

ハードウェア化に適した近似文字列照合アルゴリズム

馬場, 謙介
九州大学大学院システム情報科学研究院

<https://hdl.handle.net/2324/9132>

出版情報 : SLRC プレゼンテーション, 2006-07-19. 九州大学システムLSI研究センター
バージョン :
権利関係 :

ハードウェア化に適した 近似文字列照合アルゴリズム

馬場 謙介

九州大学

大学院システム情報科学研究所

研究紹介

私の研究経歴

- 1996-2002 古典論理に対するラムダ計算
- 2002-2004 数値演算的手法を用いた文字列処理
- 2004- セキュアな認証のための論理
- 2005- **ハードウェア化に適した文字列処理**

目標: 文字列照合を(どんな手段を使っても)高速に行う



発表のあらすじ

- 近似文字列照合高速化の意義
 - アプリケーションの観点から
- 高速化の既存手法
 - 問題の定式化
 - 既存手法の特徴
- 我々のアプローチ
 - 基本アイデア
 - 効果の大まかな見積り

文字列照合のアプリケーション

- ・ 「ぴったり」を探す
 - インターネットの検索
 - ウィルス検索
 - ID照合
 - など



- ・ 「似てる」を探す
 - バイオインフォマティクス
 - 画像・動画検索
 - 異常検知
 - など

ATCCAGTACCACGAG
CACGAGGCACACACC
GAGGGCTCCAGAA
CTACCCCGCTTCGG
CCAATCTAATAACA
GGACACGCAGGACT
TACGGGGGCGGCG

アプリケーションからの要求(1)

- ・ インターネットの検索
 - 検索のためのデータ構造を作成
 - 様々な観点からの検索対象の制限(速さの確保と感度の低下)
- ・ ウィルス検索
 - 基本的に単純な文字列の照合
 - できるだけ高速にできた方が良く, 感度は落とせない
- ・ ID照合(ICカード等による個人の識別や認証)
 - 暗号化などにより, 照合に複雑な処理を伴う場合がある
 - 時間が極端に制限される場合がある(改札など)
 - 基本的に感度は落とせない

アプリケーションからの要求(2)

- ・ **バイオインフォマティクス**
 - 一般に計算量が膨大
 - 感度を落とした高速化が一般的だが...
 - さらに複雑な構造の解析が求められる
- ・ **画像・動画検索**
 - 高次の文字列検索とみなせる
 - 量子化や低周波の除去による速さの確保と感度の低下
- ・ **異常検知(防犯カメラ, 医療画像など)**
 - 過去のデータからのパターン抽出
 - 感度が落とせない場合が多い

文字列照合高速化の意義

今さら、まだ文字列照合？

- 現実には
 - アプリケーションの中に
 - 速さと感度の更なる向上を求める
 - 速さ、または感度が落とせないもの(で重要なもの)が(まだまだ)ある
- 信念として
 - 基礎的な理論はより複雑な問題にも適用できるはずだ
 - いずれにせよ、コンピュータで扱うのは文字列だ
 - そもそも、人間が物事を記号の列によって理解しようとするアプローチこそが本質的だ

文字列照合の種類

文字列照合

2つの文字列について、片方の(テキストと呼ばれる, 長い)文字列中に、もう一方の(パターンと呼ばれる, 短い)文字列の出現を見つけること

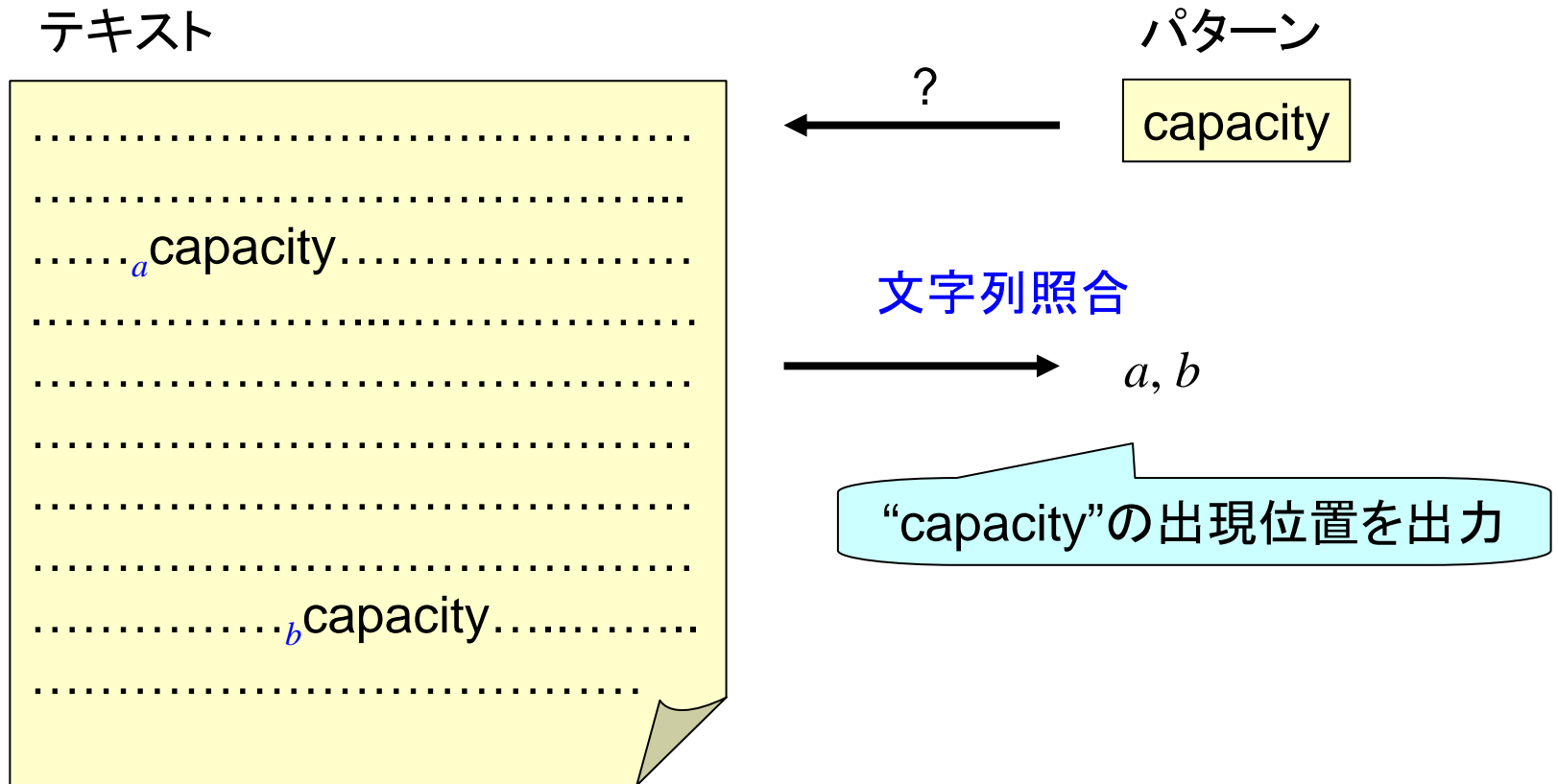
不一致を許した文字列照合

文字の置換を考慮した文字列照合. 本質はテキストの各位置でのパターンとの一致の数を数えること

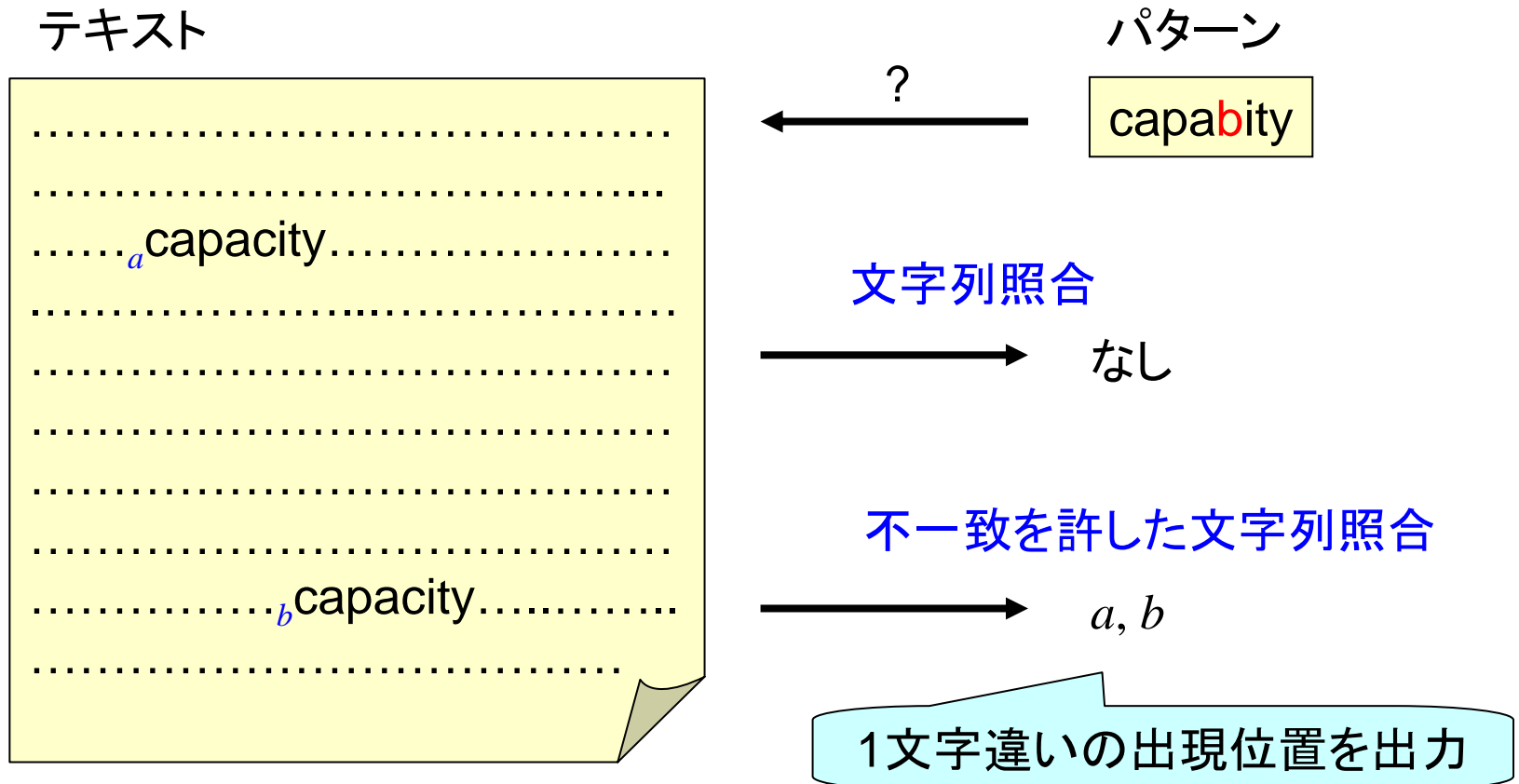
近似文字列照合

文字の置換や挿入・削除を考慮した文字列照合. 本質は文字列間の距離を計算すること. 重みを付けることで, さらに一般的な距離を考えることができる

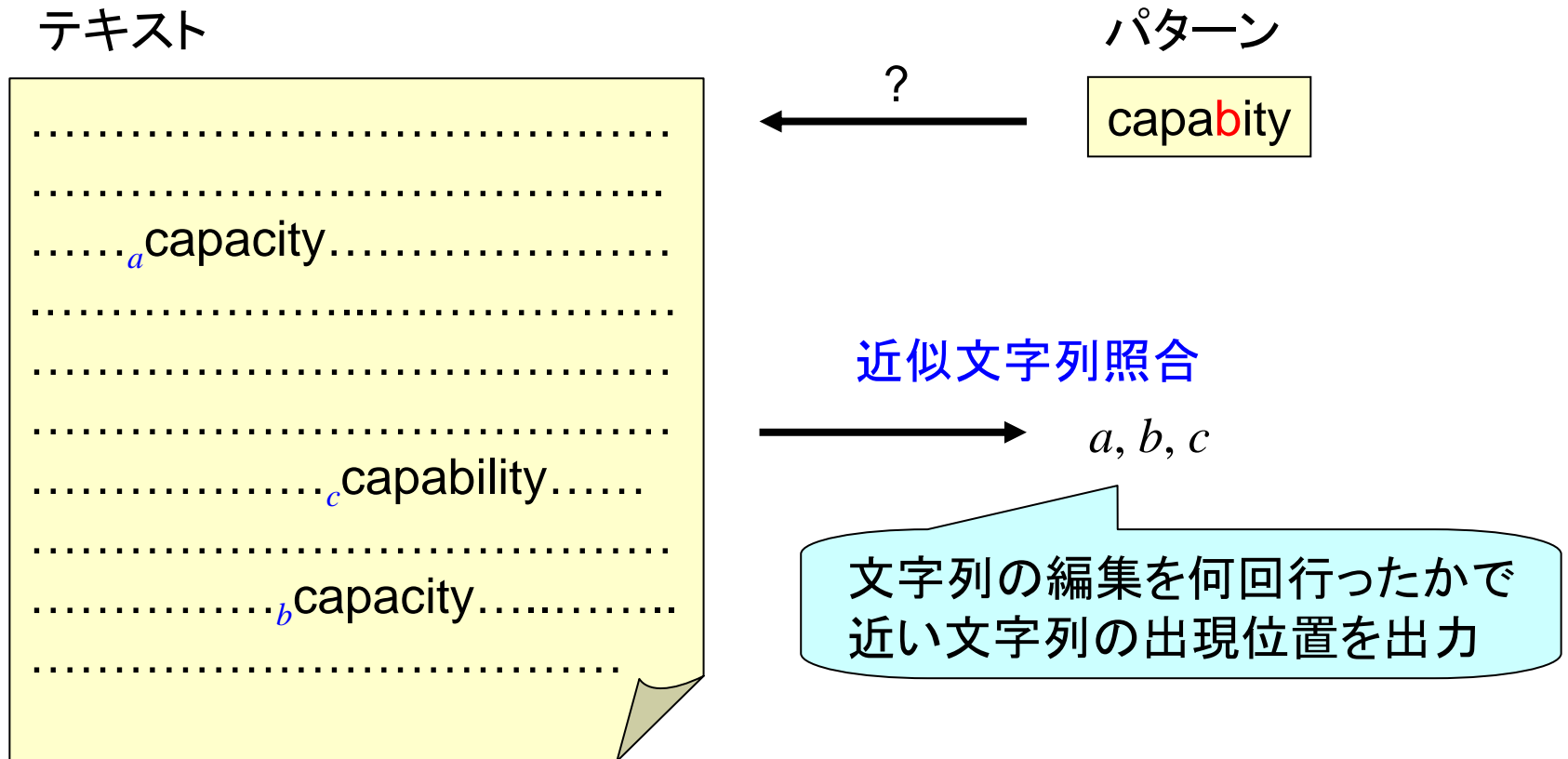
文字列照合の例(1)



文字列照合の例(2)



文字列照合の例(3)



文字列間の距離の計算

m : パターンの長さ

n : テキストの長さ

- 単純なやり方
 - 考え得る全ての文字の比較の仕方について編集の回数を数え、最小のものを見つける
 - 膨大な計算量
- うまいやり方 (動的計画法)
 - 短い部分文字列についての値を使って再帰的に計算
 - $O(mn)$ 時間

E	N	T	R	Y	-	-	-	-	-
-	-	-	-	-	E	M	P	T	Y

E	N	T	R	Y	-	-	-	-	-
-	-	-	-	-	E	M	P	T	Y

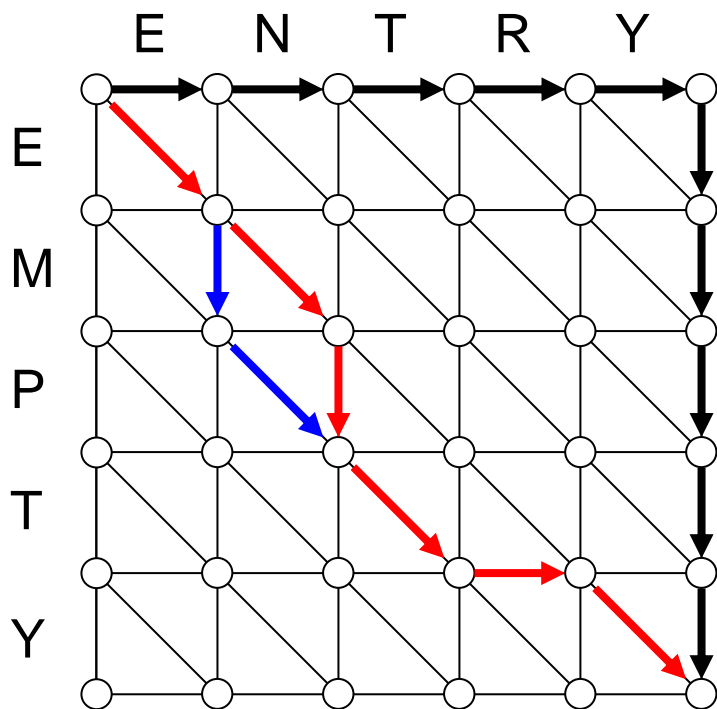
E	N	T	R	-	Y	-	-	-	-
-	-	-	-	E	-	M	P	T	Y

⋮

-	-	-	-	-	E	N	T	R	Y
E	M	P	T	Y	-	-	-	-	-

動的計画法(1)

- 与えられた文字列からグラフを作成
- 左上からの各パスが文字の比較の仕方に対応



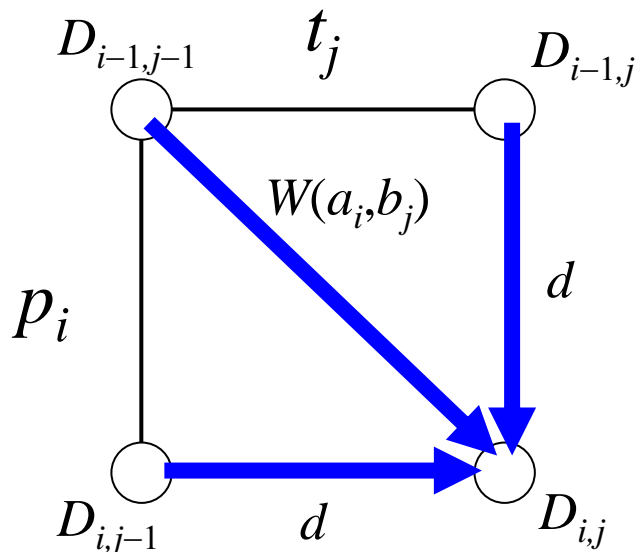
E	N	T	R	Y	-	-	-	-	-
-	-	-	-	-	E	M	P	T	Y

E	N	-	T	R	Y
E	M	P	T	-	Y

E	-	N	T	R	Y
E	M	P	T	-	Y

動的計画法(2)

- グラフの各ノードに対応する値 $D_{m,n}$ を計算する
 - $W(a,b)$: 文字 a と b の間に与えられる置換の重み
 - d : 挿入・削除の重み
 - p_i : パターンの i 番目の文字, t_j : テキストの j 番目の文字



$$D_{i,0} = id \quad (1 \leq i \leq m)$$

$$D_{0,j} = jd \quad (1 \leq j \leq n)$$

$$D_{i,j} = \min \begin{cases} D_{i-1,j-1} + W(p_i, t_j) \\ D_{i-1,j} + d \\ D_{i,j-1} + d \end{cases}$$

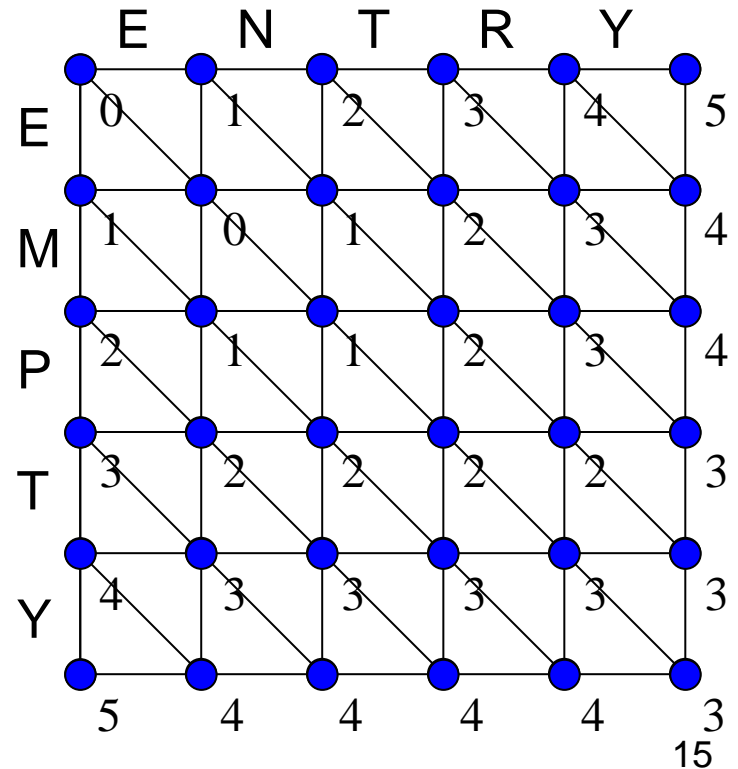
近似文字列照合の高速化(1)

あるノードの値は, 上, 左, 左上のノードの値に依存する

→ コンピュータの処理能力によって,
一度に計算できるノード数が
単純には増えない

単純な並列化

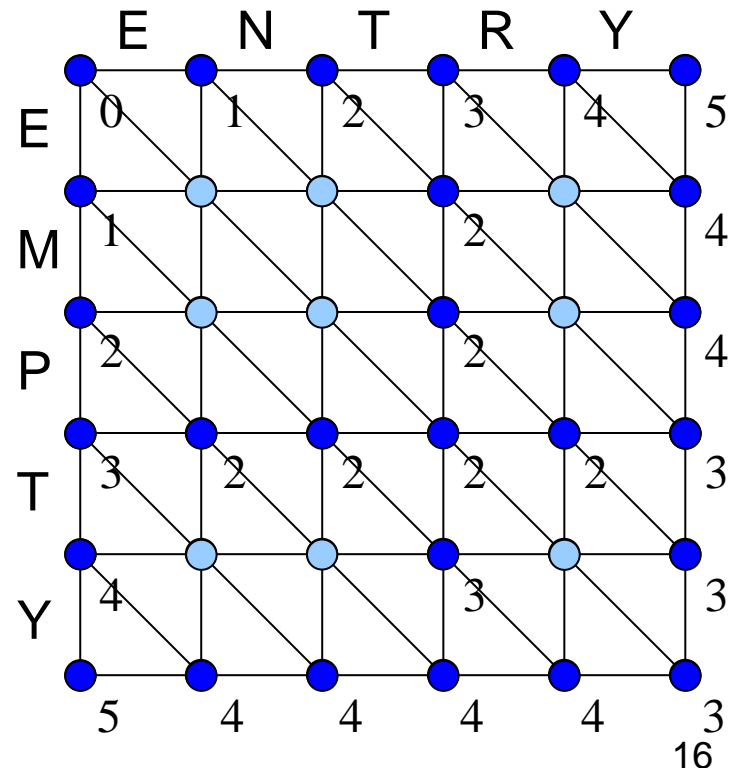
- 依存関係の無い斜めのライン上のノードの値を同時に計算
- 複数の演算器による実装としては一般的な手法
- 計算時間は $O(m+n)$
- ソフトウェア的な実装も可能



近似文字列照合の高速化(2)

事前の計算による高速化

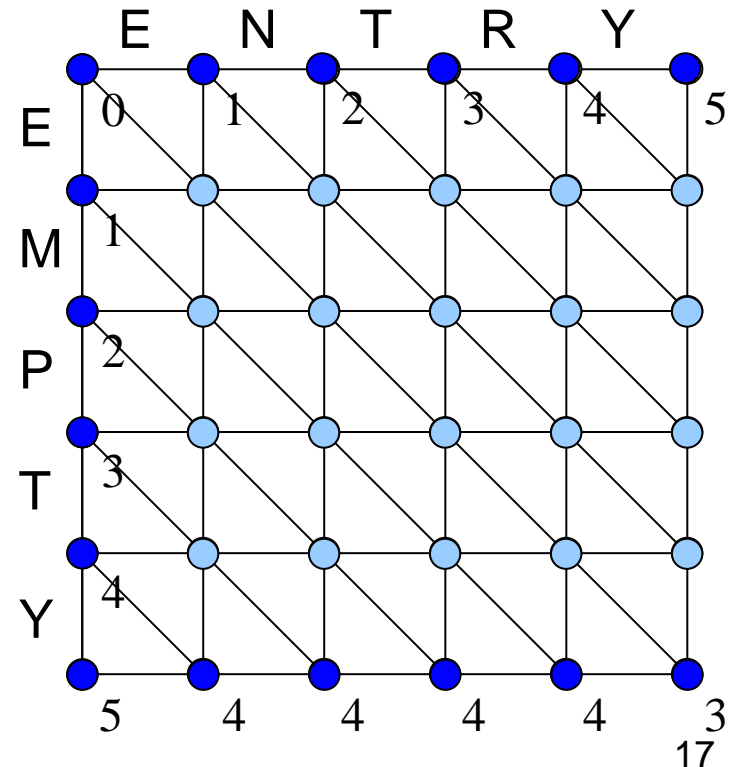
- 長さ t の2つの文字列と、行列の最初の行と列の値の組み合わせについて、最後の行と列の値のテーブルを事前に作成
- テーブルから値を $O(t)$ 時間で参照
- 計算時間は $O(mn/t)$
- テーブルのための領域が必要
- 値が計算されないノードがあるため、文字の正確な対応がわからない



近似文字列照合の高速化(3)

ビット・パラレル法

- 各行の値をビット列で表現し, 値の関係を二値の論理式で表現
- ビット演算の繰り上がりによって, 縦のノードの依存関係を表現
- 一度に w 文字の処理ができるとき, 計算時間は $O(n/w)$
- 最後の行のみ値を計算するため, 文字の正確な対応がわからない
- 重み付きの距離の計算は困難



我々のアプローチ

メモリの参照よりも, なるべく演算を行うアルゴリズムの方が良さそうだ...

- ビット・パラレル法の適用可能性の検討
 - 単純な並列化との併用
 - 文字の正確な対応への拡張
 - 重み付き距離への一般化
- ハードウェアの性能をより反映する計算への置き換え
 - 長いテキストからパターンに似ている部分を探す場合
 - 少し簡単な問題を解けば, 問題の一部を解いたことになる?
 - アプリケーションによっては効果がありそうだ!

不一致を許した文字列照合

近似文字列照合に比べて

- 根本的に計算量が少ない別の問題(挿入・削除が無い)
- テキストの分割による並列化が可能(結果の結合が容易)
- 文字の種類についての分割による並列化が可能
- 単純なやり方
 - 文字列照合と同様(加えて, 一致の数をカウント)
 - 計算時間: mn 回の文字の比較(及び, 足し算)
- うまいやり方(FFTの利用)
 - 文字を数値に置き換えて, 文字の比較を掛け算で表す
 - 計算時間: $n \log m$ 時間のFFT
 - ハードウェアによるFFTの高速化が期待できる

FFTの利用

関数 f と g の畳み込み

$$h(x) = \sum_{i=1}^n f(i) \circ g(x-i+1) \quad x=1, \dots, n$$

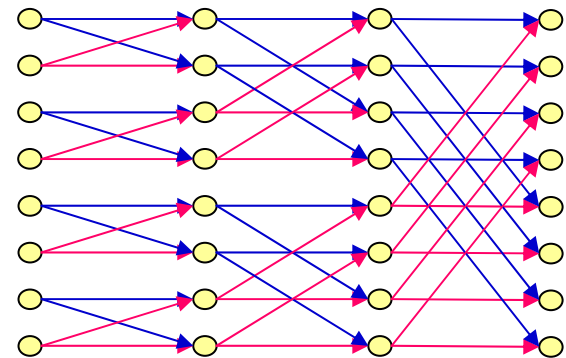
は FFT によって $O(n \log n)$ で計算できる.

first entry into ...

empty

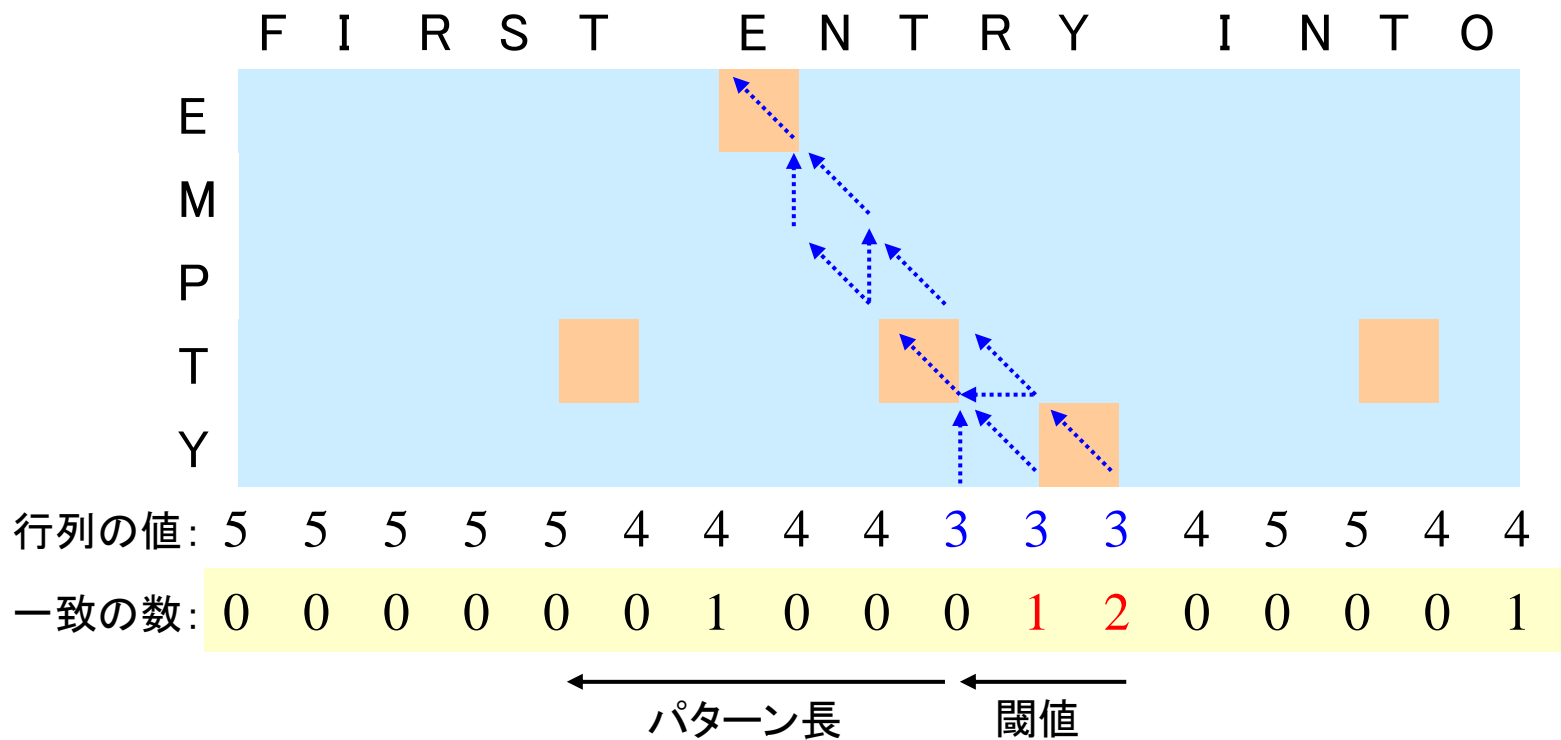
empty

→



掛け算と足し算の繰り返しの
順番をうまく変えてやると...

一致の数による前処理の例



探索の候補を，一致の数がある程度以上の位置から，高々パターン長と閾値分の長さに削減できる！

計算時間の見積もり

$m=64, n=10,000$

- 近似文字列照合
 - 単純な方法 ($O(mn)$) : 30s
 - ビットパラレル法 ($O(mn/w)$) : 1s
- 不一致を許した文字列照合
 - 単純な方法 ($O(mn)$) : 10s
 - FFTによる方法 ($O(n \log m)$) : ? (少しうまい法 ($O(mn \log m/w)$) : 1s)

見積もり

- ビットパラレル法: 演算器の性能に応じた高速化
- FFTによる前処理: 1桁少ない程度 (10倍の前処理ができる)

まとめ

- 近似文字列照合高速化の意義
 - 速さ, 感度ともに向上が求められるアプリケーションが存在
- 高速化の既存手法
 - ひとつの演算器について並列化するのが困難だ
 - 特別な場合については解決法がある
- 我々のアプローチ
 - ひとつの演算器についての並列化手法の一般化
 - 高速化が容易な計算による置き換えの実用性の考察