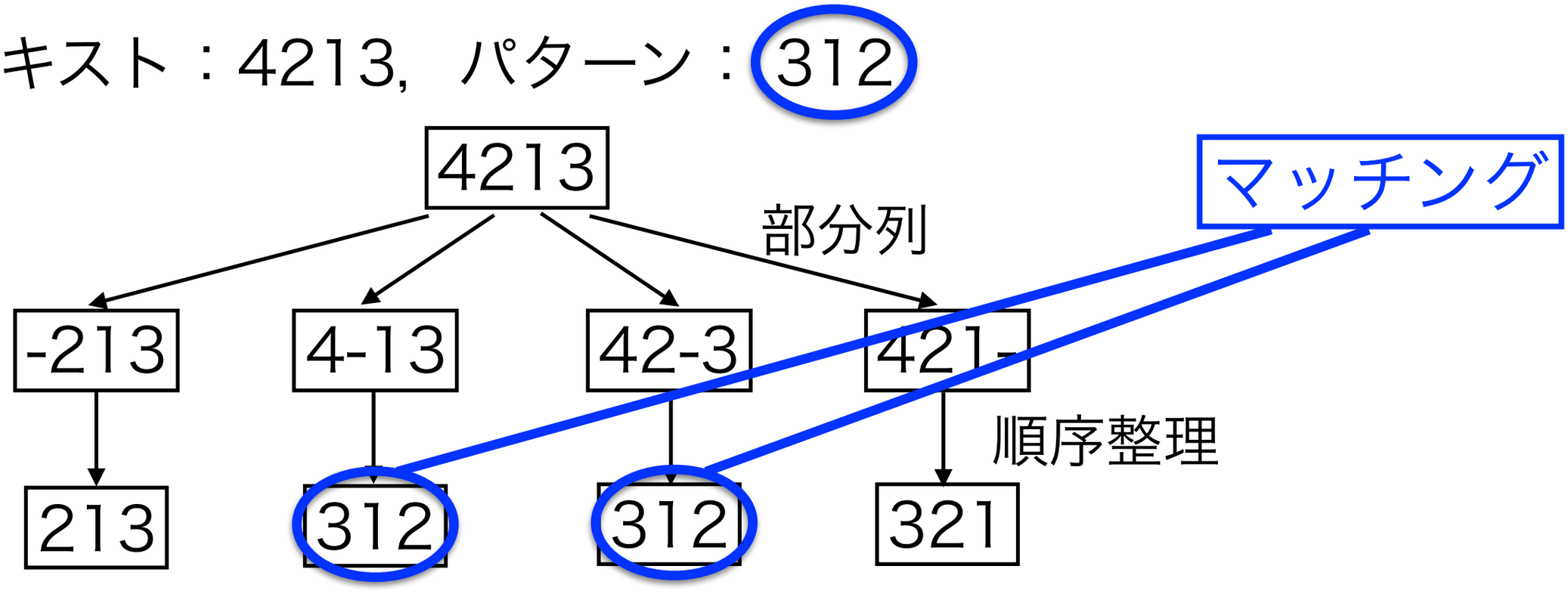
順列の共通パターン列挙

北海道大学　大学院情報科学研究科
情報理工学専攻　大規模知識処理研究室
博士3年　井上 祐馬

順列とパターン

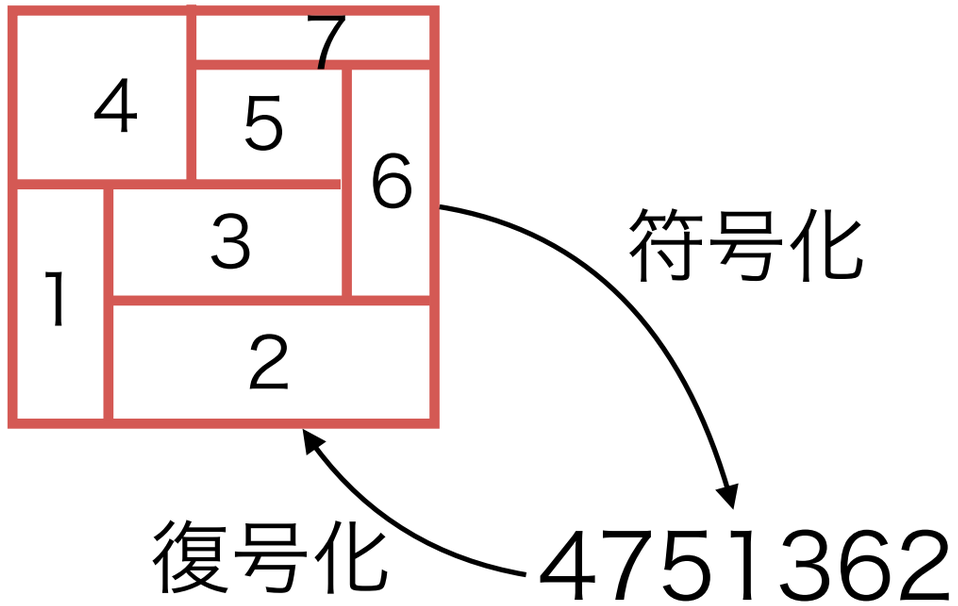
- 順列とは，{1,2,…,n}からちょうど1つずつ数字を選び、任意の順に並べた数列である (例) 52413
- 順列パターンマッチング：
  - マッチング箇所は連続列でなくてよい
  - 絶対値ではなく，相対的な順序関係が一致

例) テキスト：4213，パターン：312

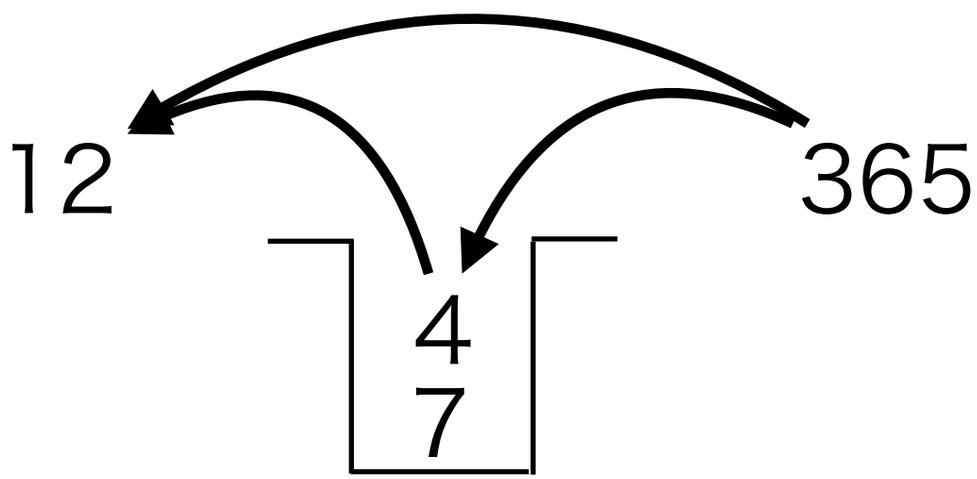


パターンによる順列の特徴づけ

- フロアプラン [Yao et al.03]
  - Baxter Permutationと対応
    (ある2つのパターンを回避)



- スタックソート [Knuth68]
  - スタック1つだけを使うソート
  - ソート可能 ⇔ 231を回避



順列パターンは順列の特徴を解析するのに役立つ

本研究における問題

与えられた2つの順列に対し、
共通して現れるパターンを全て列挙する

例) 入力: (4, 1, 2, 3), (3, 1, 2, 4)

出力: { (1), (1, 2), (2, 1), (1, 2, 3), (3, 1, 2) }

- ある順列πが別の順列τのパターンか判定する問題はNP完全 [Bose+ 98]
- 2つの順列π, τの最長共通パターンを求める問題もNP完全 (↑から言える)
- 2つの順列π, τのうち、1つが separable permutation の場合、最長共通パターンを求める $O(n^7)$ が存在 [Bouvel+ 06]

提案法の流れ

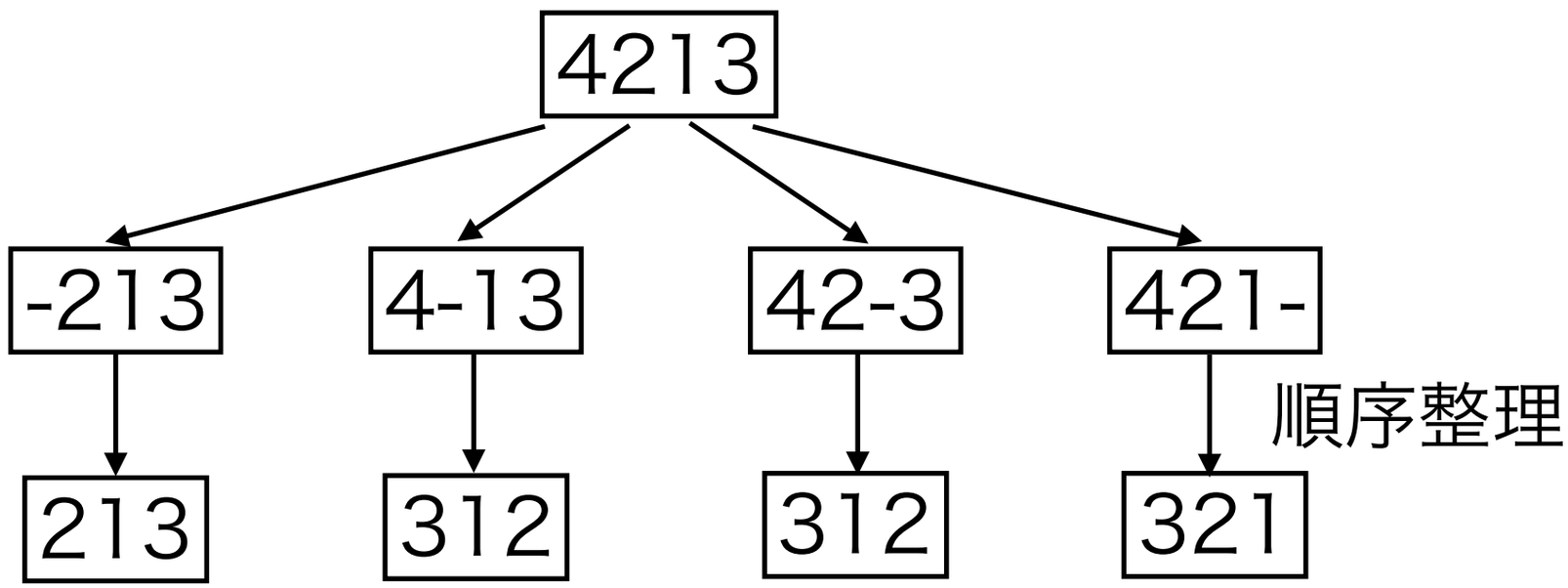
- 長さnのパターン ($=\pi$) のみからなる集合を表すRot-πDD P_n を用意
- k = n から 2 まで降順に、
  - i (1 ≤ i ≤ k) 番目に k を挿入した順列からなる集合を表すRot-πDD S_k を作成
  - $P_{k-1} \leftarrow P_k \times S_k$
  - P_{k-1} 中の順列の最大値を右寄せをする演算を行う
- これで各 P_k に長さkのパターンが入っている

- τ についても T_k を求めると、各 k について $P_k \cap T_k$ が長さ k の共通パターン

提案法のアイデア

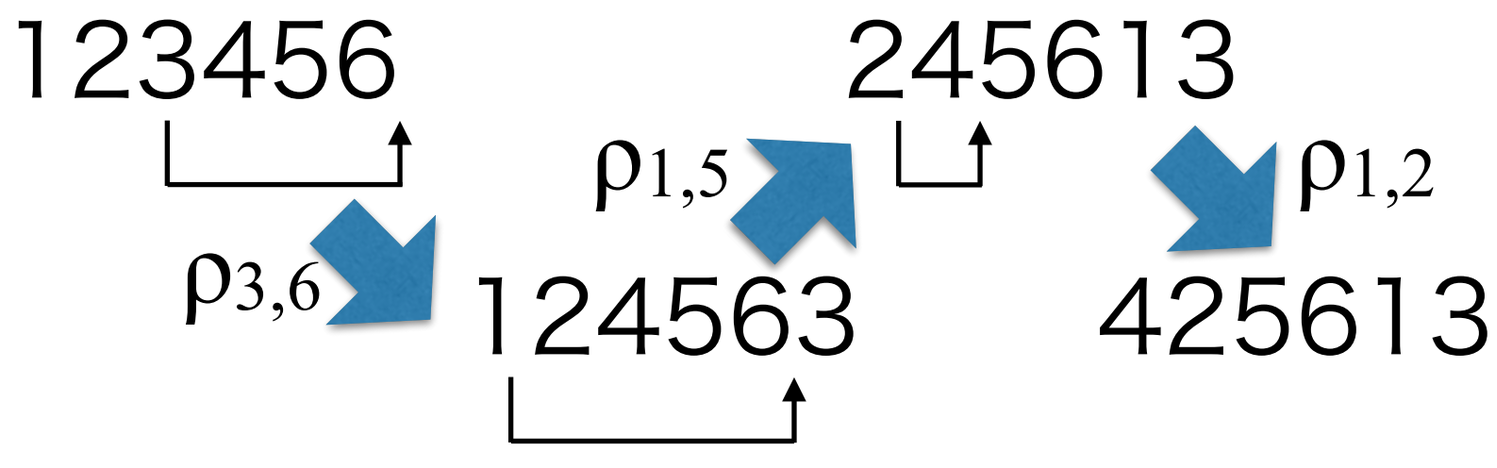
- 長さkのパターン集合から長さk-1のパターン集合を求める
  - 長さkのパターンを1つずつ見て、k個の数のうちどれか1つを抜き、順序を付け直す
    → 長さk-1のパターンになる
  - 長さkのすべてのパターンについて行えば、長さk-1のすべてのパターンが列挙できる

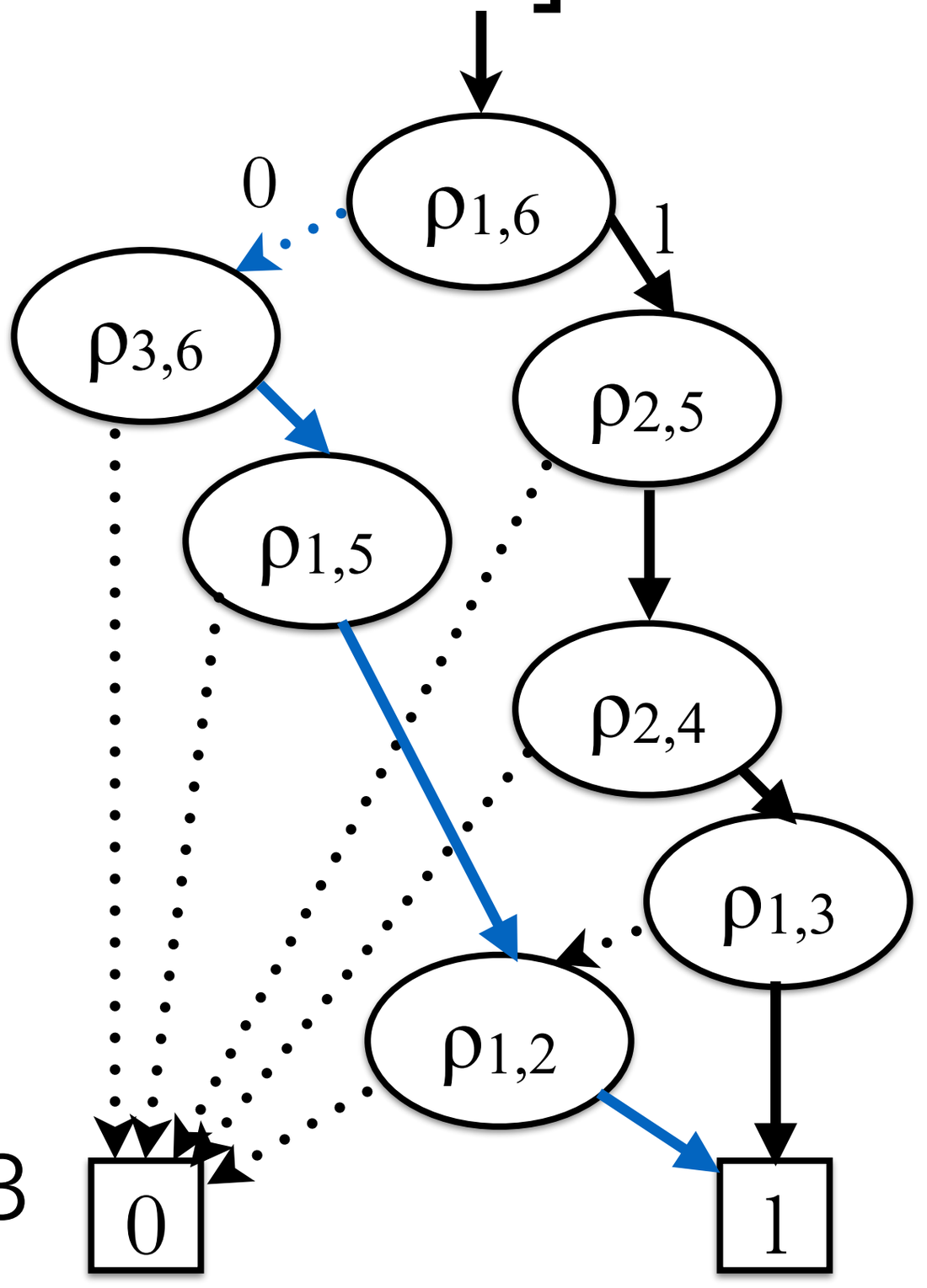
まとめてやりたい！
Rot-πDDを使おう！



Rot-πDD [Inoue+ 15]

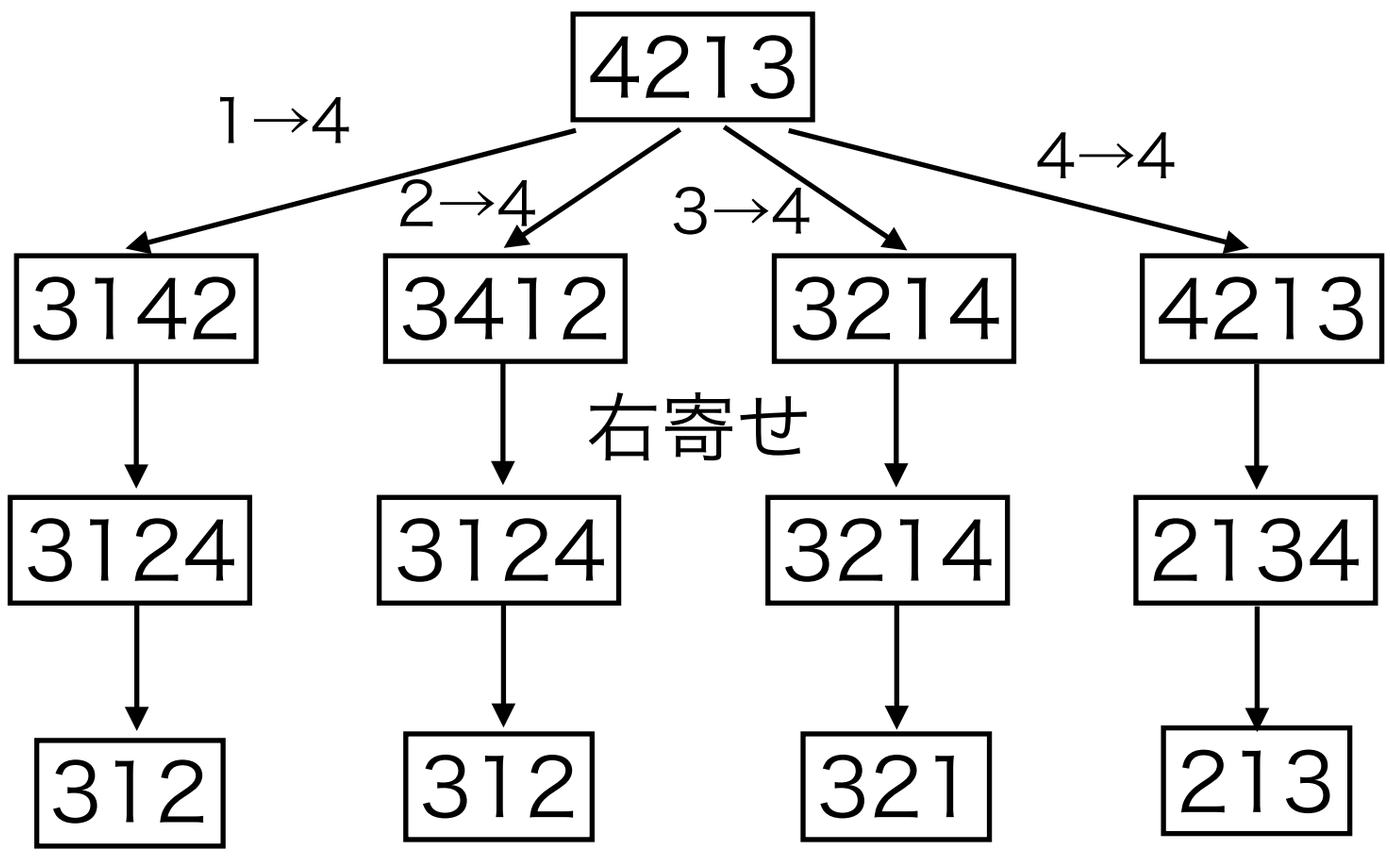
- Rot-πDD:
  順列集合を表す二分決定グラフ
  - 節点: **左ローテーション**
  - 1-終端節点へのパス: 1つの順列
  - 1つの節点に2つの枝:
    - 1-枝 (始点の節点を使う)
    - 0-枝 (始点の節点を使わない)



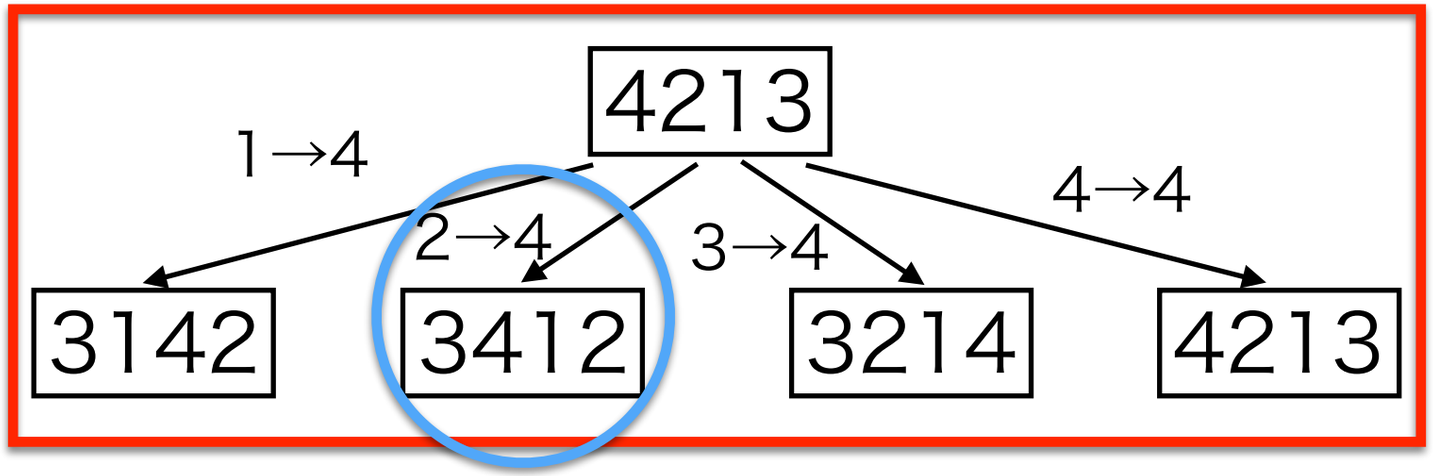


順列から要素を1つ抜く方法

- それぞれの順列に含まれる i (1 ≤ i ≤ k) を最大値 k に置き換え、それ以外の i 以上の要素をすべて 1 減らす
- すべての順列の k を一番右に寄せて、取り除く

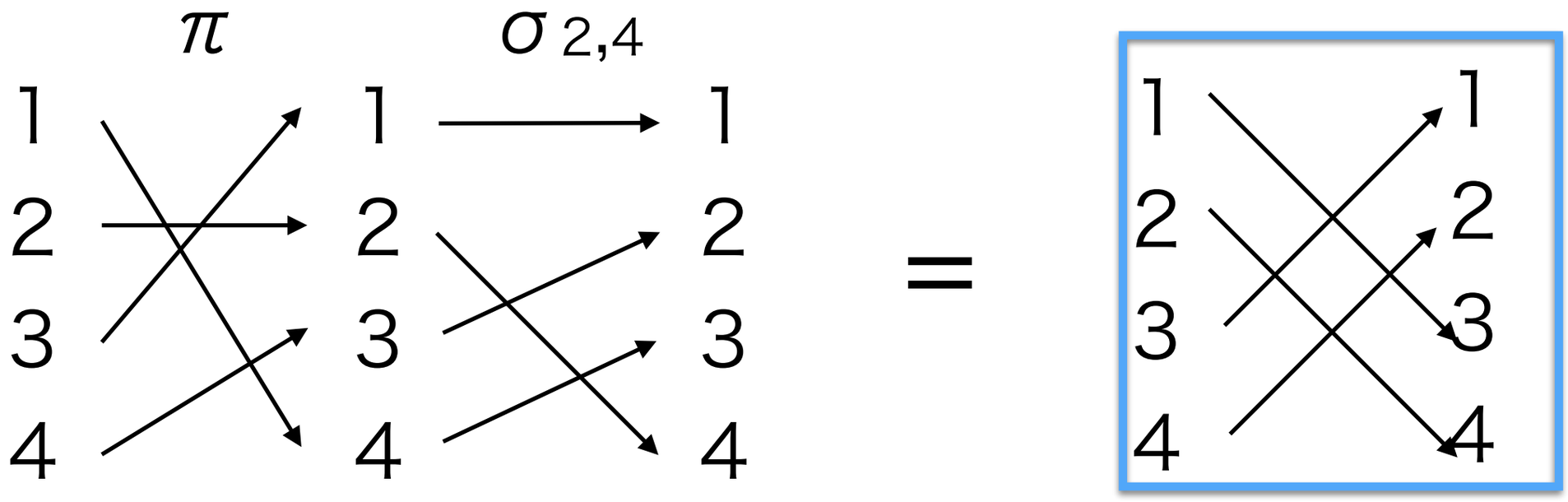


置き換え&減算

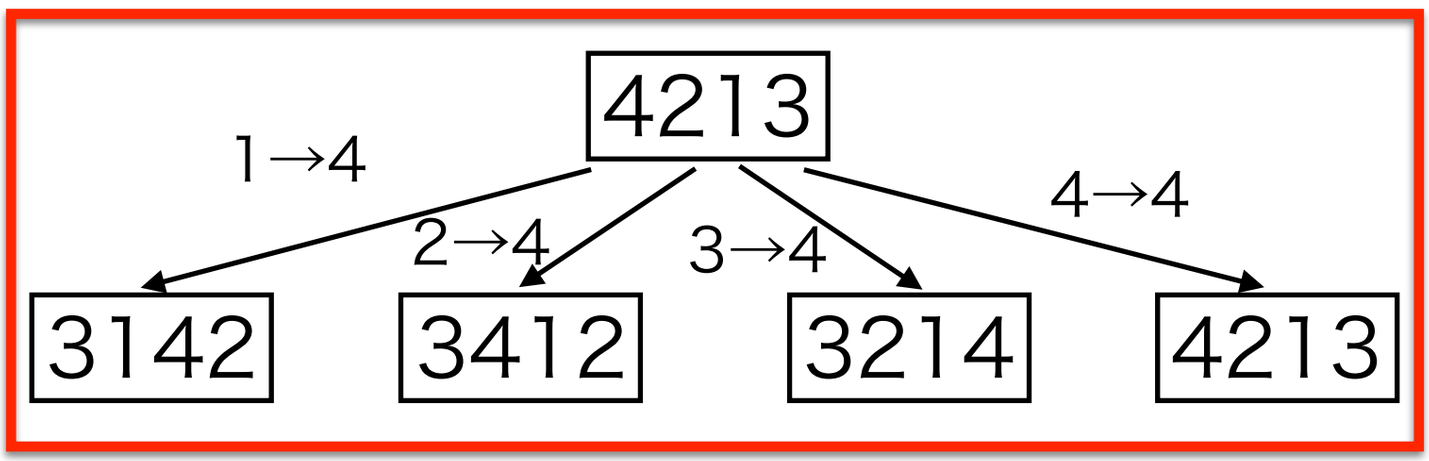


i 番目に k を挿入した順列 $\sigma_{i,k}$ をその順列 π に合わせて並べ替えれば作れる = 順列合成で実現できる

例) i=2 を k=4 で置き換えて 2 以上を減算



置き換え&減算



i 番目に k を挿入した順列は以下のローテーション

の繰り返しで実現できる

$(1, \ldots, k, i, i+1, \ldots, k-1) = \rho_{k-1,k} \ldots \rho_{i,i+1}$

結局…

直積演算

長さkのパターン集合

×

Rot-πDD

$\{\rho_{k-1,k}$,

$\rho_{k-1,k}\rho_{k-2,k-1}$,

… ,

$\rho_{k-1,k} \ldots \rho_{1,2}\}$

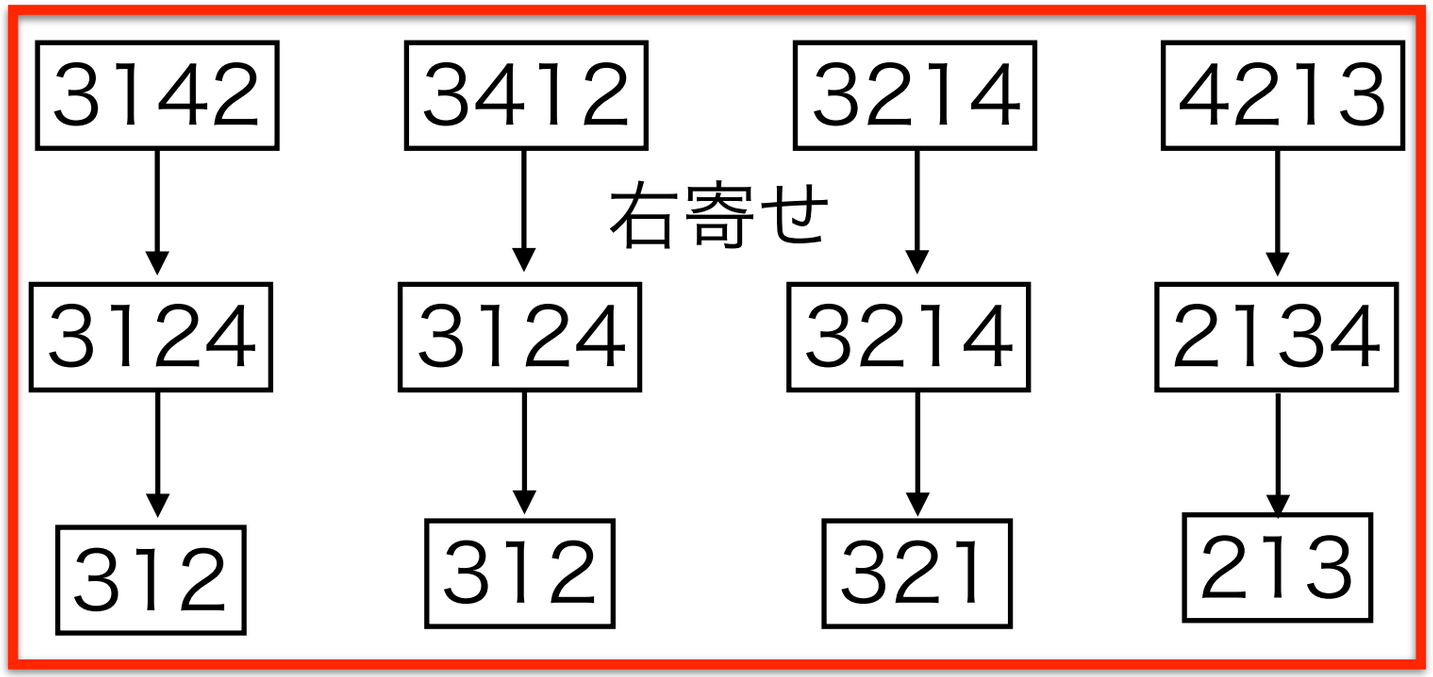
＝

Rot-πDD

長さk-1のパターン集合

で計算できる

最大値の右寄せ



- ローテーション分解する際、最大値を最初から動かさなければずっと右にある

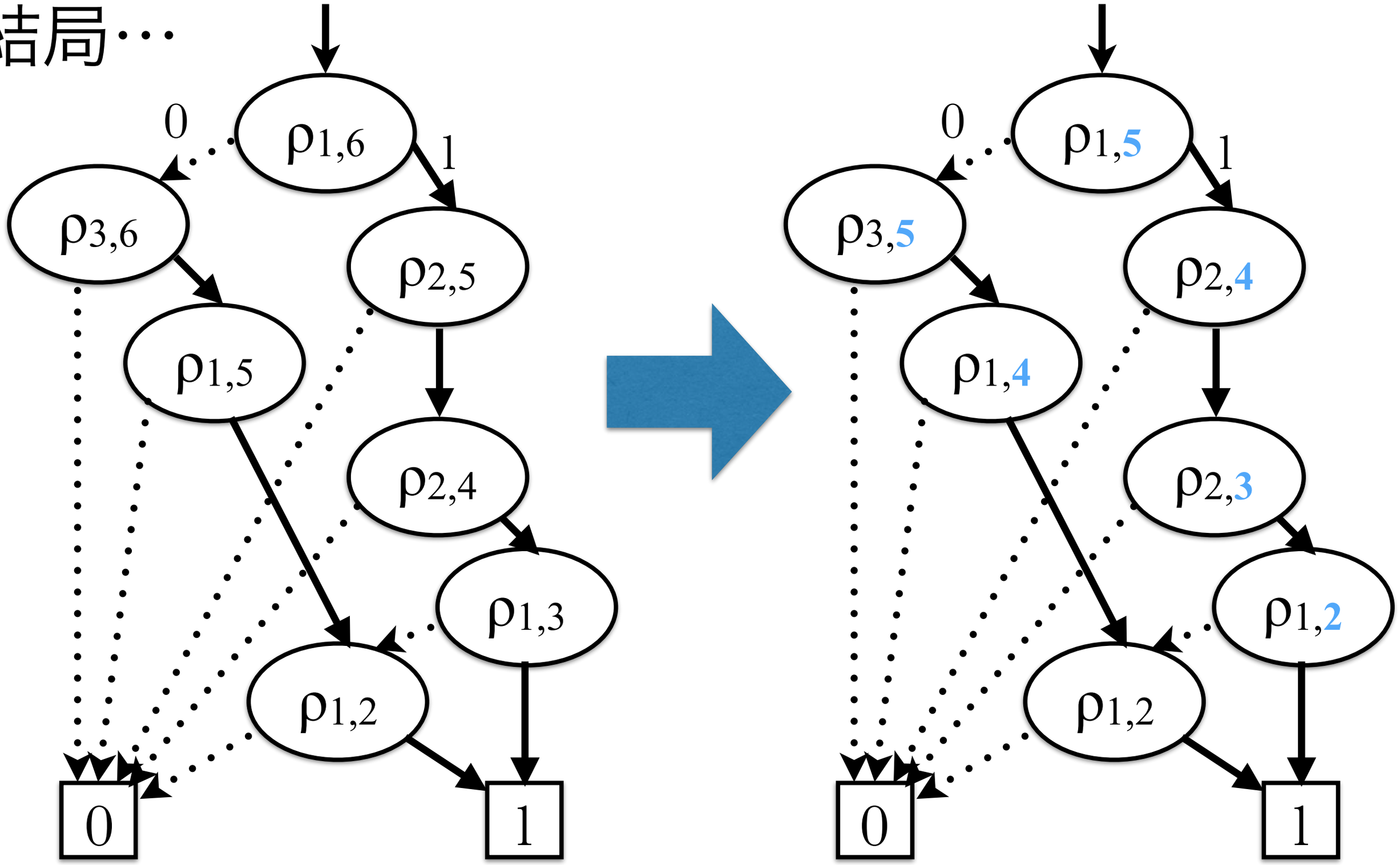
$(3, 1, 4, 2) = \rho_{2,4}\ \rho_{1,2}$ → $(3, 1, 2, 4) = \rho_{2,3}\ \rho_{1,2}$

$(3, 4, 1, 2) = \rho_{2,4}\ \rho_{1,3}$ → $(3, 1, 2, 4) = \rho_{2,3}\ \rho_{1,2}$

- 実は、ρ の右の添え字が k から連続しているところまでデクリメントするだけでOK

最大値の右寄せ

結局…



実験

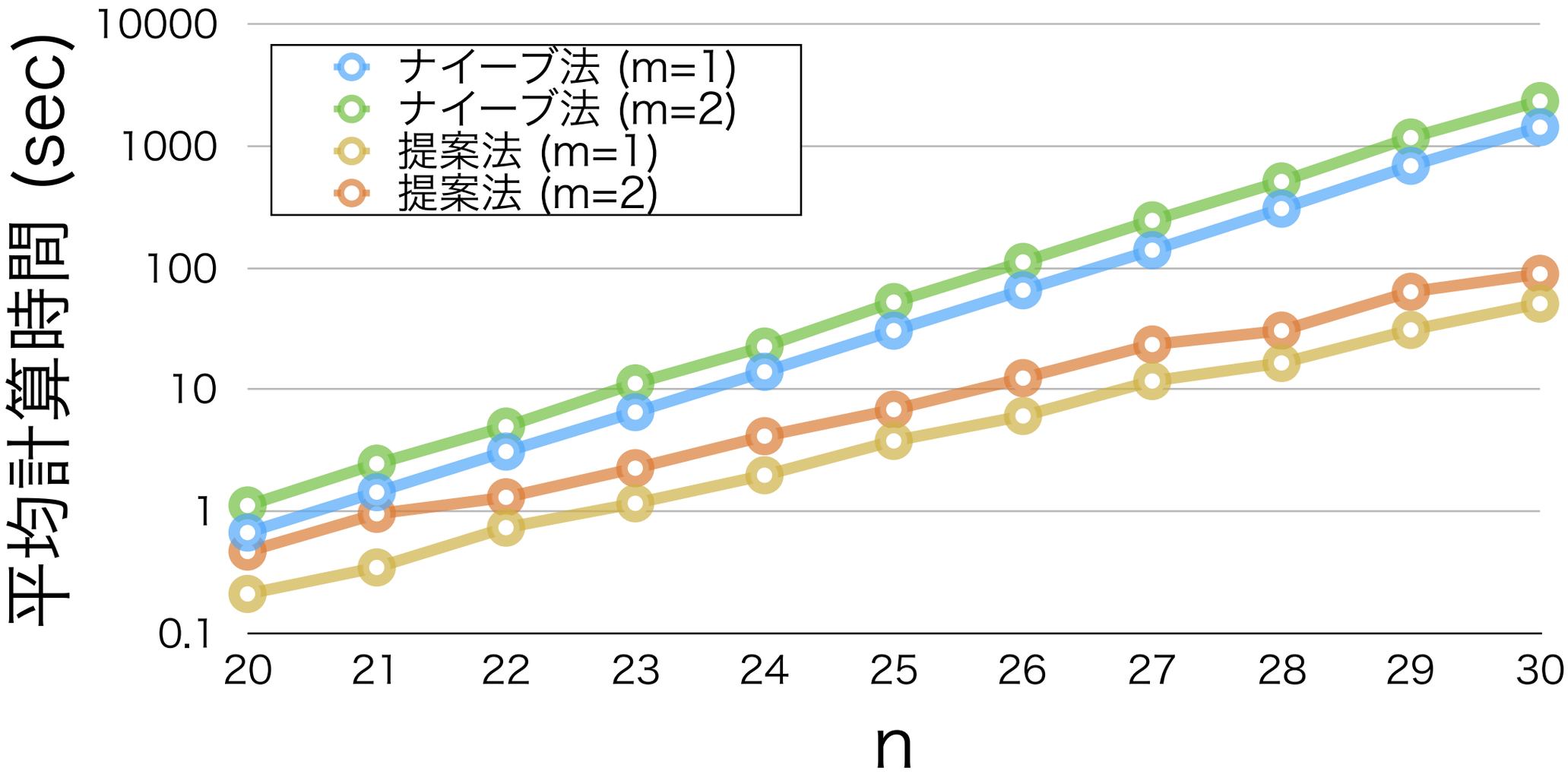
実験設定

- 順列の長さ n のランダムな順列を1 or 2個生成
- 1個: 全パターンを計算、2個: 共通パターンを計算
- 各10ケースの平均時間・メモリ使用量を計測

比較対象：ナイーブ法

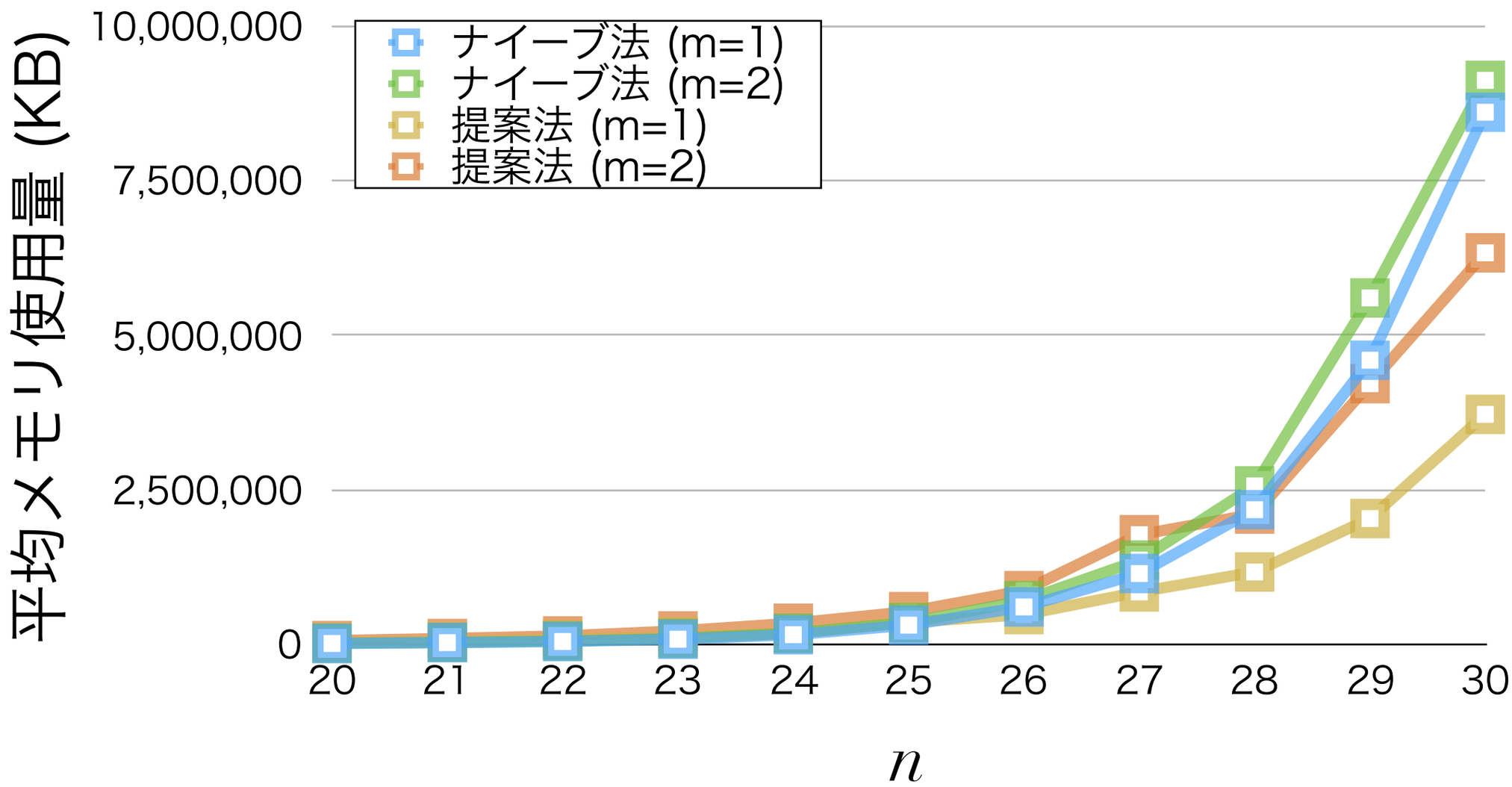
- 全部分列 (2^n 通り) を調べ、順序を整理したものを1つずつ集合 (std::set) に追加
- m 個の順列のパターンの共通を std::set.count() で調べ、共通でないものを排除していく
- 最悪計算量 O(m 2^n n logn) だが、短いパターンから調べて共通パターンがなくなった長さで打ち切れる

実験結果 (実行時間)



- 長い順列ほど相対的に高速
- ナイーブ法より 5~20 倍程度高速

実験結果(使用メモリ)



- 1つの順列のパターン列挙だけならそのまま保持するより省メモリ
- 2つの順列の共通パターンになると個数/長さが一気に減るため圧縮効果が薄くなる

まとめ

- Rot-πDDを用いた2つの順列の共通パターンを求めるアルゴリズムを提案
  - Rot-πDDの演算を利用したアルゴリズム
  - 1つ、あるいは3つ以上の共通パターンでも使える
  - ランダムな順列に対して、ナイーブ法より10倍ほど高速でそれほどメモリも使わない

- 今後の課題
  - 思いついたのでやってみたが、応用がなさすぎる
  - separable permutation に対する $O(n^7)$ のアルゴリズムが改善できそうな気持ちなので考えている