On Strings Having the Same Length-k Substrings

Dutch Optimization Seminar - PhD Talk

Giulia Bernardini[1], Alessio Conte[2], **Estéban Gabory**[1], Roberto Grossi[2], Grigorios Loukides[3], Solon P. Pissis[1,4], Giulia Punzi[2] and Michelle Sweering[1]

[1]CWI Amsterdam, [2]Università di Pisa,
[3]King's College London, [4]Vrije Universiteit Amsterdam

March 31st, 2022

Shortest equivalent string : problem definition

SHORTEST $\mathcal{S}$-EQUIVALENT STRING
Input: A set $\mathcal{S}$ of n length-k strings
Output: A shortest string T such that the set of k substrings of T is $\mathcal{S}$, or FAIL if that is not possible.

Shortest equivalent string : problem definition

SHORTEST $\mathcal{S}$-EQUIVALENT STRING
Input: A set $\mathcal{S}$ of n length-k strings
Output: A shortest string T such that the set of k substrings of T is $\mathcal{S}$, or FAIL if that is not possible.

If $k = 3$, $\mathcal{S} = \{$"abr","bra","rac","aca","cad","ada","dab"$\}$

Shortest equivalent string : problem definition

SHORTEST $\mathcal{S}$-EQUIVALENT STRING
Input: A set $\mathcal{S}$ of n length-k strings
Output: A shortest string T such that the set of k substrings of T is $\mathcal{S}$, or FAIL if that is not possible.

If $k = 3$, $\mathcal{S} = \{$"abr","bra","rac","aca","cad","ada","dab"$\}$
The string "abracadabra" has $\mathcal{S}$ for set of 3-substrings

Shortest equivalent string : problem definition

SHORTEST $\mathcal{S}$-EQUIVALENT STRING
Input: A set $\mathcal{S}$ of n length-k strings
Output: A shortest string T such that the set of k substrings of T is $\mathcal{S}$, or FAIL if that is not possible.

If $k = 3$, $\mathcal{S} = \{$"abr","bra","rac","aca","cad","ada","dab"$\}$
The string "abracadabra" has $\mathcal{S}$ for set of 3-substrings
The strings "abracadab" or "cadabraca" (for example) are shortest for this property

Motivations

- Data privacy

Motivations

- Data privacy
  - Private string : "abracadabra".

Motivations

- Data privacy
  - Private string : "abracadabra".
  - Public string for pattern matching queries : "cadabraca"

Motivations

- Data privacy
  - Private string : "abracadabra".
  - Public string for pattern matching queries : "cadabraca"
  - One can answer pattern matching queries for patterns shorter than 3

Motivations

- Data privacy
  - Private string : "abracadabra".
  - Public string for pattern matching queries : "cadabraca"
  - One can answer pattern matching queries for patterns shorter than 3
- Data compression

Motivations

- Data privacy
  - Private string : "abracadabra".
  - Public string for pattern matching queries : "cadabraca"
  - One can answer pattern matching queries for patterns shorter than 3
- Data compression
- Bioinformatics

De Bruijn graph of a string

Definition (De Bruijn graph of order k of a string T)

- Nodes are the length $(k-1)$ substrings of T

De Bruijn graph of a string

Definition (De Bruijn graph of order k of a string T)

- Nodes are the length $(k-1)$ substrings of T
- Each time a k substring u is in T, we add a directed edge from $u[0..k-2]$ to $u[1..k-1]$.

De Bruijn graph of a string

Definition (De Bruijn graph of order k of a string T)

- Nodes are the length $(k-1)$ substrings of T
- Each time a k substring u is in T, we add a directed edge from $u[0..k-2]$ to $u[1..k-1]$.

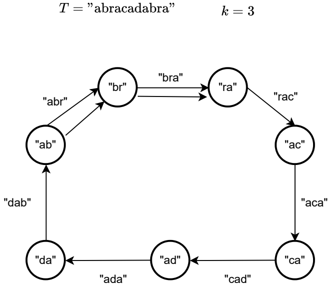


Figure: De Bruijn graph of order 3 for $T =$ "abracadabra"

De Bruijn graph of a string

Definition (De Bruijn graph of order k of a string T)

- Nodes are the length $(k-1)$ substrings of T
- Each time a k substring u is in T, we add a directed edge from $u[0..k-2]$ to $u[1..k-1]$.

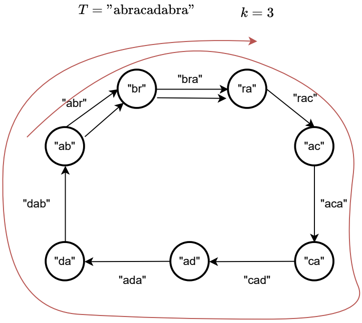


Figure: De Bruijn graph of order 3 for $T =$ "abracadabra"

De Bruijn graph of a string

Definition

A graph G is *semi-Eulerian* if it can be traversed by visiting each edge once and only once

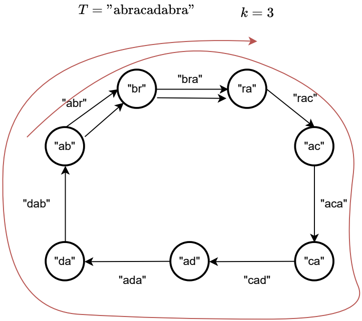


Figure: De Bruijn graph of order 3 for $T =$ "abracadabra"

De Bruijn graph of a string

Definition

A graph G is *semi-Eulerian* if it can be traversed by visiting each edge once and only once

Proposition

If a graph G is a de Bruijn graph of a string, then G is semi-Eulerian

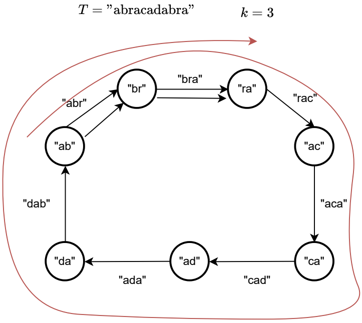


Figure: De Bruijn graph of order 3 for $T =$ "abracadabra"

Eulerian trails

Definition

An *Eulerian trail* on a graph G is a walk on G that traverses every edges *exactly* once

De Bruijn graph of a set of k strings

$\mathcal{S}_k = \{\text{"abr"}, \text{"aca"}, \text{"dab"}, \text{"abc"}\}$ $\qquad k = 3$

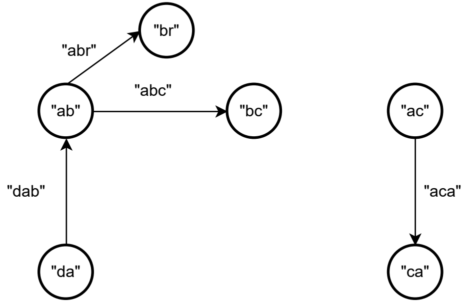


Figure: De Bruijn graph of order 3 for $\mathcal{S} = \{\text{"abr"}, \text{"aca"}, \text{"dab}, \text{"abc"}\}$

De Bruijn graph of a set of k strings

$$k = 3$$

$$\mathcal{S}_k = \{\text{"abr"}, \text{"bra"}, \text{"rac"}, \text{"aca"}, \text{"cad"}, \text{"ada"}, \text{"dab"}\}$$

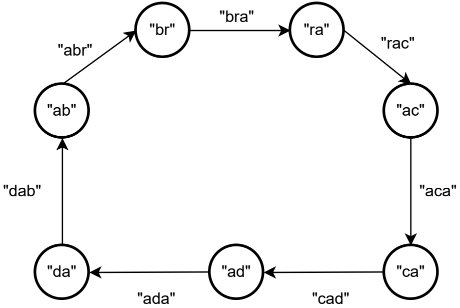


Figure: De Bruijn graph of order 3 for
$\mathcal{S} = \{\text{"abr"}, \text{"bra"}, \text{"rac"}, \text{"aca"}, \text{"cad"}, \text{"ada"}, \text{"dab}\}$

De Bruijn graph of a set of k strings

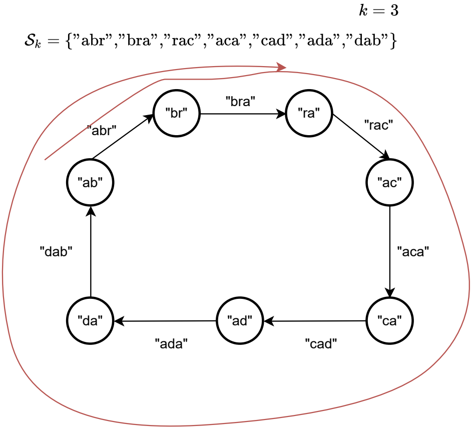


Figure: De Bruijn graph of order 3 for
$\mathcal{S} = \{$"abr","bra","rac","aca","cad","ada","dab"$\}$

Eulerian walks

An *Eulerian walk* on a graph G is a walk on G that traverses every edges *at least* once

Solving shortest equivalent k-string via Eulerian walks

Proposition

The strings having $\mathcal{S}$ as a set of k-substrings corresponds to walks traversing *at least* once each edge of the order k de Bruijn graph of $\mathcal{S}$.

Solving shortest equivalent k-string via Eulerian walks

Proposition

The strings having $\mathcal{S}$ as a set of k-substrings corresponds to walks traversing *at least* once each edge of the order k de Bruijn graph of $\mathcal{S}$.

- If such a walk exists, then one can make the graph semi-Eulerian by copying the edges traversed several times

Solving shortest equivalent k-string via Eulerian walks

Proposition

The strings having $\mathcal{S}$ as a set of k-substrings corresponds to walks traversing *at least* once each edge of the order k de Bruijn graph of $\mathcal{S}$.

- If such a walk exists, then one can make the graph semi-Eulerian by copying the edges traversed several times
- We solve Shortest $\mathcal{S}$-Equivalent String by :

Solving shortest equivalent k-string via Eulerian walks

Proposition

The strings having $\mathcal{S}$ as a set of k-substrings corresponds to walks traversing *at least* once each edge of the order k de Bruijn graph of $\mathcal{S}$.

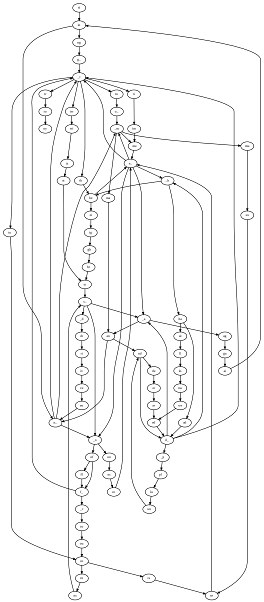
- If such a walk exists, then one can make the graph semi-Eulerian by copying the edges traversed several times
- We solve SHORTEST $\mathcal{S}$-EQUIVALENT STRING by :
  - Finding a minimal set of edge that can be copied to make the de Bruijn graph semi-Eulerian : we call *Eulerian extension* such a graph with copied edges

Solving shortest equivalent k-string via Eulerian walks

Proposition

The strings having $\mathcal{S}$ as a set of k-substrings corresponds to walks traversing *at least* once each edge of the order k de Bruijn graph of $\mathcal{S}$.

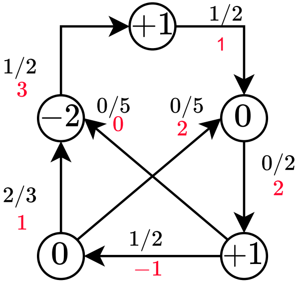
- If such a walk exists, then one can make the graph semi-Eulerian by copying the edges traversed several times
- We solve SHORTEST $\mathcal{S}$-EQUIVALENT STRING by :
  - Finding a minimal set of edge that can be copied to make the de Bruijn graph semi-Eulerian : we call *Eulerian extension* such a graph with copied edges
  - Find Eulerian trails in the graph



$T =$"sing_to_me_of_the_man_muse
_the_man_of_twists_and_turns
_driven_time_and_again_off_course
_once_he_had_plundered
_the_hallowed_heights_of_troy"
$k = 3$

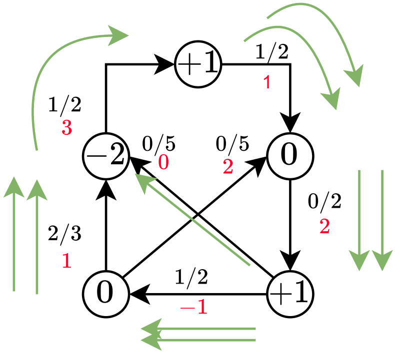
Flows : definition

To find Eulerian extensions, we use flows :



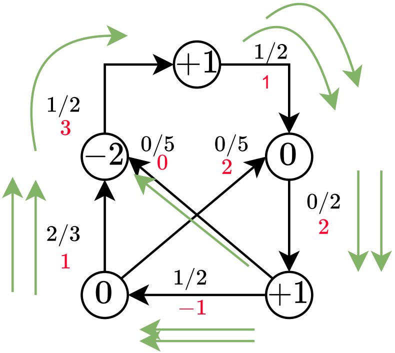
Flows : definition

To find Eulerian extensions, we use flows :



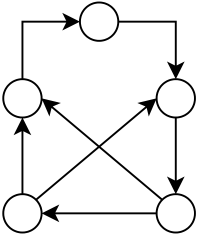
Flows : definition

To find Eulerian extensions, we use flows :



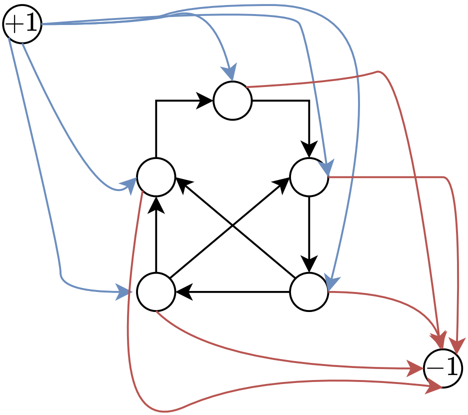
$$c(f) = 3 \cdot 1 + 1 \cdot 2 + 2 \cdot 2 - 1 \cdot 2 + 1 \cdot 2 + 0 \cdot 1 = 12$$

Construction of a flow problem



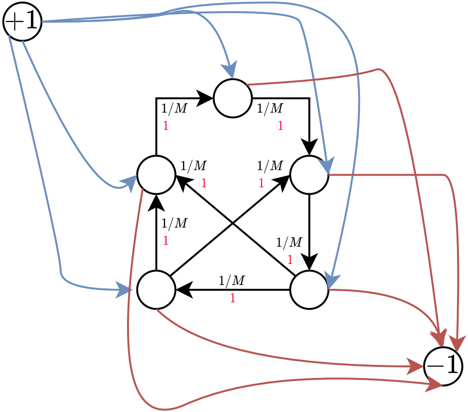
Now we want to model our extension problem with a flow problem.

Construction of a flow problem



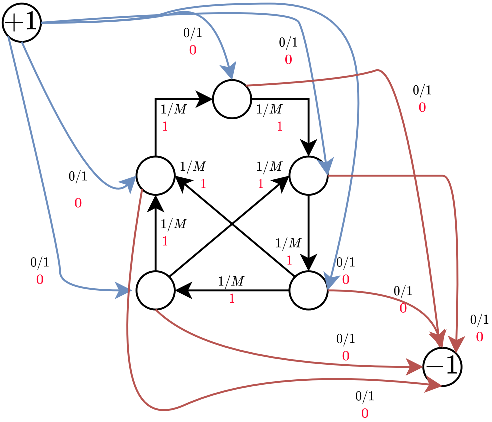
Now we want to model our extension problem with a flow problem.
For this we create bogus nodes and we connect then so the flow can "start" and "end" anywhere on the graph

Construction of a flow problem



Now we want to model our extension problem with a flow problem.
We give minimal capacity 1 to each original edge so the flow has to visit each edge at least once

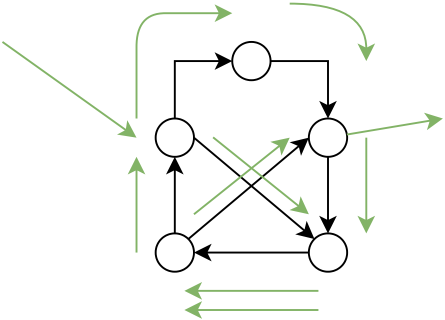
Construction of a flow problem



Now we want to model our extension problem with a flow problem.
The maximal capacity is a very large integer on each edge, and the costs are 1 on edges from the original graph, 0 on bogus edges

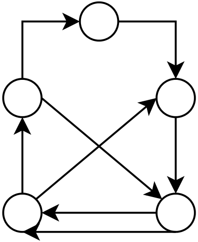
Link between flows on G_{ext} and Eulerian extensions

- We can obtain a minimum cost flow in $\tilde{\mathcal{O}}(|E|^2 + |E||V|)$ ([Orlin, 1993])



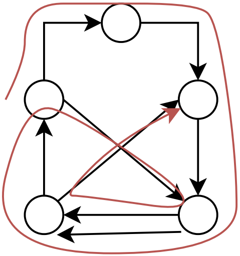
Link between flows on G_{ext} and Eulerian extensions

- We can obtain a minimum cost flow in $\tilde{\mathcal{O}}(|E|^2 + |E||V|)$ ([Orlin, 1993])
- A flow gives us a semi-Eulerian extension of G obtained by copying the edges as many time as they are traversed



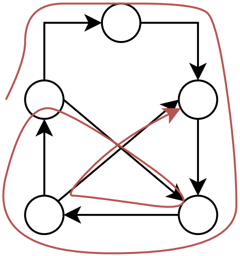
Link between flows on G_{ext} and Eulerian extensions

- We can obtain a minimum cost flow in $\tilde{\mathcal{O}}(|E|^2 + |E||V|)$ ([Orlin, 1993])
- A flow gives us a semi-Eulerian extension of G obtained by copying the edges as many time as they are traversed
- Find an Eulerian trail $||\mathcal{W}||$ on the extended graph (in $\mathcal{O}(||\mathcal{W}||)$)



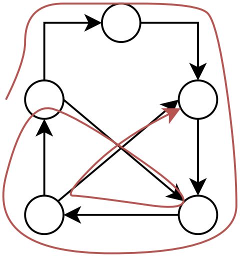
Link between flows on G_{ext} and Eulerian extensions

- We can obtain a minimum cost flow in $\tilde{\mathcal{O}}(|E|^2 + |E||V|)$ ([Orlin, 1993])
- A flow gives us a semi-Eulerian extension of G obtained by copying the edges as many time as they are traversed
- Find an Eulerian trail $||\mathcal{W}||$ on the extended graph (in $\mathcal{O}(||\mathcal{W}||)$)
- This Eulerian trail corresponds to an Eulerian walk on G



Link between flows on G_{ext} and Eulerian extensions

- We can obtain a minimum cost flow in $\tilde{\mathcal{O}}(|E|^2 + |E||V|)$ ([Orlin, 1993])
- A flow gives us a semi-Eulerian extension of G obtained by copying the edges as many time as they are traversed
- Find an Eulerian trail $||\mathcal{W}||$ on the extended graph (in $\mathcal{O}(||\mathcal{W}||)$)
- This Eulerian trail corresponds to an Eulerian walk on G
- A minimal cost flow gives us a shortest walk



Results

Theorem

The SHORTEST $\mathcal{S}$-EQUIVALENT STRING *problem can be solved in* $\tilde{\mathcal{O}}(nk + n^2)$ *time.*

Reduction : The Directed Chinese Postman

DIRECTED CHINESE POSTMAN (DCP)
Input: A directed graph $G(V, E)$.
Output: A shortest closed Eulerian walk, or FAIL if that is not possible.

Reduction : The Directed Chinese Postman

DIRECTED CHINESE POSTMAN (DCP)
Input: A directed graph $G(V, E)$.
Output: A shortest closed Eulerian walk, or FAIL if that is not possible.

Theorem

Any instance of the Directed Chinese Postman problem can be reduced to an instance of the SHORTEST $\mathcal{S}$-EQUIVALENT STRING *problem in linear time*

Reduction : The Directed Chinese Postman

Directed Chinese Postman (DCP)
Input: A directed graph $G(V, E)$.
Output: A shortest closed Eulerian walk, or FAIL if that is not possible.

Theorem

Any instance of the Directed Chinese Postman problem can be reduced to an instance of the Shortest $\mathcal{S}$-Equivalent String *problem in linear time*

Theorem

The Directed Chinese Postman problem can be solved in $\tilde{O}(|E|^2)$ time

Looking for z strings

- z-Shortest $\mathcal{S}$-Equivalent String: One wants the z shortest string instead of only the shortest

Looking for z strings

- z-Shortest $\mathcal{S}$-Equivalent String: One wants the z shortest string instead of only the shortest
- Finding z minimal cost flows can be done in $\tilde{\mathcal{O}}(z|V|^3)$ time, or in $\tilde{\mathcal{O}}(z(|E||V| + |V|^2))$ time. ([Könen et al., 2021])

Looking for z strings

- z-Shortest $\mathcal{S}$-Equivalent String: One wants the z shortest string instead of only the shortest
- Finding z minimal cost flows can be done in $\tilde{\mathcal{O}}(z|V|^3)$ time, or in $\tilde{\mathcal{O}}(z(|E||V| + |V|^2))$ time. ([Könen et al., 2021])
- In a semi Eulerian graph, one can compute the set $\mathcal{W}_z$ of the z smallest Eulerian walks in linear time ([Kurita and Wasa, 2021] or [Conte et al., 2021])
- As before, each flow corresponds at least to an Eulerian walk and the z minimal cost flows correspond to at least z Eulerian walks

Looking for z strings

- z-SHORTEST $\mathcal{S}$-EQUIVALENT STRING: One wants the z shortest string instead of only the shortest
- Finding z minimal cost flows can be done in $\tilde{\mathcal{O}}(z|V|^3)$ time, or in $\tilde{\mathcal{O}}(z(|E||V|+|V|^2))$ time. ([Könen et al., 2021])
- In a semi Eulerian graph, one can compute the set $\mathcal{W}_z$ of the z smallest Eulerian walks in linear time ([Kurita and Wasa, 2021] or [Conte et al., 2021])
- As before, each flow corresponds at least to an Eulerian walk and the z minimal cost flows correspond to at least z Eulerian walks

Theorem

The z-SHORTEST $\mathcal{S}$-EQUIVALENT STRING problem can be solved in $\tilde{\mathcal{O}}(nk + zn^2 + ||\mathcal{T}_z||)$ time.

References I

Conte, A., Grossi, R., Loukides, G., Pisanti, N., Pissis, S. P., and Punzi, G. (2021).
Beyond the BEST theorem: Fast assessment of eulerian trails.
In *Fundamentals of Computation Theory - 23rd International Symposium,*, volume 12867 of *Lecture Notes in Computer Science*, pages 162–175. Springer.

Könen, D., Schmidt, D. R., and Spisla, C. (2021).
Finding all minimum cost flows and a faster algorithm for the K best flow problem.
CoRR, abs/2105.10225.

Kurita, K. and Wasa, K. (2021).
Constant amortized time enumeration of Eulerian trails.
CoRR, abs/2101.10473.

References II

Orlin, J. B. (1993).
A faster strongly polynomial minimum cost flow algorithm.
Oper. Res., 41(2):338–350.