A Fast Algorithm for Permutation Pattern Matching Based on Alternating Runs

Marie-Louise Bruner and Martin Lackner

Vienna University of Technology

July 6, 2012
SWAT 2012, Helsinki

Permutation patterns

A permutation T (the text) contains P as a pattern if we can find a subsequence of T that is *order-isomorphic* to P.

Permutation patterns

A permutation T (the text) contains P as a pattern if we can find a subsequence of T that is *order-isomorphic* to P.

Does 53142 contain 231 as a pattern?

Permutation patterns

A permutation T (the text) contains P as a pattern if we can find a subsequence of T that is *order-isomorphic* to P.

Does 53142 contain 231 as a pattern?

✓ (342 is a matching)

Permutation patterns

A permutation T (the text) contains P as a pattern if we can find a subsequence of T that is *order-isomorphic* to P.

Does 53142 contain 231 as a pattern?

✓ (342 is a matching)

Does 53142 contain 123 as a pattern?

Permutation patterns

A permutation T (the text) contains P as a pattern if we can find a subsequence of T that is *order-isomorphic* to P.

Does 53142 contain 231 as a pattern?

✓ (342 is a matching)

Does 53142 contain 123 as a pattern?

✗

Permutation patterns

A permutation T (the text) contains P as a pattern if we can find a subsequence of T that is *order-isomorphic* to P.

Does 53142 contain 231 as a pattern?

✓ (342 is a matching)

Does 53142 contain 123 as a pattern?

✗

Does 53142 contain 4231 as a pattern?

Permutation patterns

A permutation T (the text) contains P as a pattern if we can find a subsequence of T that is *order-isomorphic* to P.

Does 53142 contain 231 as a pattern?

✓ (342 is a matching)

Does 53142 contain 123 as a pattern?

✗

Does 53142 contain 4231 as a pattern?

✓ (5342 is a matching)

Enumerative combinatorics

Theorem

The n-permutations that do not contain the pattern 123 are counted by the n-th Catalan number.

Enumerative combinatorics

Theorem

The n-permutations that do not contain the pattern 123 are counted by the n-th Catalan number.
The same holds for every other pattern of length 3.

Enumerative combinatorics

Theorem

The n-permutations that do not contain the pattern 123 are counted by the n-th Catalan number.
The same holds for every other pattern of length 3.

Stanley-Wilf conjecture, shown by Marcus and Tardos (2004)

For every permutation P there is a constant c such that the number of n-permutations that *do not contain* P as a pattern is bounded by c^n.

Permutation Pattern Matching

Permutation Pattern Matching (PPM)

Instance:	A permutation T of length n (the text) and a permutation P of length $k \leq n$ (the pattern).
Question:	Is there a matching of P into T?

Permutation Pattern Matching

PERMUTATION PATTERN MATCHING (PPM)

Instance:	A permutation T of length n (the text) and a permutation P of length $k \leq n$ (the pattern).
Question:	Is there a matching of P into T?

1993 (Bose, Buss, Lubiw): PPM is in general NP-complete.

Tractable cases of PPM

- Pattern avoids both 3142 and 2413 $\mathcal{O}(kn^4)$

Tractable cases of PPM

- Pattern avoids both 3142 and 2413 $\mathcal{O}(kn^4)$
- $P = 12 \ldots k$ or $P = k \ldots 21$ $\mathcal{O}(n \log \log n)$

Tractable cases of PPM

- Pattern avoids both 3142 and 2413 — $\mathcal{O}(kn^4)$
- $P = 12 \dots k$ or $P = k \dots 21$ — $\mathcal{O}(n \log \log n)$
- P has length at most 4 — $\mathcal{O}(n \log n)$

Tractable cases of PPM

- Pattern avoids both 3142 and 2413 $\mathcal{O}(kn^4)$
- $P = 12\ldots k$ or $P = k\ldots 21$ $\mathcal{O}(n \log\log n)$
- P has length at most 4 $\mathcal{O}(n \log n)$
- Pattern and Text avoid 321 $\mathcal{O}(k^2 n^6)$

The general case

Anything better than the

$\mathcal{O}^*(2^n)$

runtime of brute-force search?

Parameterized Complexity Theory

Idea: confine the combinatorial explosion to a *parameter* of the input

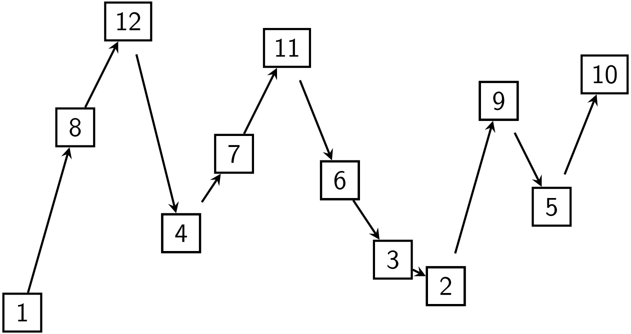
Parameterized Complexity Theory

Idea: confine the combinatorial explosion to a *parameter* of the input

Fixed-parameter tractability

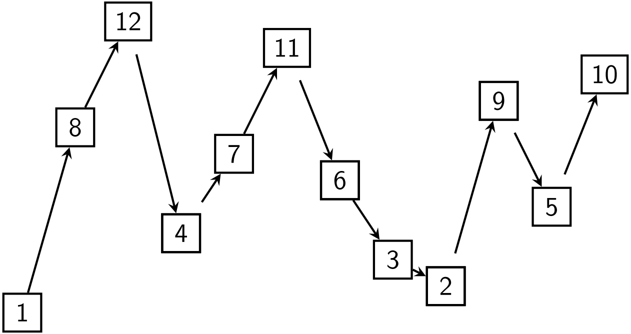
A problem is *fixed-parameter tractable with respect to a parameter k* if there is a computable function f and an integer c such that there is an algorithm solving the problem in time $\mathcal{O}(f(k) \cdot |I|^c)$.

Alternating runs



1 8 12 (up), 4 (down), 7 11 (up), 6 3 2 (down), 9 (up), 5 (down), 10 (up)

Alternating runs



1 8 12 (up), 4 (down), 7 11 (up), 6 3 2 (down), 9 (up), 5 (down), 10 (up)

Notation

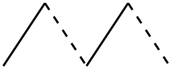
run(π)...the number of alternating runs in π,

The alternating run algorithm

- Matching functions:
  Reduce the search space
- Dynamic programming algorithm:
  Checks for every matching function whether there is a compatible matching

Matching functions

Pattern P

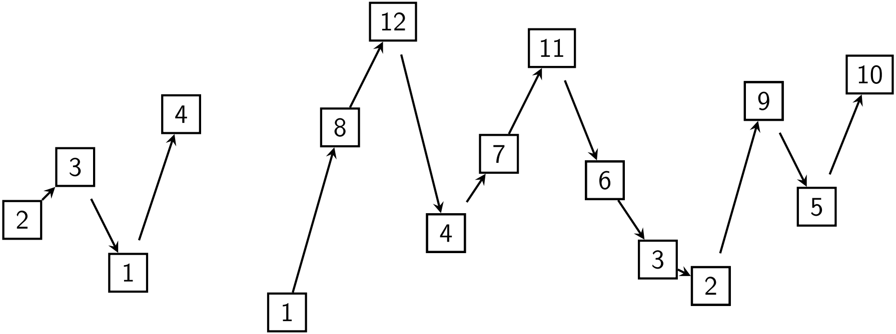


$\downarrow$ matching function $\downarrow$

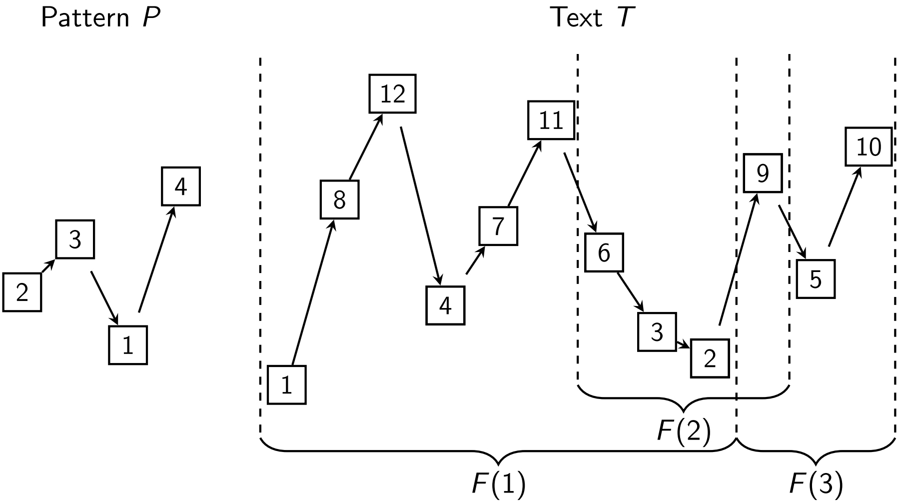
Text T

Matching functions - an example

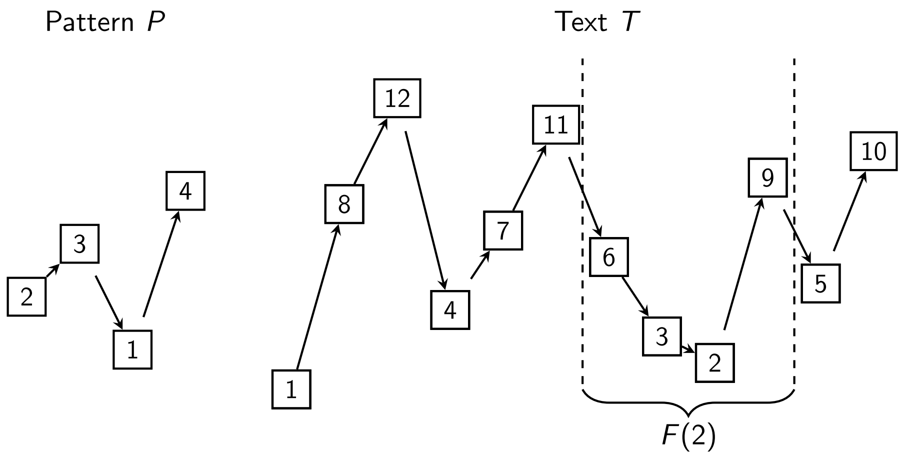
Pattern P Text T



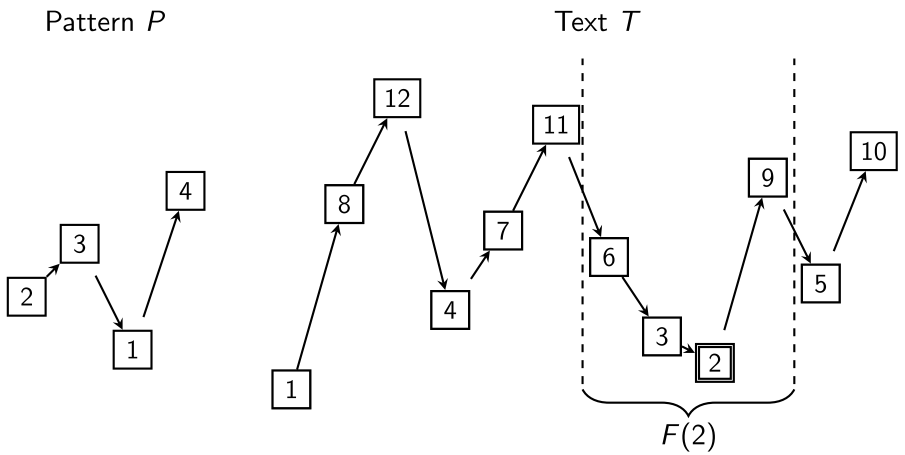
Matching functions - an example



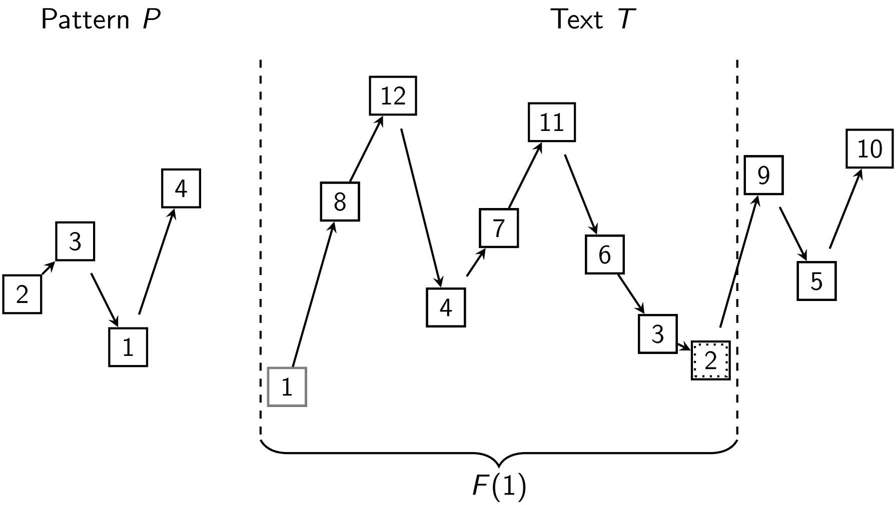
The algorithm - finding a matching



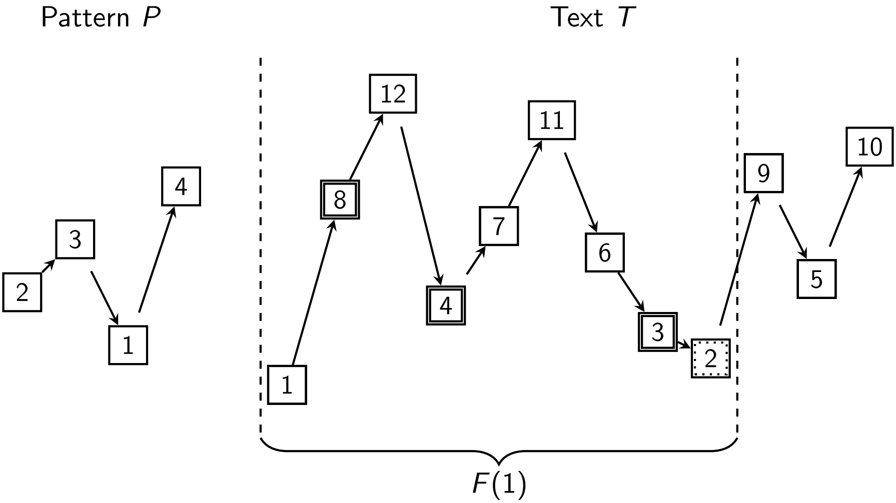
The algorithm - finding a matching



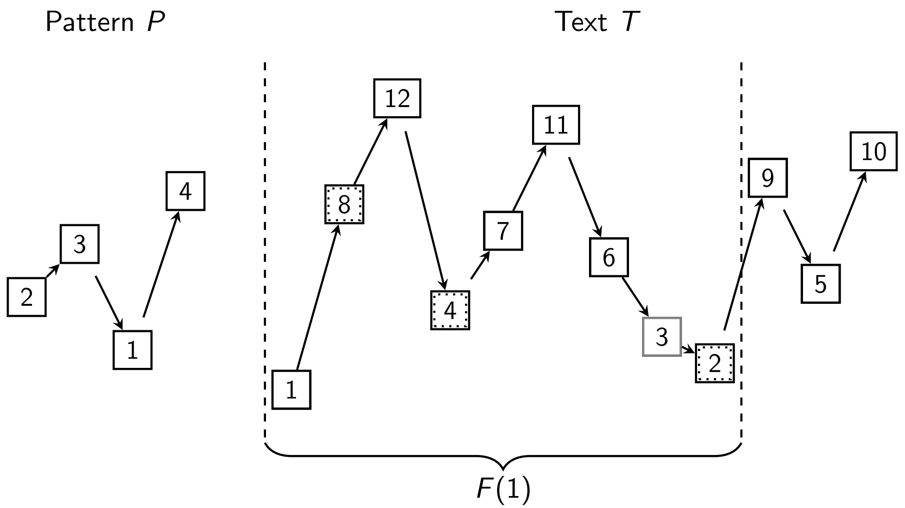
The algorithm - finding a matching



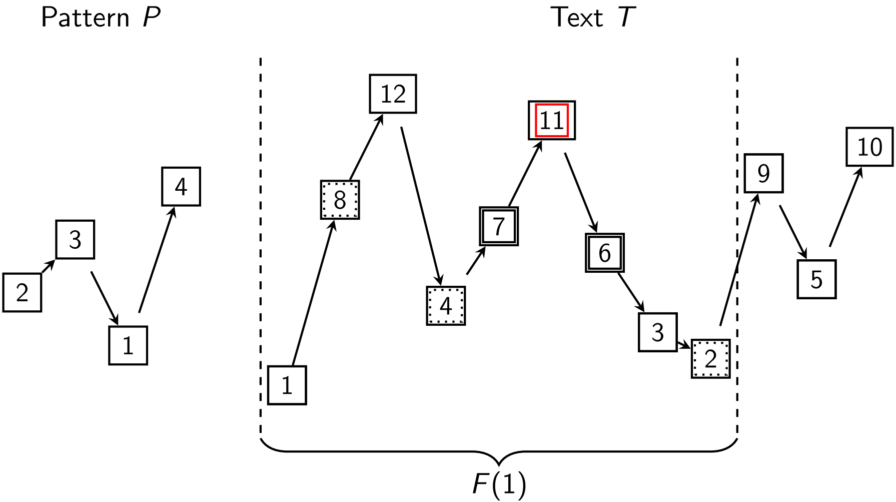
The algorithm - finding a matching



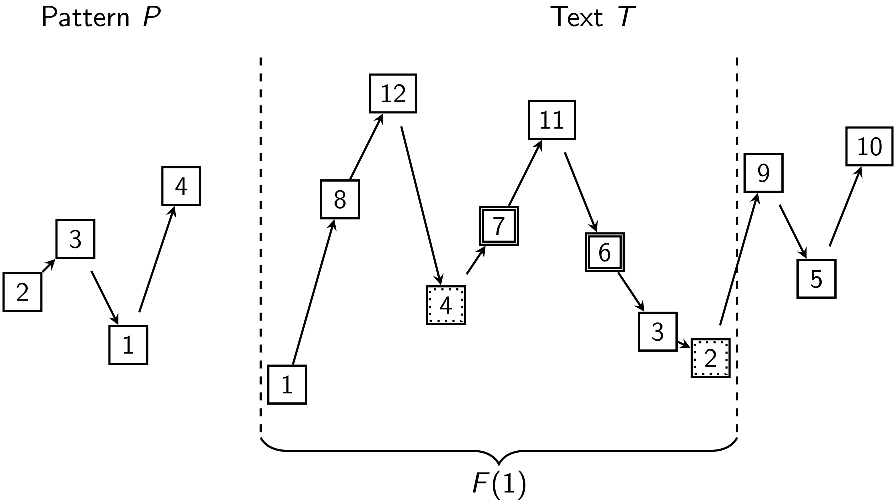
The algorithm - finding a matching



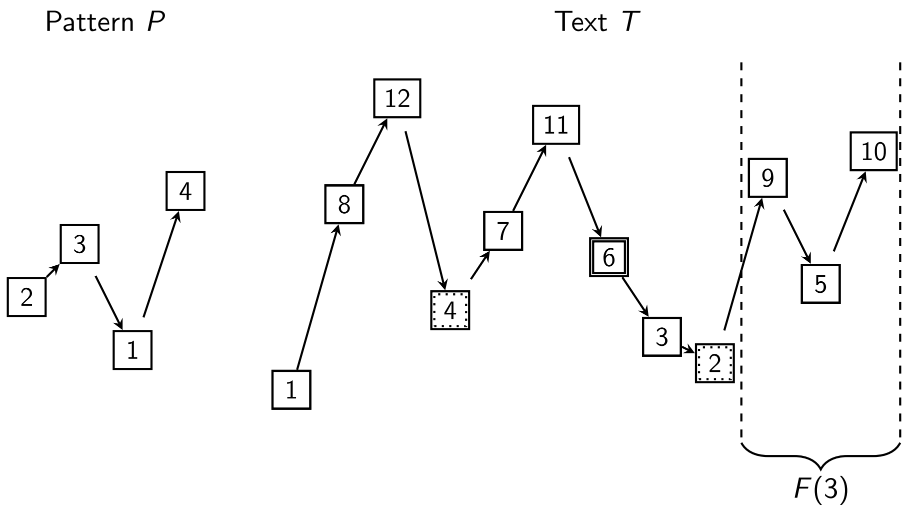
The algorithm - finding a matching



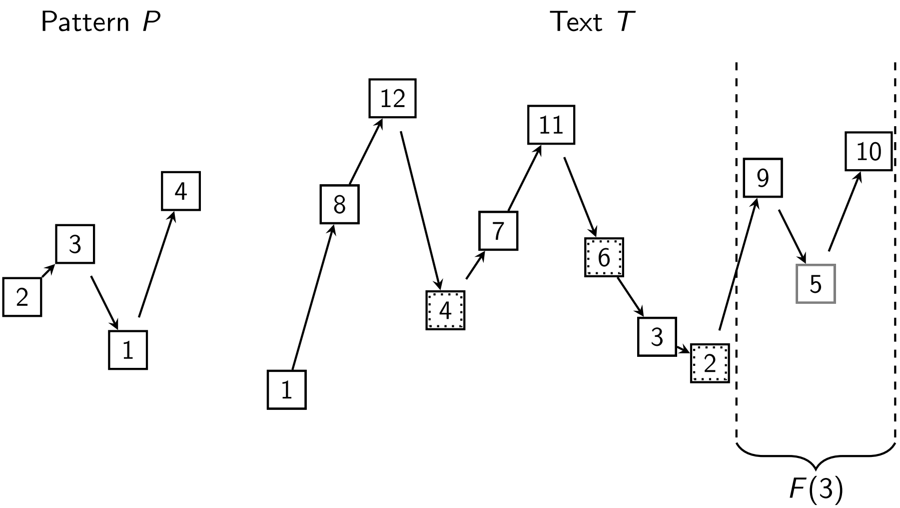
The algorithm - finding a matching



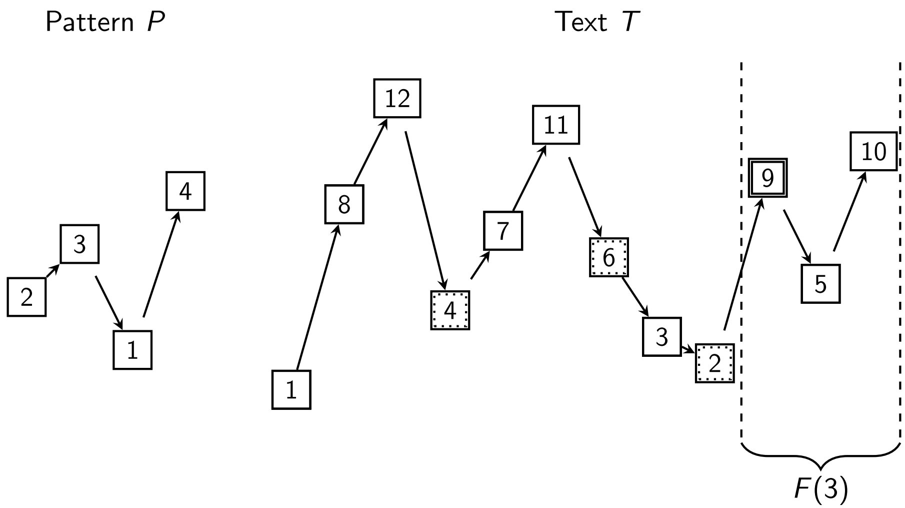
The algorithm - finding a matching



The algorithm - finding a matching



The algorithm - finding a matching



Runtime

Matching functions: $\sqrt{2}^{\mathrm{run}(T)}$

Dynamic programming algorithm: $\mathcal{O}^*(1.2611^{\mathrm{run}(T)})$

Runtime

Matching functions: $\sqrt{2}^{\mathrm{run}(T)}$

Dynamic programming algorithm: $\mathcal{O}^*(1.2611^{\mathrm{run}(T)})$

In total: $\mathcal{O}^*(1.784^{\mathrm{run}(T)})$

Runtime

Matching functions: $\sqrt{2}^{\mathrm{run}(T)}$

Dynamic programming algorithm: $\mathcal{O}^*(1.2611^{\mathrm{run}(T)})$

In total: $\mathcal{O}^*(1.784^{\mathrm{run}(T)})$

$\rightarrow$ This is a fixed-parameter tractable (FPT) algorithm,
i.e. a runtime of $f(k) \cdot n^c$.

Runtime

Matching functions: $\sqrt{2}^{\mathrm{run}(T)}$

Dynamic programming algorithm: $\mathcal{O}^*(1.2611^{\mathrm{run}(T)})$

In total: $\mathcal{O}^*(1.784^{\mathrm{run}(T)})$

$\rightarrow$ This is a fixed-parameter tractable (FPT) algorithm,
i.e. a runtime of $f(k) \cdot n^c$.

Since $\mathrm{run}(T) \leq n$, we also obtain $\mathcal{O}^*(1.784^n)$

Alternating runs in the pattern run(P)

$\mathcal{O}^*(1.784^{\mathrm{run}(T)})$ FPT, i.e. $f(k) \cdot n^c$

$\mathcal{O}^*\left(\left(\frac{n^2}{2\mathrm{run}(P)}\right)^{\mathrm{run}(P)}\right)$ XP, i.e. $n^{f(k)}$

Alternating runs in the pattern run(P)

$\mathcal{O}^*(1.784^{\mathrm{run}(T)})$ FPT, i.e. $f(k) \cdot n^c$

$\mathcal{O}^*\left(\left(\frac{n^2}{2\mathrm{run}(P)}\right)^{\mathrm{run}(P)}\right)$ XP, i.e. $n^{f(k)}$

no FPT result possible (W[1]-hardness)

Conclusion

Main results

- $\mathcal{O}^*(1.784^{\mathrm{run}(T)}) \to$ FPT result
- $\mathcal{O}^*(1.784^n) \to$ fastest general algorithm
- W[1]-hardness for run(P)

Conclusion

Main results

- $\mathcal{O}^*(1.784^{\mathsf{run}(T)}) \rightarrow$ FPT result
- $\mathcal{O}^*(1.784^n) \rightarrow$ fastest general algorithm
- W[1]-hardness for $\mathsf{run}(P)$

Future work

- PPM parameterized by some other parameter of P? By $k = |P|$?

Conclusion

Main results

- $\mathcal{O}^*(1.784^{\mathsf{run}(T)}) \rightarrow$ FPT result
- $\mathcal{O}^*(1.784^n) \rightarrow$ fastest general algorithm
- W[1]-hardness for run(P)

Future work

- PPM parameterized by some other parameter of P? By $k = |P|$?
- Kernelization results

Conclusion

Main results

- $\mathcal{O}^*(1.784^{\mathsf{run}(T)}) \rightarrow$ FPT result
- $\mathcal{O}^*(1.784^n) \rightarrow$ fastest general algorithm
- W[1]-hardness for run(P)

Future work

- PPM parameterized by some other parameter of P? By $k = |P|$?
- Kernelization results
- Other permutation statistics of the text?