

d.) Sequenzierung

(S. Schuitzer Skript 2.7
Vazirani Chapter 7)

1. Einführung, Grundkonzepte

Definition: Ein String ist ein endliches Wort über einem endlichen Alphabet Σ .

Wir beschäftigen uns mit dem folgenden Problem:

SHORTEST COMMON SUPERSTRING

Eingabe: Eine Menge U von Strings $|U|=n$

Ausgabe: Ein String S minimaler Länge, der alle Strings in U als Teilstring enthält.

(Minimierungsproblem)

Wir dürfen annehmen, dass kein String in U Teilstring eines anderen Strings in U ist, (weil wir solche Teilstrings aus der Instanz entfernen dürften).

Anwendung: Sequenzierung: die Reihenfolge der Basenpaare für einen Strang eines DNA-Moleküls zu bestimmen. (siehe Skript!)

Im Beispiel beschrieben im Skript: 137 Millionen Basenpaare

direkte Sequenzierung:



Die DNA-Sequenz wird sukzessive in fast-disjunkte kleine Fragmente von ~ 500 Basenpaaren zerlegt, und diese Fragmente werden iterativ (eins nach dem anderen) sequenziert. \rightarrow sehr zeitaufwändig; für lange Sequenzen nicht geeignet.

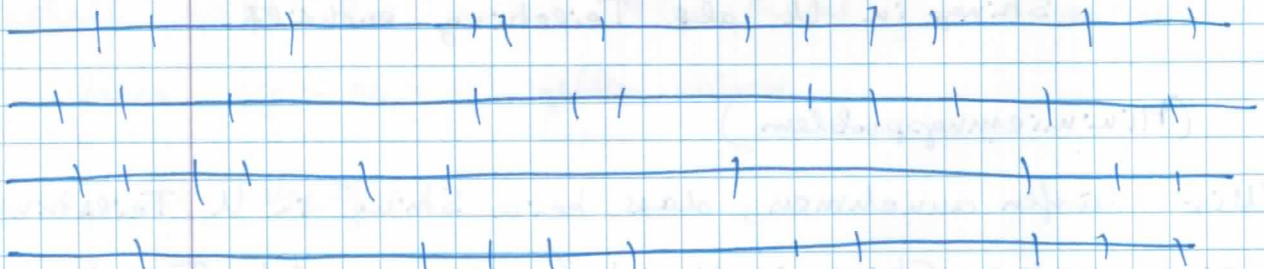
52. Wie kann man diesen iterativen Prozess parallelisieren?

Shotgun - Sequenzierung

Es wird parallel auf kleine Fragmente zerschnitten und parallel sequenziert.
(an zufälligen Positionen). Aber: die Reihenfolge der Fragmente geht so verloren.

Deshalb werden zuerst mehrere Kopien der DNA - Sequenz erstellt (14 im Beispiel) und an unabhängig zufällig gewählten Positionen zerschnitten.

Beachte: die sich überlappenden Teilfragmente die aus verschiedenen Kopien geschnitten werden, sollen Information über die gesamte Reihenfolge geben

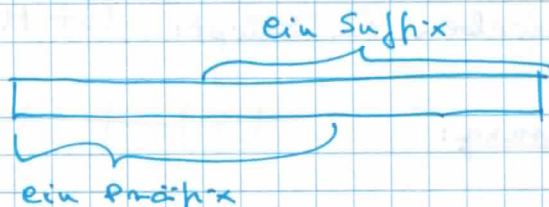


Die Eingabe - Sequenz ist ein Superstring aller dieser Fragmente.
Deshalb wird ein kürzester Superstring aller Fragmente gesucht.)

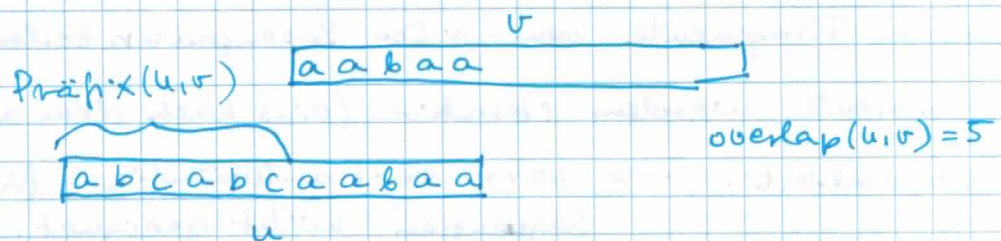
Definition: Für Strings u und v ist $\text{overlap}(u, v)$ die Länge des längsten Suffix von u der auch Präfix von v ist.

Präfix (u, v) ist der von v nicht überdeckte Präfix in u .

(Präfix $()$ ist also ein String, $\text{overlap}()$ ist eine Zahl)



Beispiel:

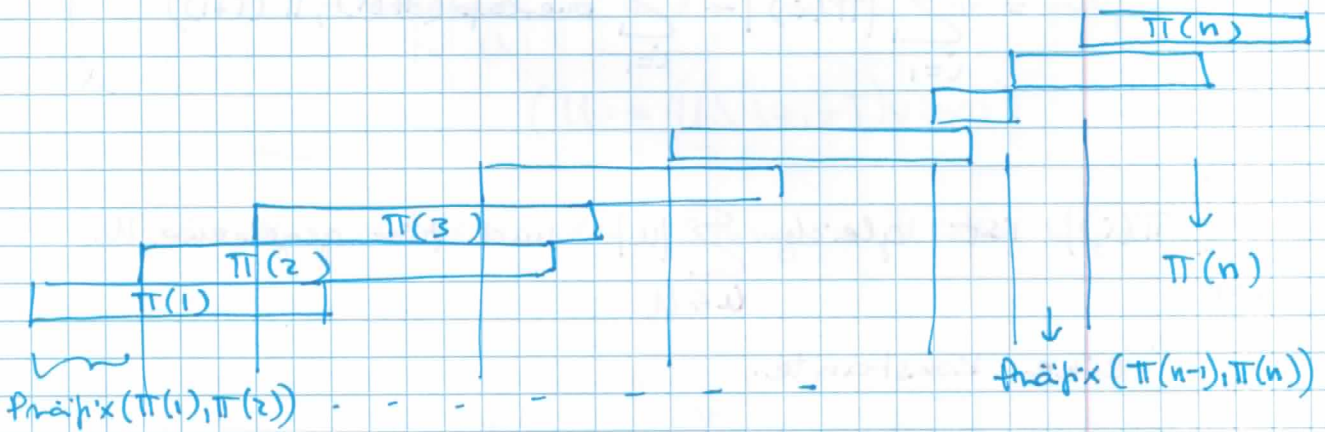


Definition: Eine Bijektion $\pi: \{1, 2, \dots, n\} \rightarrow U$ definiert eine Reihenfolge der Strings in U ($\pi(1), \pi(2), \pi(3), \dots, \pi(n)$ sind die Strings), und damit den kürzesten Superstring $S(\pi)$ für diese Reihenfolge:

$$S(\pi) = \text{präfix}(\pi(1), \pi(2)) \cdot \text{präfix}(\pi(2), \pi(3)) \cdot \dots \cdot \text{präfix}(\pi(n-1), \pi(n)) \cdot \pi(n)$$

↓
Konkatenation

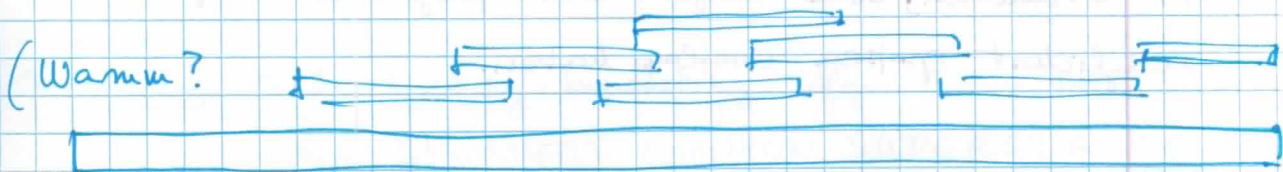
($S(\pi)$ ist der durch π induzierte Superstring von U)



(Die Reihenfolge der Endpositionen bleibt gleich, da kein String einen anderen enthält.)

Beobachtung 1.

Wenn S ein kürzester Superstring von U ist, dann existiert eine Reihenfolge (Bijektion) π , so dass $S = S(\pi)$.



- legen wir jeden $u \in U$ auf eine Position in S wo er vorkommt

- S wird vollständig überdeckt, sonst wäre er kein kürzester Superstring.

- die Reihenfolge der Startpositionen der Strings in U bestimmt π (die gleiche Reihenfolge gilt für ihre Endpositionen)

- Die Überdeckung $\text{overlap}(u, v)$ zwischen Nachbar-Strings ist immer größtmöglich, sonst könnte S kürzer werden.)

Beobachtung? Für die Länge des durch π induzierten Superstrings $S(\pi)$ gilt:

$$|S(\pi)| = \sum_{u \in U} |u| - \sum_{i=1}^{n-1} \text{overlap}(\pi(i), \pi(i+1))$$

(Warum? $|S(\pi)| = \sum_{i=1}^{n-1} |\text{Präfix}(\pi(i), \pi(i+1))| + |\pi(n)| =$

$$= \sum_{i=1}^{n-1} (|\pi(i)| - \text{overlap}(\pi(i), \pi(i+1))) + |\pi(n)| =$$

$$= \sum_{i=1}^n |\pi(i)| - \sum_{i=1}^{n-1} \text{overlap}(\pi(i), \pi(i+1)))$$

$\sum_{i=1}^n |\pi(i)|$ ist gleich $\sum_{u \in U} |u|$ und für gegebene U eine Konstante.

\Rightarrow um einen kürzesten Superstring zu erhalten, brauchen wir eine Reihenfolge $\pi()$ der Strings in U , die die Summe der $\text{overlap}()$ -s maximiert!

(Wir brauchen, dass aufeinander-folgende Strings möglichst großen Overlap haben.)

2. Der Greedy-Superstring Algorithmus

Eingabe: eine Menge U von Strings, so dass keine zwei Strings Teilstings voneinander sind.

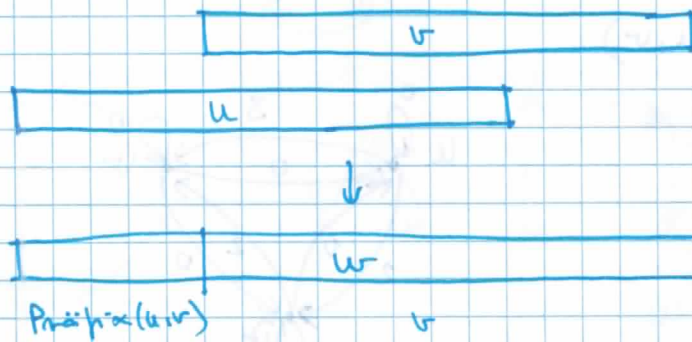
- WHILE $|U| > 1$ DO

- bestimme zwei Strings $u \neq v$ mit maximalem $\text{overlap}(u, v)$

- ersetze u und v durch $w = \text{pr\u00e4fix}(u, v) \cdot v$
in U

$(U := (U \setminus \{u, v\}) \cup \{w\})$

- Gib den verbleibenden String in U als Superstring aus



Theorem: Der Greedy-Superstring Algorithmus ist 4 -approximativ.
f\u00fcr SHORTEST COMMON SUPERSTRING

Vermutung: Er ist sogar 2 -approximativ.

3. Noch ein Approximationsalgorithmus für

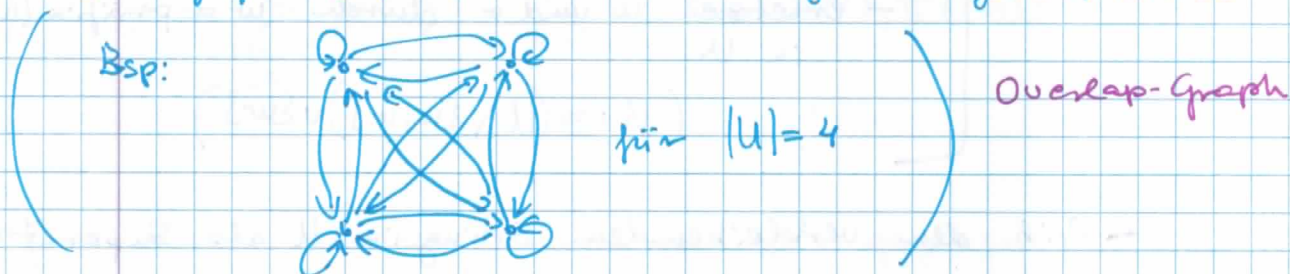
SHORTEST COMMON SUPERSTRING

(der eine minimale sog. Zyklus-Überdeckung benutzt)

Für diesen Algorithmus brauchen wir längere Vorbereitung mit einigen neuen Konzepten und einem weiteren Algorithmus:

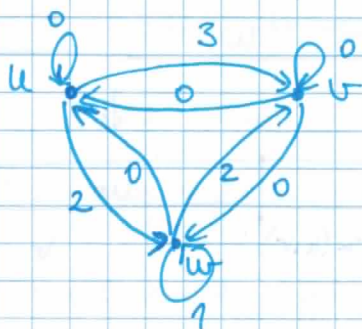
Idee: Bauen wir einen vollständigen gerichteten

Graphen $G(U, E)$ über den Teilstrings als Knoten.
Der Graph enthält auch die Eigenschleifen der Knoten.



Weiterhin sei jede Kante (u, v) gerichtet mit
 $\text{overlap}(u, v)$

u: xabc
v: abcy
w: bcab



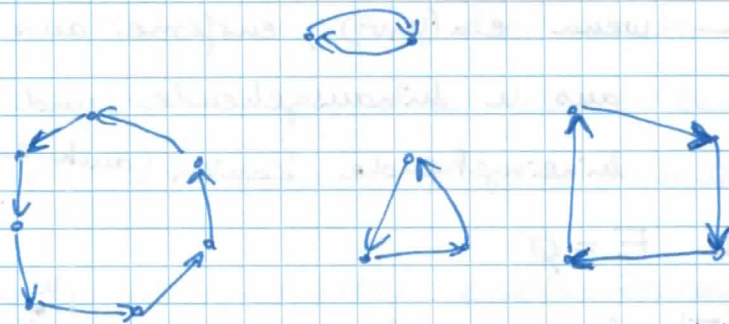
— Einer Reihenfolge $\pi(1) \pi(2) \dots \pi(n)$ der Strings
entspricht jetzt ein ^{gerichteter} Hamiltonscher Pfad in diesem
Graphen (jeder Knoten ist genau einmal ^{besucht} im Pfad)

— Wir wollen ja $\sum_{i=1}^{n-1} \text{overlap}(\pi(i), \pi(i+1))$ maximieren

dies entspricht einem Hamiltonschen Pfad mit
maximalem Gewicht. (= Summe der Kantengewichte)

— leider ist die Bestimmung eines Hamiltonschen Pfades mit maximalem Gewicht ein NP-vollständiges Problem (die Entscheidungsversion). Deshalb begnügen wir uns mit der folgenden schwächeren Lösung: wir suchen eine sog. gerichtete Kreis-Zerlegung mit maximalem Gewicht.

Definition: Eine (gerichtete) Kreis-Zerlegung (directed cycle-cover) von $G(U, E)$ ist ein Teilgraph (U, E') in dem jeder Knoten genau eine ausgehende und eine eingehende Kante hat. (Eine solche Kreis-Zerlegung besteht aus Kreisen, die ^{gemeinsam} jeden Knoten genau einmal überdecken.)



Wieviele Kanten hat jede Kreis-Zerlegung?

Im Folgenden diskutieren wir diese drei Punkte:

- i.) Die Berechnung einer Kreis-Zerlegung mit maximalem Gewicht.
- ii.) Die Bestimmung eines Superstrings S von U von dieser Kreis-Zerlegung.
- iii.) Wie gut $|S|$ die Länge eines kürzesten Superstrings approximiert.

(i) Die Berechnung einer Kreis-Zerlegung mit maximalem Gewicht:

Greedy Algorithmus für maximale Kreis-Zerlegung

Eingabe: ein Overlap-Graph $G(U, \vec{E})$
mit Overlap-Gewichten

→ setze $E' = \emptyset$ (die Kanten der Kreis-Zerlegung)

→ REPEAT

- bestimme die Kante $e \in E$ mit größtem Gewicht

- füge e zu E' hinzu $E' := E' \cup \{e\}$

- wenn $e = (u, v)$, entferne aus E alle aus u hinausgehende, und alle in v hineingehende Kanten, auch die Kante e

UNTIL $E = \emptyset$

→ gib E' als Kreis-Zerlegung aus

Theorem: In einem beliebigen Overlap-Graph gibt der Greedy Algorithmus eine Kreis-Zerlegung mit maximalem Gewicht aus.

Bemerkung: Greedy ist nicht optimal für beliebige positive Kantengewichte! Nur in Overlap-Graphen ist die Optimalität garantiert.

Ein anderer Algorithmus für maximale Kreis-Zerlegung, mit

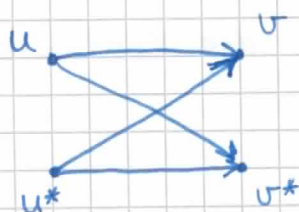
59.

Hilfe eines maximum-MATCHING funktioniert optimal auch für beliebige Kantengewichte.
Der Beweis des Theorems benutzt das folgende Hilfstheorem:

Lemma: Für beliebige Strings u, v, u^*, v^* gilt:

Wenn $\text{overlap}(u, v) \geq \text{overlap}(u, v^*)$ und
 $\text{overlap}(u, v) \geq \text{overlap}(u^*, v)$, dann

$$\text{overlap}(u, v) + \text{overlap}(u^*, v^*) \geq \text{overlap}(u, v^*) + \text{overlap}(u^*, v)$$



beliebige dieser Kanten dürfen Eigenschleifen sein.

(Beweis des Lemma in der Übung)

Beweis des Theorems: (Optimalität des Greedy Alg. für Overlap-Graph)

durch Induktion über die Anzahl n der Knoten/Teilstrings

Basisschritt: für $n=1$ trivial

☞ das ist die einzige Kreiszerlegung

Induktionsschritt: Angenommen, Greedy ist optimal

für n Knoten, dann ist er auch optimal für $n+1$ Knoten.

Sei $G(V, E)$ ein Overlap-Graph auf $n+1$ Knoten.

Sei $e_1 = (u, v)$ eine Kante mit maximalem Overlap.

($u=v$ ist erlaubt)

Wir zeigen:

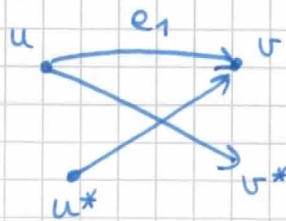
- 1.) Es gibt mindestens eine optimale Kreis-Zerlegung, die e_1 enthält (d.h. der Alg. darf mit e_1 anfangen):

Wann?

Sei $E^* \subset E$ irgendeine optimale Kreis-Zerlegung.

falls $e_1 \in E^* \rightarrow$ fertig mit 1.)

falls $e_1 \notin E^*$, dann gibt es eine u -verlassende, und eine in v -eingehende Kante in E^* , seien diese (u, v^*) und (u^*, v)



($u^* = v^*$, $u^* = v$, usw. ist erlaubt)

Laut Lemma gilt:

$$\text{overlap}(u, v) + \text{overlap}(u^*, v^*) \geq \text{overlap}(u^*, v) + \text{overlap}(u, v^*)$$

Tauschen wir in E^* (u^*, v) und (u, v^*) gegen

(u, v) und (u^*, v^*) ; diese Lösung ist auch optimal, und enthält e_1 .

- 2.) Der Alg. darf mit Greedy weitermachen.

Wann?

Kontrahieren wir die Kante e_1 (identifizieren u und v), nachdem alle u -verlassende und alle in v -eingehende Kanten entfernt wurden. ^{Durch} Nach Kontraktion von e_1 erhalten wir G' , ein Overlap-Graph auf n Knoten (dies entspricht der Verschmelzung der Strings $u \rightarrow v$).

- Eine maximale Kreis-Zerlegung von G' entspricht einer maximalen Kreis-Zerlegung von G , die $e_1 = (u, v)$ enthält.
- Laut Induktionsannahme, ergibt der Greedy Alg. eine maximale Kreis-Zerlegung für G' (einen Overlap-Graph auf n Knoten).

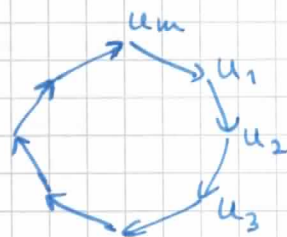
ii.) Die Bestimmung eines Superstrings S für die Menge \mathcal{U} von Teilstrings, mit Hilfe einer optimalen Kreis-Zerlegung

→ Die Idee, wie S erstellt werden soll, ist naheliegend:

Für jeden Kreis C_k in der Kreis-Zerlegung kann ein String S_k definiert werden, der (mit großen Overlaps) alle Strings enthält, die den Knoten des Kreises C_k entsprechen.

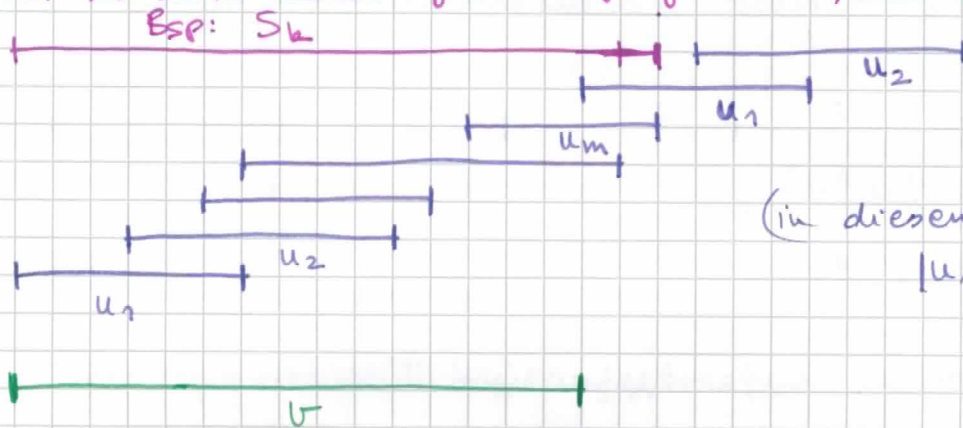
Anschließend werden die Strings S_k für jeden Kreis konkateniert, um einen Superstring S für alle Strings in \mathcal{U} zu erhalten.

→ Wie wird S_k definiert? Insbesondere, was für ein String "entspricht" einem gerichteten Kreis $C_k (u_1 u_2 u_3 \dots u_m u_1)$ im Overlap-Graphen?



In dem Kreis C_k überlappen sich die Strings u_1, u_2, \dots, u_m zyklisch (es ist egal welchen String wir als u_1 wählen)

S_k sei der kürzeste ~~String~~ String vom Anfang von u_1 , der alle enthält.



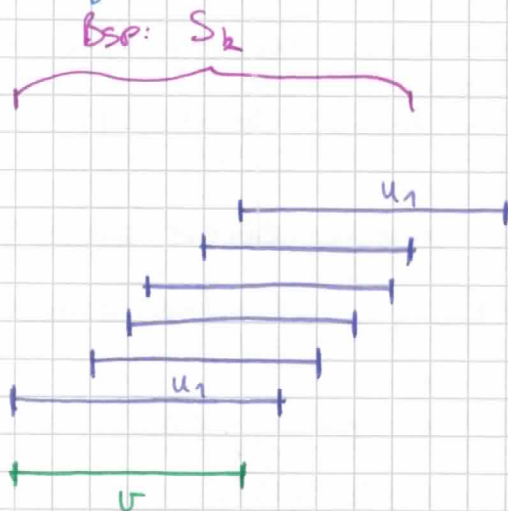
Sei v der String vom Anfang von u_1 bis zum Anfang vom nächsten u_1 in dieser periodischen Darstellung.

es gilt: S_k ist ein Präfix von $v \cdot u_1$, und

$$|S_k| \leq |v| + |u_1|$$

Enthält v alle Strings u_1, u_2, \dots, u_m ? \rightarrow NEIN

Es kann sogar vorkommen, dass v keinen ganzen String aus u_1, u_2, \dots, u_m enthält:



dies passiert, wenn

$$|v| < |u_1|$$

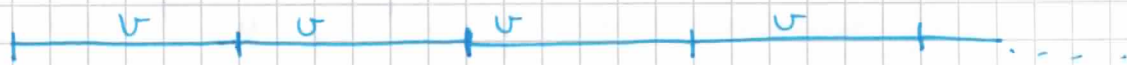
u_1 sich überlappt, und

u_1 ist periodisch mit Periode v

auch hier gilt $|S_k| \leq |v| + |u_1|$

Bemerkung:

Sei $v^\infty = v v v v v \dots$ der unendlich oft mit sich selbst konkatenierte String v



Es gilt:

→ u_1, u_2, \dots, u_m sind Teilstrings von v^∞ , und diejenigen mit $|u_i| > |v|$ sind zyklisch (periodisch) mit Periode v

→ man sagt, dass v die Strings u_1, u_2, \dots, u_m zyklisch überdeckt; ~~was~~ man nennt v einen Zyklus des Kreises C_k (wird ein anderer String im Kreis als u_i gewählt, wird v einfach zyklisch verschoben)

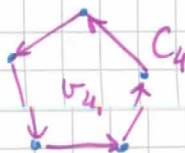
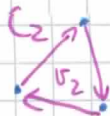
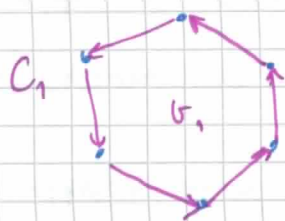
→ Um S_k zu erhalten "brechen" wir den Zyklus v "auf" und konkatenieren mit sich selbst bis er alle Strings im Kreis überdeckt.

Betrachten wir jetzt alle Kreise einer Kreis-Zerlegung

E' : $C_1 C_2 \dots C_k \dots C_\ell$, und ihre entsprechenden

Zyklen: $v_1 v_2 \dots v_k \dots v_\ell$

E'



Algorithmus für SHORTEST COMMON SUPERSTRING

- bestimme eine Kreis-Zerlegung mit maximalem Gewicht im Overlap-Graph, (mit dem Greedy Algorithmus)

$$C_1 \ C_2 \ \dots \ C_k \ \dots \ C_e$$

- sei v_k der Zyklus von Kreis C_k ($u_1^k \ u_2^k \ \dots \ u_{m_k}^k$)

für jedes k

- sei S_k der Zyklus v_k aufgebrochen und mit sich konkateniert, so dass S_k jeden String in C_k enthält.

$$\text{Es gilt: } |S_k| \leq |v_k| + |u_1^k|$$

- gib die Konkatenation von jedem S_k als Superstring aus:

$$S = S_1 \cdot S_2 \cdot S_3 \cdot \dots \cdot S_k \cdot \dots \cdot S_e$$

(iii.) Approximationsfaktor

Theorem: Dieser Algorithmus für SHORTEST COMMON SUPERSTRING ist 4-approximativ.

(Wenn S aus $S_1 \ S_2 \ \dots \ S_e$ mit dem Greedy-Superstring Algorithmus erzeugt wird, dann ist der Alg. 3-approximativ.)

Wenn die Strings in U nicht-periodisch sind

(wenn für mindestens einen String pro Kreis $|u_i^k| < |v_k|$ gilt), dann ist der Algorithmus sogar 2-approximativ.

Beweis: (des Approximationsfaktors 4)

Sei σ der Zyklus des Kreises $C(u_1 u_2 u_3 \dots u_m u_1)$

→ Dann ist

$$\sigma = \text{Präfix}(u_1, u_2) \cdot \text{Präfix}(u_2, u_3) \cdot \dots \cdot \text{Präfix}(u_{m-1}, u_m) \cdot \text{Präfix}(u_m, u_1)$$

→ und seine Länge:

$$|\sigma| = \sum_{i=1}^m |\text{Präfix}(u_i, u_{i+1})| = \left(\text{wobei } i+1 \text{ 'modulo } m \text{' berechnet wird} \right)$$

$$i+1 = 1 \pmod{m}$$

$$= \sum_{i=1}^m |u_i| - \sum_{i=1}^m \text{overlap}(u_i, u_{i+1}) = \pmod{m}$$

$$= \sum_{i=1}^m |u_i| - \text{Gewicht des Kreises } C(u_1 \dots u_m) \text{ in der Kreis-Zerlegung}$$

→ Addieren wir die Längen der Zyklen σ_k über alle Kreise der Kreis-Zerlegung

$$\sum_{k=1}^l |\sigma_k| = \sum_{i=1}^n |u_i| - \sum_{(u, \sigma) \in E'} \text{overlap}(u, \sigma) =$$

$$= \sum_{i=1}^n |u_i| - \text{Gewicht der Kreis-Zerlegung}$$

Definition: $\sigma_1 \sigma_2 \dots \sigma_l$ (die Zyklen aller Kreise einer Kreis-Zerlegung E') heißt eine Zyklus-Überdeckung von U weil jeder $u \in U$ von einem σ_k zyklisch überdeckt wird.

Beobachtung: v_1, v_2, \dots, v_k ist eine minimale Zyklus-Überdeckung ($\sum_{k=1}^k |v_k|$ minimal) genau dann, wenn E' eine maximale Kreis-Zerlegung ist.

Vergleich mit der Länge von $S(\pi)$, (des von π induzierten Superstrings)

$$|S(\pi)| = \sum_i |u_i| - \text{Gewicht}(\text{Ham. Pfad})$$

$$\sum_{k=1}^k |v_k| = \sum_i |u_i| - \text{Gewicht}(\text{Kreis-Zerlegung})$$

$$\min_{\text{Superstring } S(\pi)} |S(\pi)| \iff \min_{\pi} |S(\pi)| \iff \text{max. Gewicht Hamilt. Pfad}$$

NP-schwer

$$\min_{v_1, v_2, \dots, v_k} \text{Zyklus-Überdeckung} \iff \min_{k=1}^k |v_k| \iff \text{max. Gewicht Kreis-Zerlegung}$$

polynomiell optimierbar

Wir erhalten: Für eine minimale Zyklus-Überdeckung

v_1, v_2, \dots, v_k und einen kürzesten Superstring S^* von U gilt

$$\sum_{k=1}^k |v_k| \leq |S^*|$$

→ Bei nicht-periodischen Strings:

falls für mind. einen String u_i^k in jedem Kreis

$$|u_i^k| \leq |v_k|, \text{ dann}$$

$$|S_k| \leq |v_k| + |u_i^k| \leq 2|v_k|$$

$$|S| \leq \sum_{k=1}^l |S_k| \leq 2 \cdot \sum_{k=1}^l |v_k| \leq 2 \cdot |S^*|$$

d.h. die Ausgabe S des Algorithmus ist
2-approximativ.

→ Was wenn $|v_k| \ll |u_i^k|$?

Für beliebige (auch periodische) Strings:

Wir betrachten den ersten String in jedem Kreis:

$$u_1^1 u_1^2 \dots u_1^k \dots u_1^l$$

Sei S' ein Superstring minimaler Länge von diesen
ersten Strings, wobei wir annehmen, dass sie in S'
in dieser Reihenfolge vorkommen.

$$|S^*| \geq |S'| = \sum_{k=1}^l |u_1^k| - \sum_{k=1}^{l-1} \text{overlap}(u_1^k, u_1^{k+1}) \stackrel{(*)}{\geq}$$

$$\stackrel{(*)}{\geq} \sum_{k=1}^l |u_1^k| - \sum_{k=1}^{l-1} (|v_k| + |v_{k+1}|) \geq \sum_{k=1}^l |u_1^k| - 2 \sum_{k=1}^l |v_k|$$

$$\Rightarrow \sum_{k=1}^l |u_1^k| \leq |S^*| + 2 \sum_{k=1}^l |v_k|$$

Damit folgt der Approximationsfaktor 4:

$$|S| = \sum_{k=1}^l |S_k| < \sum_{k=1}^l |u_1^k| + \sum_{k=1}^l |v_k| \leq |S^*| + 3 \sum_{k=1}^l |v_k| \leq 4|S^*|$$

⊗ folgt aus dem folgenden Overlap-Lemma:

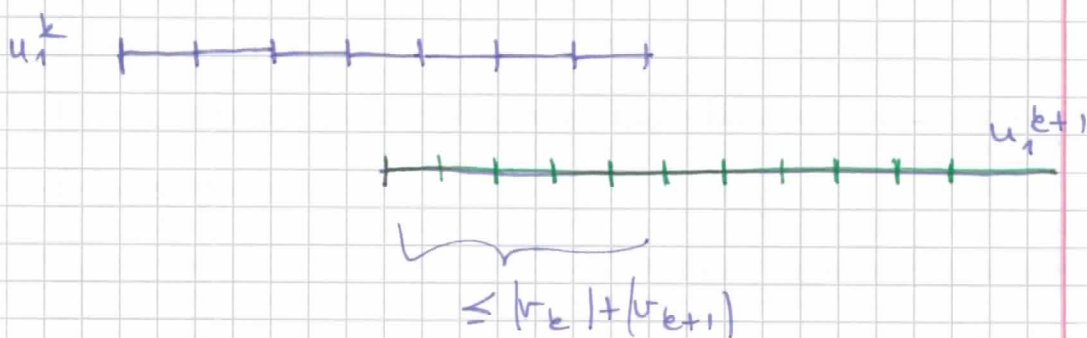
Overlap-Lemma: Wenn u_1^k zyklisch ist mit

Periode v_k , und u_1^{k+1} zyklisch ist mit Periode v_{k+1} ,

(und v_k keine zyklische Verschiebung von v_{k+1} ist),

dann gilt

$$\text{overlap}(u_1^k, u_1^{k+1}) \leq |v_k| + |v_{k+1}|$$



Bemerkung: Da in der Ausgabe S vom Algorithmus

die verschiedenen S_k Strings nicht übereinander geschoben werden, sollte gezeigt werden, dass sie auch im optimalen Superstring S^* nicht lang übereinander geschoben werden können. Dies folgt aus dem Overlap-Lemma.

MATROIDE (zu greedy Algorithmen)

a.) Motivation

Erinnern wir uns zunächst an Kruskal's Algorithmus für minimale Spannbäume. Genauer gesagt: wir schreiben hier den Algorithmus von Kruskal um einen maximalen Spannb Baum zu berechnen (das macht keinen wesentlichen Unterschied, ist aber besseres Beispiel zu Matroiden).

Der Greedy Algorithmus für max. SPANNBAUM (Wiederholung)

Eingabe: Graph $G(V, E)$ mit Kantengewichten (zusammenhängend)

$S := \emptyset$ (Menge der gewählten Baumkanten)

WHILE $E \neq \emptyset$ DO

- entferne aus E eine Kante $e \in E$ mit maximalem Gewicht w_e ; setze $E := E \setminus \{e\}$
- falls e keinen Kreis schließt mit (manchen) Kanten aus S (d.h. $S \cup \{e\}$ kreisfrei) dann setze $S := S \cup \{e\}$
- sonst verwerfe e

gib S aus

Warum funktioniert dieser Typ von Greedy Algorithmus für maximale (oder minimale) Spannbäume, und nicht für viele andere Probleme wie z.B. max-INDEPENDENT SET? Im Folgenden beschäftigen wir uns mit dieser Frage...

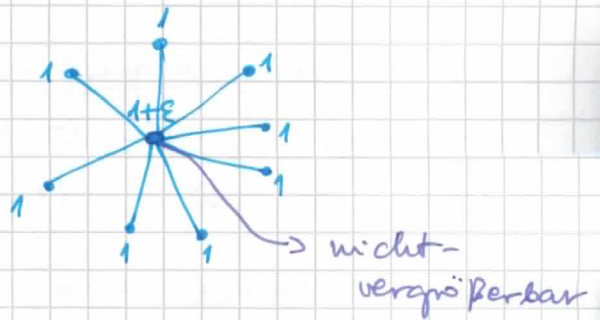
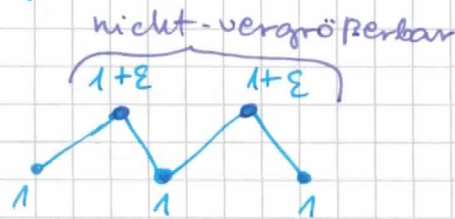
Erstmal eine grobe Erklärung ...

→ Was wäre ein analoger greedy Algorithmus für max-INDEPENDENT SET?

„Nimm Knoten mit maximalem Gewicht zu I hinzu, falls I eine unabhängige Knotenmenge bleibt (entferne die bereits betrachteten Knoten aus V)“

Beispiele, warum dieser Algorithmus max-IS nicht

optimal löst:



dieser Algorithmus für max-IS ist schon mal deswegen nicht optimal, weil

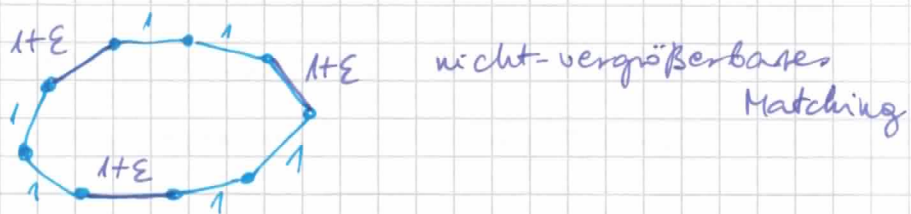
nicht-vergrößerbare unabhängige Knotenmengen (Engl: maximal)

// Eigenschaft des Mengensystems nicht der Gewichte (aller unabhängigen Mengen)

größtmögliche unabhängige Knotenmengen (Engl: largest maximum)

(größtmöglich ist immer auch nicht-vergrößerbar, aber andersrum nicht immer)

→ Ein anderes Problem für das ein solcher Greedy Algorithmus nicht optimal sein kann, ist max-MATCHING



Beachte, dass diese Gegenbeispiele funktionieren (d.h. Greedy funktioniert nicht) sogar wenn alle Gewichte gleich ($=1$) sind, d.h. in der ungewichteten Variante des Problems.

In dem ungewichteten Fall gilt sogar, dass die obige Eigenschaft eine entscheidende (äquivalente) Eigenschaft ist für das Funktionieren/nicht-Funktionieren von Greedy:

Der Greedy-Algorithmus funktioniert für die ungewichtete Variante "solcher" Probleme genau dann wenn alle nicht-vergrößerbare "gesuchte" (^{kreisfreie} unabhängige) Teilmengen auch ^{gleichgroß} größtmöglich sind. nenne es "einfache Maximalitätseigenschaft"

(Prüfen wir diese einfache Eigenschaft für kreisfreie Kantenmengen, (die im max-SPANNBAUM Problem gesucht werden): eine nicht-vergrößerbare kreisfreie Kantenmenge ist immer ein Spannbaum mit genau $|V|-1$ Kanten, also auch größtmöglich) (allgemein Spannwald mit $|V|-r$ Kanten, siehe später)

Für die (eigentliche) gerichtete Varianten unserer Probleme muss eine Maximalitätseigenschaft in viel stärkerer Form erfüllt sein, damit der Greedy Algorithmus immer optimal ist (und umgekehrt: ~~ist~~ ^{ist} der Greedy Alg. für jede Gewichtung optimal, ~~dann ist~~ ^{wenn} diese Maximalitätseigenschaft erfüllt ^{ist}), nämlich "nicht-vergrößerbar \Leftrightarrow größtmöglich" muss in jedem Teilproblem (Teilmenge) der Eingabe erfüllt sein (klar, weil die restlichen Elemente z.B. Gewicht $=0$ haben können, dann muss man im gegebenen Teilproblem maximieren).

14. b.) Beispiel Matroidaleigenschaften für kreisfreie Kantenmengen (das Mengensystem von) (in einem fix. Graphen)

Wir definieren und weisen nach die Maximalitätseigenschaft an Beispiel von kreisfreien Kantenmengen:

Maximalitätseigenschaft für kreisfreie Kantenmengen:

Alle nicht-vergrößerbaren kreisfreien Teilmenge einer beliebigen fixierten Kantenmenge $E' \subseteq E$ in einem beliebigen ^{fixierten} Graphen $G(V, E)$ haben dieselbe Größe (und sind somit größtmögliche solche Teilmengen).

Für den Beweis brauchen wir die folgende Beobachtung:

Sei G ein beliebiger Graph mit r Zusammenhangskomponenten die jeweils n_1, n_2, \dots, n_r Knoten haben. Dann hat ~~ein~~ nicht-vergrößerbare kreisfreie Kantenmenge (ein sog. Spannwald) genau $n_1 - 1 + n_2 - 1 + n_3 - 1 + \dots + n_r - 1 = n - r$ Kanten, weil ein Spannbaum der i -ten Zusammenhangskomponente $n_i - 1$ Kanten hat.

Beweis der Maximalitätseigenschaft: $G'(V, E')$ ist

auch ein Graph (ein Teilgraph von G), und die nicht-vergrößerbaren kreisfreien Kantenmengen in G' haben laut Beobachtung dieselbe Größe $n - r$ (wenn G' r Zusammenhangskomponenten hat). \square

(In einer Zusammenhangskomponente kann man eine kreisfreie Kantenmenge solange vergrößern bis die Kantenmenge ein Spannbaum ist



Die folgende Eigenschaft ist scheinbar stärker als die Maximalitätseigenschaft, wird sich aber als äquivalent zu ihr herausstellen. Wir formulieren und beweisen die Eigenschaft für kreisfreie Kantenmengen:

Ergänzungseigenschaft für kreisfreie Kantenmengen:

Seien $E_1, E_2 \subseteq E$ kreisfreie Kantenmengen, so dass

$|E_1| < |E_2|$, dann gibt es eine Kante $e \in E_2 \setminus E_1$

so dass $E_1 \cup \{e\}$ immer noch kreisfrei ist.

(d.h. kreisfreie Mengen die nicht größtmöglich sind kann man erweitern/vergrößern aus einer beliebigen größeren kreisfreien Menge; dasselbe gilt nicht z.B. für unabhängige Knotenmengen oder Matchings)

Beweis der Ergänzungseigenschaft:

Aufwärmung: sei E_2 ein Spannbau, also zusammenhängend.

E_1 ist noch kein Spannbau, weil $|E_1| < |E_2|$, also nicht zusammenhängend;

denn gibt es zwei Knoten u, v

die in E_1 nicht mit einem Weg

verbunden sind, aber in E_2 mit

einem (eindeutigen) Weg verbunden sind.

Es gibt (mindestens) eine Kante e auf diesem Weg die

verschiedene Komponenten von E_1 verbindet und deshalb

$E_1 \cup \{e\}$ kreisfrei ist.

[Der Beweis ist fast dasselbe wenn E_2 auch kein Spannbau ist: (V, E_2) hat $n - |E_2|$ Zusammenhangskomponenten und (V, E_1) hat $n - |E_1| > n - |E_2|$ Zusammenhangskomponenten.