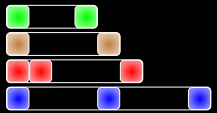
疎行列圧縮による二重対数行列幅の実現

Dominik Köppl[1] Vincent Limouzy[2] Andrea Marino[3] Giulia Punzi[4]
Jannik Olbrich[5] Takeaki Uno[6]

[1]: University of Yamanashi
[2]: University Clermont Auvergne
[3]: University of Florence
[4]: University of Pisa
[5]: University of Ulm
[6]: National Institute of Informatics

game slides

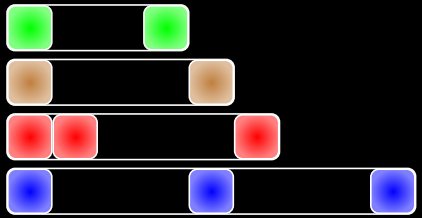


冬のLAシンポジウム2025 [28]

問題の設定

入力

- n 個の1次元ポリオミノ (=タイル)
- タイルには隙間があってもよい



目標

- タイルを1次元結合タイルにする
- 隙間を埋めることができるが、埋めたブロックは重なってはならない
- 目的: 最短の結合タイルを構築

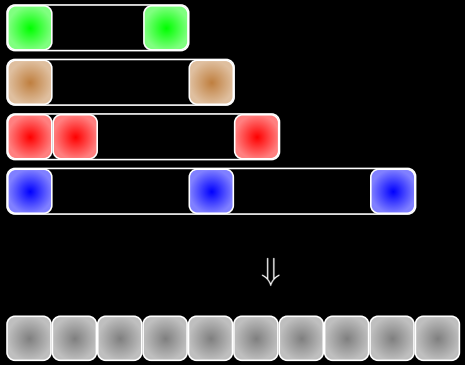
問題の設定

補題

隙間のない結合タイルは解である

Proof.

ブロックは重ならないから □



決定問題

minLength すべてのタイルを長さ k の結合タイルに結合できるか?

minMaxShift すべてのタイルの最初のブロックが最初の列に配置された場合、最大シフトが k 以下の結合タイルを作成できるか?

minMaxShift はすでに Sparse Matrix Compression (SMC) problem として研究されていた

Garey,Johnson'79 は $k \geq 2$ の場合に SMC が $\mathcal{NP}$ 困難であることを示した
英夫+'24 は幅が $\Omega(\lg n)$ の場合、両方の問題が $\mathcal{NP}$ 困難であることを示した

問題 (SMC, [Garey,Johnson'79, Chapter A4.2, Problem SR13])

入力：
- $n \times \ell$ 行列 $A[1..n][1..\ell]$、n 行 ℓ 列として、すべての $i \in [1..n]$, $j \in [1..\ell]$ に対して $A[i][j] \in \{0,1\}$ の要素を持つ
- 整数 $k \in [0..\ell \cdot (n-1)]$

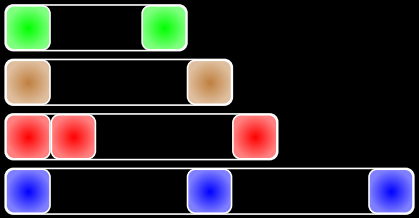
目標: 以下の2つが存在できるかどうかを調べる:

- 整数配列 $C[1..\ell+k]$、$\forall i \in [1..\ell+k]$ について $C[i] \in [0..n]$
- シフト関数 $s : [1..n] \to [0..k]$ が存在し、$\forall i \in [1..n]$, $\forall j \in [1..\ell]$ について $A[i][j] = 1 \Leftrightarrow C[s(i)+j] = i$ が成り立つ
- $A[0][j] = 0 \ \forall\, j$ であると仮定し、$C[i] = 0$ を設定することで、この要素が未割り当てであることをモデル化する

応用：

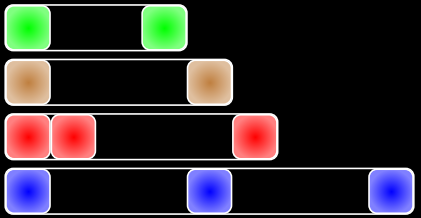
- 行列の圧縮 Ziegler'77
- 探索トライの実装 Tarjan,Yao'79
- コンパイラ Aho+'86
- ブルームフィルタ Chang,Wu'91

タイルから行列へ



$$A = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

タイルから行列へ



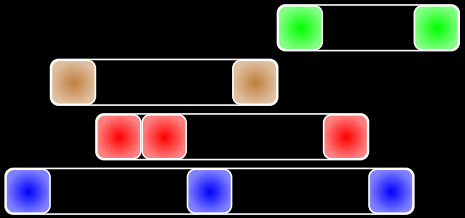
$$A = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

解答：

- $s = [6, 1, 2, 0]$
- $C = [4, 2, 3, 3, 4, 2, 1, 3, 4, 1]$

$$B = \begin{pmatrix} & & & & & & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & & & & & \\ & & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & & & & \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & & & & & & \end{pmatrix}$$

タイルから行列へ



$$A = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

解答：

- $s = [6, 1, 2, 0]$
- $C = [4, 2, 3, 3, 4, 2, 1, 3, 4, 1]$

$$B = \begin{pmatrix} & & & & & & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & & & & & \\ & & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & & & & \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & & & & & & \end{pmatrix}$$

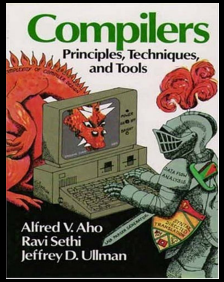
近似アルゴリズム

Ziegler'77: 貪欲アルゴリズム: 先頭から配置

- 最初のタイルを最初の位置に配置
- それ以降のタイルについて: 最も左に適合する位置に配置
- 繰り返す

使用例: "Compilers: Principles, Techniques, and Tools" の節 3.9.8

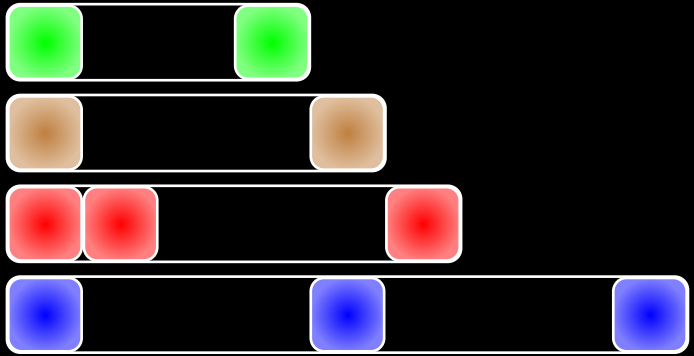
While we may not be able to choose *base* values so that no *next-check* entries remain unused, experience has shown that the simple strategy of assigning *base* values to states in turn, and assigning each *base*[*s*] value the lowest integer so that the special entries for state *s* are not previously occupied utilizes little more space than the minimum possible.



近似アルゴリズム

Ziegler'77: 貪欲アルゴリズム: 先頭から配置

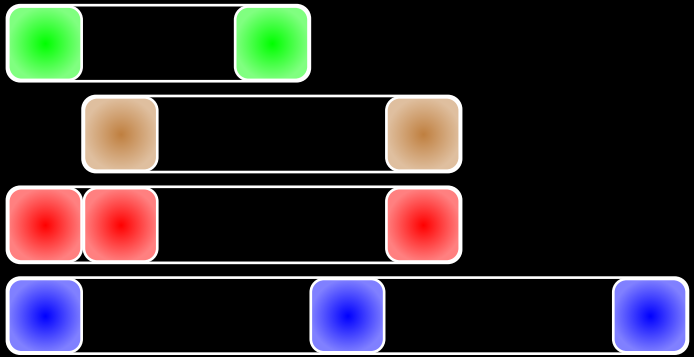
- 最初のタイルを最初の位置に配置
- それ以降のタイルについて: 最も左に適合する位置に配置
- 繰り返す



近似アルゴリズム

Ziegler'77: 貪欲アルゴリズム: 先頭から配置

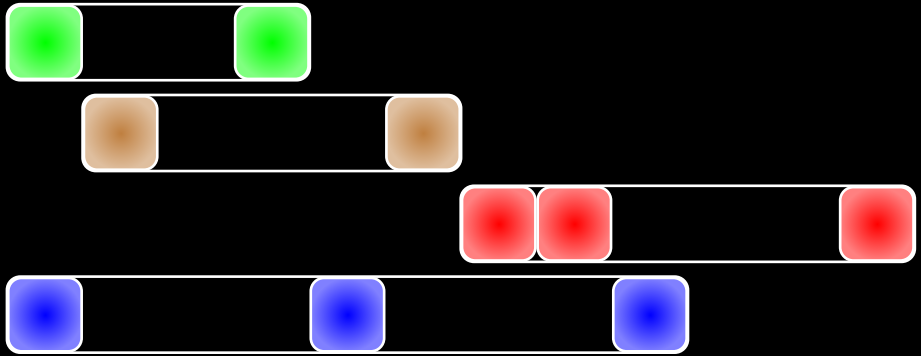
- 最初のタイルを最初の位置に配置
- それ以降のタイルについて: 最も左に適合する位置に配置
- 繰り返す



近似アルゴリズム

Ziegler'77: 貪欲アルゴリズム: 先頭から配置

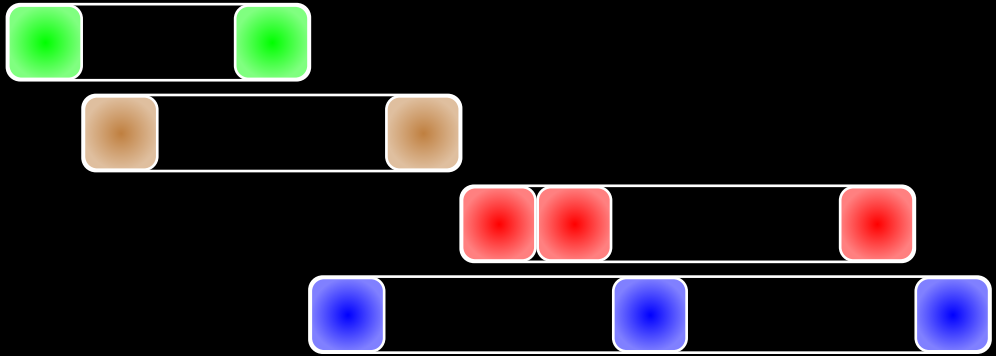
- 最初のタイルを最初の位置に配置
- それ以降のタイルについて: 最も左に適合する位置に配置
- 繰り返す



近似アルゴリズム

Ziegler'77: 貪欲アルゴリズム: 先頭から配置

- 最初のタイルを最初の位置に配置
- それ以降のタイルについて: 最も左に適合する位置に配置
- 繰り返す



近似アルゴリズム

Ziegler'77: 貪欲アルゴリズム: 先頭から配置

- 最初のタイルを最初の位置に配置
- それ以降のタイルについて: 最も左に適合する位置に配置
- 繰り返す

- 近似比は本当に小さいか?
- 実際に、近似比は $\Theta(\sqrt{m})$, ここで m は最適値! Köppl+'25

本発表

どの条件で多項式時間に最適に解けるか?

- タイルの種類が1つの場合: $\ell \in O(\lg n)$ まで
- 一般の場合: $\ell \in O(\lg \lg n)$ まで

影響があるのか？

ヒューリスティックはタイルの事前ソートを可能にする

- none: ソートなし
- incFreq: 1の個数の昇順でソート
- decFreq: 1の個数の降順でソート
- incDens: 密度$\frac{1\text{の個数}}{\text{長さ}}$の昇順で
- decDens: 密度の降順でソート
- random: 10回ランダムにシャッフルし、最短の結果を取る

悪い入力

- 2種類のタイル X と Y
- 整数パラメータ c と g
- $X = (10^g)^c 1$, $Y = (10^{g-1})^c 10^c 1$
- $|X| = |Y| = c(g+1)+1$
- $\#_1 X = \#_1 Y = c+1$
- 交互順序でn個のタイルを要求 $X, Y, X, Y, \ldots$.

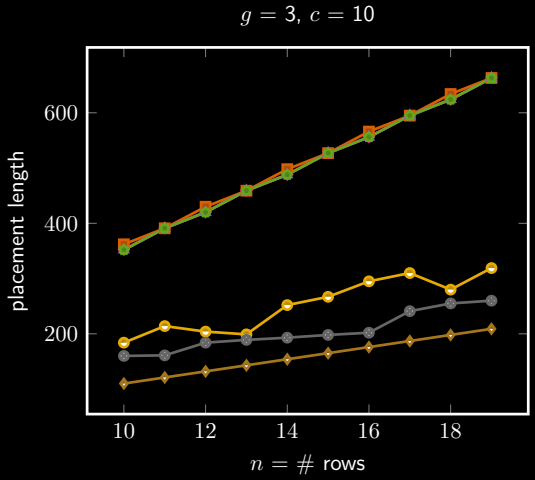
$(c = 5,\ g = 4)$

悪い実例 1/2

- lowerbound: 1の総数
- dp: 最適解

randomが良い理由

- 選択するタイルの種類が少なすぎる
- この実例の最良解は $X, \ldots, X, Y, \ldots, Y$ の順でソート



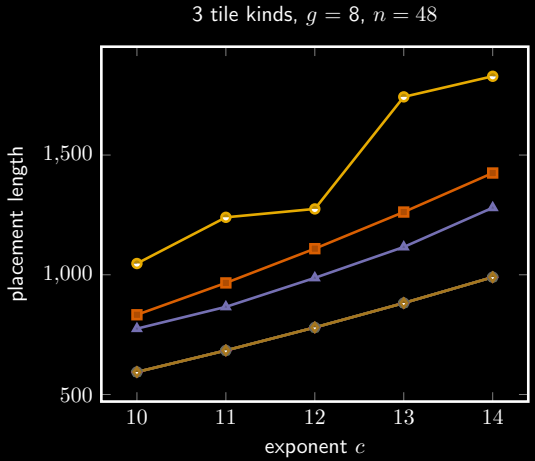
悪い実例 2/2

新しいタイル種類：

- $Z = (10^{g-2})^c 10^{2c} 1$
- 順序：$X, Y, Z, X, Y, Z, \ldots$
- すべての穴を埋めるためにヘルパータイルを追加
  $\Rightarrow \mathrm{dp} = \mathrm{lowerbound}$

random は最悪の性能！

したがって：厳密アルゴリズムの研究には意味がある



1種類のタイルのアイデア

動的計画法

- すべての可能な組み合わせを見つけるためにワイルドカードとのマッチを使用
- パラメータ：
  - 使用済みタイル数 i
  - 配置の長さ ℓ の接尾辞 S
- DP 表 $D[i][S]$ を段階的に構築

DP表 $D[i][S]$

- i: 使用されたタイルの数
- S: 長さ ℓ の接尾辞
- 値：最短の配置長

部分語マッチング

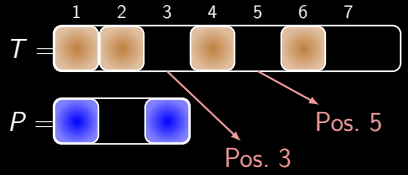
定義 (部分語)

任意の文字にマッチするワイルドカード記号 0 を含む文字列

補題 (P. Clifford,R. Clifford'07)

- 与えられた：テキスト Tとパターン P、両方は部分語である
- タスク：パターンがマッチするすべてのテキスト位置を特定
- $O(n \lg m)$ 時間で処理

- 各タイルに異なる色を与える (= 文字)
- T の接尾辞をワイルドカードで延長する



結合操作

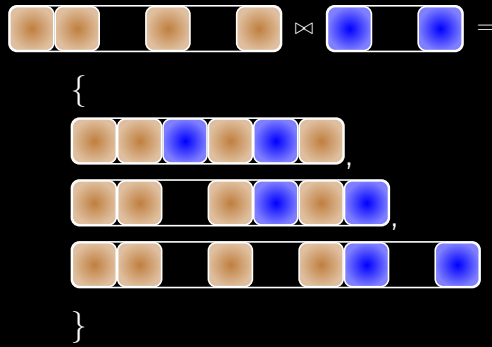
定義

2つのタイル A と B について、集合 $A \bowtie B$ はすべての A と B の有効な結合を含む

有効な結合には以下が必要：

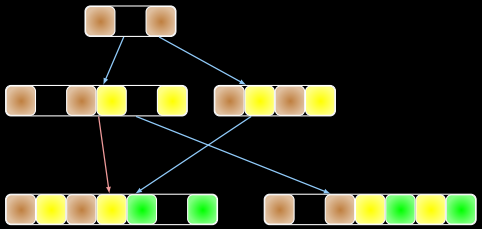
- 両方が非ワイルドカード記号を持つ位置がない
- 各位置で記号を結合

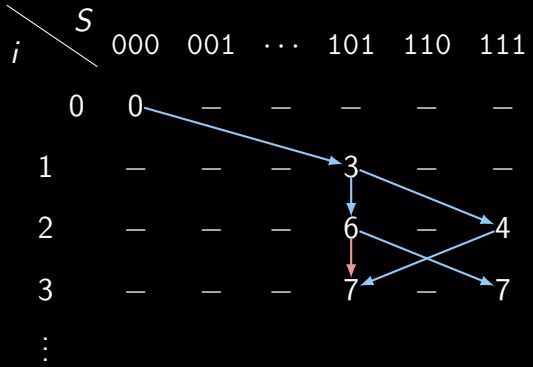
P. Clifford,R. Clifford'07 を用いて $O(\ell \log \ell)$ 時間で計算可能、かつ $|A|, |B| \in O(\ell)$



DP アルゴリズム：1種類の異なるタイル

- 設定：すべての n 個のタイルが同一、長さ $\ell = O(\lg n)$
- 例：n 個の 101 タイル





アルゴリズムの計算量

定理

タイルの種類はちょうど一つ場合、MinLength は $O(n^2 2^\ell \ell \log \ell)$ 時間で解ける、ただし n はタイルの個数、ℓ はタイルの長さ

解析

- DP 表のサイズ： $O(n \cdot 2^\ell)$ マス（各 $i = [0..n]$ と $S \in \{0,1\}^\ell$）
- 各マスについて：結合集合 $A \bowtie B$ を $O(\ell \log \ell)$ 時間で計算
- 総時間：$O(n \cdot 2^\ell \cdot \ell \log \ell)$

ℓ について固定パラメータ容易、$\ell = O(\lg n)$ のとき多項式時間：
$O(n \cdot 2^{O(\lg n)} \cdot O(\lg n) \cdot O(\lg \lg n)) = O(n \cdot n^{O(1)} \cdot \mathrm{poly}(\lg n))$

一般の場合：複数のタイル種類

入力:

- κ 種類の異なるタイル種類 $T_1, \ldots, T_\kappa$
- c_i: i 番目のタイル種類の個数、$\sum_{i=1}^{\kappa} c_i = n$

課題：タイルが異なる場合、順序が重要

素朴なアプローチ

- 順序を明示的に追跡
- $\frac{n!}{c_1! c_2! \cdots c_\kappa!}$ 通りの順序
- 小さい κ でも指数的

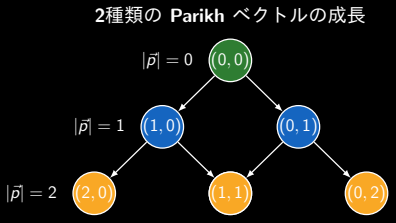
解決策：Parikh ベクトル

- 各採用されたタイル種類の個数をベクトル $\vec{p}$ で追跡
- $\vec{p} = (p_1, p_2, \ldots, p_\kappa)$ かつ $p_i \in [0..c_i]$
- 順序は追跡しない！

Parikh ベクトルでの DP

修正された DP 表： $D[\vec{p}][S]$

- $\vec{p}$: Parikh ベクトル
  （各採用されたタイル種類の個数）
- S: 長さ ℓ の接尾辞
- 値：最短配置長



アルゴリズム

1. 開始： $D[\vec{p}_0][0] = 0$ 、だだし $\vec{p}_0 = (0, 0, \ldots, 0)$
2. 各$\vec{p}$に対し、$|\vec{p}|$ を 0 から $n-1$ まで、および D 中の接尾辞 S に対し
   - 各タイル種類 T_i について、$p_i < c_i$ のとき：
     - 新しい Parikh ベクトル $\vec{p'} = \vec{p}$ を位置 i で増分して計算
     - 各結合 $Z \in S \bowtie T_i$ について $D[\vec{p'}][Z]$ を更新
3. $D[\vec{p}_n][\cdot]$ の最小値を返す、ただし $\vec{p}_n = (c_1, \ldots, c_\kappa)$

計算量：一般の場合

定理

長さ ℓ の n 個のタイルを持つ MINLENGTH は $O(n^{2^\ell}\ell n 2^\ell n)$ 時間で解ける。

解析

- タイル種類数 $\kappa \leq 2^\ell$
- Parikh ベクトルの数 $\leq n^\kappa \leq n^{2^\ell}$
- 各 Parikh ベクトルについて：$O(2^\ell)$ 個の接尾辞
- 接尾辞ごと：$O(\ell \log \ell)$ 結合時間

多項式時間

1. $\ell = O(\lg \lg n)$ のとき： $O(n^{2^{O(\lg \lg n)}} \cdot \mathrm{poly}(\lg n)) = O(n^{(\lg n)^{O(1)}} \cdot \mathrm{poly}(\lg n))$
2. $\ell = O(\lg n)$ かつ $\kappa = O(\lg n)$ 個の異なるタイルのとき：
   $O(n^{O(\lg n)} \cdot \mathrm{poly}(n)) = \mathrm{poly}(n)$

実装の詳細

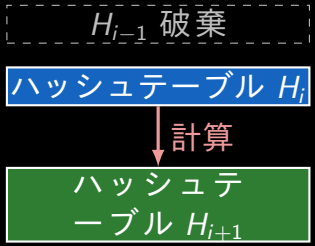
空間最適化

- DP行列全体を保存しない
- $|\vec{p}| = i$ の Parikh ベクトルにハッシュテーブル H_i を使用
- メモリには連続する2レベルのみを保持
- 空間：$O(2^\ell)$、 $O(n \cdot 2^\ell)$ や $O(n^{2^\ell} \cdot 2^\ell)$ の代わり

ハッシュテーブル構造

- キー：$(\vec{p}, S)$, 値：最短配置長
- H_i から H_{i+1} を計算, H_{i-1} を破棄

ステップiのメモリ：



メモリには2レベルのみ

まとめ

主要な結果

1. 問題は $\ell \in \Omega(\lg n)$ に対して $\mathcal{NP}$ 困難
2. 以下の場合は $\mathcal{P}$ に含まれる
   - $\ell = O(\lg n)$ の1種類のタイル
   - 以下の一般の場合
     - $\ell = O(\lg \lg n)$
     - $\ell = O(\lg n)$ で $O(\lg n)$ 個の異なるタイル

主要な技術

- 接尾辞に対する動的計画法
- 部分語マッチング
  P. Clifford,R. Clifford'07
- 順序追跡の代わりに Parikh ベクトル
- 空間のためのハッシュテーブル最適化

結論

二重対数タイル幅により多項式時間解が可能！