Yannick Müller muelleya@student.ethz.ch
yajm.ch

ETH

MADE EASY

Computer Science
Basisprüfung - Block 2

August 2019

Algorithm & Probabilities - Lecture

Prof. Angelika Steger
TA: schweril@student.ethz.ch

Contents

1 Graphentheorie **2**

1.1 Kreise . . . 3

Algorithmus: Eulertour finden . . . 3

1.2 NP-Problem . . . 5

1.3 Traveling Salesman Problem . . . 5

Algorithmus: α- Approximations . . . 5

1.4 Matchings . . . 6

Algorithmus: Find Perfect Matchings . . . 6

Algorithmus: 3.5 Approximationsalgorithmus . . . 6

1.5 Färbungen . . . 7

2 Wahrscheinlichkeit **8**

2.1 Bedingte Wahrscheinlichkeit . . . 8

2.2 Unabhängigkeit . . . 9

Algorithmus: Finde stabile Menge . . . 10

2.3 Alle n Bilder sammeln . . . 10

2.4 Poisson Verteilung . . . 10

2.5 Varianz . . . 10

2.6 Abschätzen von Wahrscheinlichkeiten . . . 11

2.6.1 Markov-Ungleichung . . . 11

2.6.2 Chebyshev-Ungleichung . . . 11

2.6.3 Chernoff-Ungleichung . . . 12

2.7 Randomisierte Algorithmen . . . 12

2.8 Fehlerreduktion . . . 12

Algorithmus: QuickSelect . . . 13
2.9 Target-Shooting . . . 13
2.10 Hashing und Zuordnungsverfahren . . . 13
2.11 Minimum Cut . . . 14
2.12 Bootstrapping . . . 14
2.13 Primzahltest . . . 14
2.14 Color-Coding . . . 15
2.15 Lange Pfade . . . 15
2.16 Flüsse in Netzwerken . . . 16
2.17 Flussproblem . . . 17
2.18 Intern Kanten-disjunkte Pfade in gerichteten Graphen . . . 17
2.19 Bildsegmentierung . . . 18

3 Geometrische Algorithmen **18**

3.1 Kleinster umschiessender Kreis . . . 18
3.2 Konvexe Hülle . . . 19
3.3 Voronoi Diagram . . . 19
3.4 Schöner Graph Zeichnen . . . 20
3.5 Manhatten Norm . . . 20
3.6 Finden einer konvexen Hülle . . . 20
Algorithmus: JarvisWrap . . . 20

The following was presented in the lecture on 18. February 2019.

1 Graphentheorie

k-Zusmmanhängend bedeutdet, man muss mindestens k Kanten wegnehmen um den zusammenhangenden Graphen zu zerstören.

Theorem 1.1: Menger

G ist k-zusammenhängend genau dann wenn $\forall u, v \in V,\ u \neq v$ gibt es k intern-knotendisjunkte u-v-Pfade.
G ist k-Kanten-zusammenhängend genau dann wenn $\forall u, v \in V,\ u \neq v$ gibt es k intern-kantendisjunkte u-v-Pfade.

k-Kanten-zusammenhängend sagt wieviele Kanten man wegnehmen müsste.
Es gilt immer:
Knotenzusammenhang ≤ Kantenzusmmenhang ≤ minimaler Grad
Aritkulationknoten (eng.:cut vertex) genau dann wenn G[V {v}] nicht zusammenhängend ist.
Brücke (eng.:cut edge) genau dann wenn G - e nicht zusammenhängend ist.

Lemma 1.2

Sei G=(V,E) ein Zusammenhängender Graph. Ist $\{x, y\} \in E$ eine Brücke so gilt: $deg(x) = 1$ oder x ist Aritkulationsknoten.

low-Nummer: Kleinste dfs-Nummer, die man von v aus durch einen Pfad aus Vorwärtskanten und maximal einer Rückwärtskante erreichen kann.
v ist ein Artikulationsknoten $\Leftrightarrow$ $v = s$ und s hat in T Grad mindestens zwei, oder $v \neq s$ und es gibt $w \in V$ mit $\{v, w\} \in E(T)$ und low[w] ≥ dfs[v].

The following was presented in the lecture on 21. February 2019.

Vorwärtskante Kanten die im Baum nach unten zeigen

Rückwärtskante Kanten die im Baum nach oben zeigen

Ein Block ist eine maximale Menge von Kanten, so dass je zwei dieser Kanten auf einem gemeinsamen Kreis liegen.

1.1 Kreise

Hamiltonkreis Ein Kreis in G, der jeden Knoten genau einmal enthählt

Eulertour Ein geschlossener Weg in G, der jede Kante genau einmal enthählt/

Theorem 1.3

Ein zusammenhängender Graph G=(V,E) enthählt eine Eulertour genau dann wenn der Grad jedes Knotens gerade ist. Eine solche kann man in $\mathcal{O}(|E|)$ Zeit finden.

Algorithmus 1.1: Eulertour finden

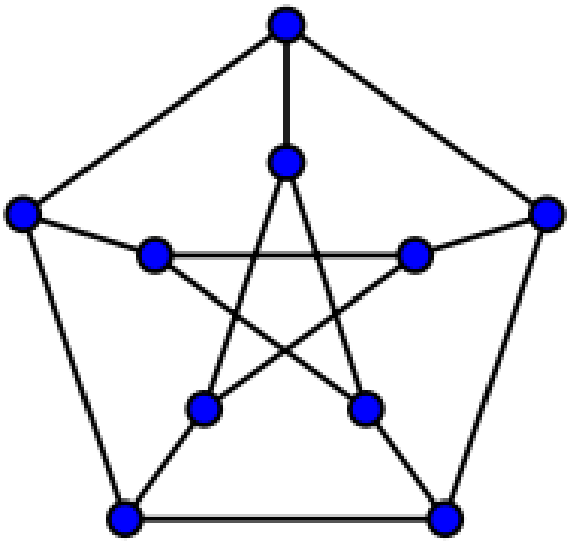
1. Läufer findet einen Kreis
2. Schildköte läuft hinterher und lässt überall wo es noch nicht abgelaufene Kanten hat,

den Läufer wieder nach einem Weg suchen.

3. Die Schildkröte verknüpft den alten und den neuen Kreis.
4. Schildkröte läuft den neuen Weg entlangt und wiederholt das Ganze.

Laufzeit $\mathcal{O}(|E|)$

Petersongraph



Theorem 1.4

Ein $n \times m$ Gitter enthählt einen Hamiltonkreis genau dann wenn $n \times m$ gerade ist.

The following was presented in the lecture on 26. February 2019.

Theorem 1.5

Ein zusammenhängender Graph enthählt eine Eulertour genau dann wenn der Grad jedes Knotens gerade ist.

Algorithmus 1.2: Finde Eulerkreis

```
1 W ← Randomtour(v_start)
2 v_langsam ← Startknoten von W
3 while v_langsam ist nicht letzter Knoten in W do
4     v ← Nachfolger von v_langsam in W
5     if N_G(v) ≠ ∅ then
6         W' ← Randomtour(v)
7         W ← W_1 + W' + W_2
8     v_langsam ← Nachfolger von v_langsam in W
9 return W
```

Laufzeit: $\mathcal{O}(|E|)$ (E = Anzahl Edges

Bei einem d-dimensionalen Hyperwürfel unteschieden sich alle Knotenpaare an genau einer Koordiante.
Bei einem Hamiltonkreis wird jeder Vertex genau einmal besucht.

Theorem 1.6: Dirac

$G = (V, E)$, $|v| \geq 3$ sodass $\forall v \in V \ deg(v) \geq \frac{|v|}{2} \Rightarrow$ G enthählt einen Hamiltonkreis.

The following was presented in the lecture on 28. February 2019.

Theorem 1.7: Karp

Das Problem: "Enthählt ein Graph einen Hamiltonkreis?" ist NP-vollständig.

1.2 NP-Problem

P Effizient entscheidbare Probleme

NP effizient verifizierbare Probleme

Ein Problem Π aus NP heisst NP-vollständig, falls gilt:

$$\Pi \in P \Rightarrow P = NP \tag{1}$$

Die Frage ist nun, ob gilt $P = NP$?

1.3 Traveling Salesman Problem

Gegeben ein vollständiger Graph und gesucht ist die kürzeste Rundreise die all Knoten besucht.

Note 1. Ein Hamiltonkreis ist ein Spezialfall von Traveling Salesman Problem α - Approximationsalgorithmus $\mathscr{A}$

$$\mathscr{A}(n, l) \leq \alpha \cdot OPT(n, l) \quad \forall n \forall l \tag{2}$$

Algorithmus 1.3: α- Approximations

Eingabe: $n \in \mathbb{N}, l \cdot (\binom{[n]}{2} \to \mathbb{R}_{\geq 0})$
Notation: $OPT(n, l)$ = Länge des kürzestens Hamiltonkreises
$MSt(n, l)$ = Länge eines minimalen Spannbaumes
Dann gilt:

$$MST(n,l) \geq OPT(n,l) \tag{3}$$

The following was presented in the lecture on 05. March 2019.

1.4 Matchings

Definition 1. Ein Matching M in einem Graphen $G = (V, E)$ ist eine Teilmenge der Kanten (also $M \subseteq E$) mit der Eigenschaft $\forall v \in V$: Grad von v bezüglich M ist ≤ 1

Definition 2. Ein Matching M heisst perfekt, genau dann wenn $|M| = \frac{|v|}{2}$

Theorem 1.8

$\forall k \in \mathbb{N}$ gilt in einem 2^k - regulären bipartiten Graphen kann man m $\mathcal{O}(|E|)$ ein perfektes Matching bekommen.

Algorithmus 1.4

```
for i=k downto 1 do:
    Fuer jede Zusammenhangskomponente bestimme Eulertour und loesche jede zweite
        Kante.
```

Laufzeitanalyse:
$\mathcal{O}(|E|)$

Theorem 1.9

$G = (A \cup B, E)$ bipartit und l-regulär $\Rightarrow |A| = |B|$

Corollary 1.1

⟨ wie oben ⟩ ... für jede Zusammenhangskomponente gilt # Knoten ist gerade.

Theorem 1.10

Es gilt immer:

$$|M_{Greedy}| \geq \frac{1}{2}|M_{max}| \tag{4}$$

The following was presented in the lecture on 07. March 2019.

Algorithmus 1.5

ingabe: $V \cdot [n]$ und $l : \binom{[n]}{e} \to \mathbb{N}_0$, so dass die Dreiecksungleichung gilt.
Ausgabe: Hamiltonkreis C mit $l(C) \leq \frac{3}{2} \cdot minl(e)$.

1. Bestimme minimaler Spannbaum bezüglich $l \rightsquigarrow T$
2. $S := \{v \in V$ v hat ungeraden Grad in $T\}$
   Bestimme minimal perfektes Matching: $K[S]$ $l \rightsquigarrow M$ dann gilt alle Knoten in $T \cup M$ haben geraden Grad.
3. Bestimme Eulertour in $T \cup M$, durchlaufe mit Abkürzungen.

Laufzeit: $\mathcal{O}(|V|^3)$

`https://uva.onlinejudge.org/` Milenium Ceremony

Theorem 1.11: Heiratssatz

Sei $G = (A \cup B, E)$ ein bipartiter Graph. Dann gilt:
$\exists$ Matching M das alle Knoten aus A überdeckt

$$\Leftrightarrow \forall x \subseteq A : \quad |x| \leq |\Gamma(x)| \tag{5}$$

Note 2. $A \dot{\cup} B$ bedeutet, das die Mengen disjunkt sind.

The following was presented in the lecture on 12. March 2019.

Corollary 1.2: Forbenius

Für alle k gilt: jeder k-reguläre bipartiete Grpah enthählt ein perfektes Matching.

1.5 Färbungen

k-Färbung / chromatische Zahl ist die minimale Anzahl Farben, die für eine Knotenfärbung von G benötigt wird.

Algorithmus 1.6

Gegeben: Graph $G = (V, E)$
Aufgabe: Färbe mit möglichst wenigen Farben.
Greedy-Algorithmus
Laufzeit: $\mathcal{O}(|V| + |E|)$

Theorem 1.12

$G \neq K_n, \quad C_{2n+1}$ zusammenhängend, mit Adjazenzlisten.
$\Rightarrow \mathcal{O}(|E|)$ kann man eine Färbung mit $\Delta(G)$ vielen Farben bestimmen.

The following was presented in the lecture on 14. March 2019.

2 Wahrscheinlichkeit

Definition 3. Ein diskreter Wahrscheinlichkeitsraum ist bestimmt durch eine Ergebnismenge $\Omega = \{\omega_1, \omega_2, ...\}$ bestimmt welche je eine Wahrscheinlichkeit zwischen $0 \leq x \leq 1$ und die Summe ist 1.

Theorem 2.3: Siebformel

$$\Pr[\bigcup_{i=1}^{n} A_i] = \sum_{i=1}^{n} \Pr[A_i] - \sum_{i\leq 11\leq l2\leq n} \Pr[A_{i1} \cap A_{i2}] \pm ... + (-1)^{i+1} \cdot \sum_{i\leq 11\leq ..\leq lq\leq n} \Pr[A_{i1} \cap ... \cap A_{iq}] ... \quad (6)$$

Definition 4. Laplace Raum - endlicher Wahrscheinlichkeitsraum, in dem alle Elementarergeignisse gleich wahrscheinlich sind.

2.1 Bedingte Wahrscheinlichkeit

Definition 5. Bedingte Wahrscheinlichkeit ist definiert durch:

$$\Pr[A|B] \stackrel{\text{def}}{=} \frac{\Pr[A] \cap \Pr[B]}{\Pr[B]} \quad (7)$$

The following was presented in the lecture on 19. March 2019.

Theorem 2.12: Additionssatz

$A_1, ..., A_n$ seien paarweise disjunkt. und alle A zusammen geben den Wahrscheinlichkeitsraum, dann gilt:

$$\Pr[B] = \sum_{i=1}^{n} \Pr[B|A_i] \cdot \Pr[A_i] \quad (8)$$

Theorem 2.10: Multiplikationssatz

$$\Pr[A_1 \cap ... \cap A_n] = \Pr[A_1] \cdot \Pr[A_2|A_1] \cdot \Pr[A_3|A_1 \cap A_2] \cdot ... \cdot \Pr[A_n|A_1 \cap ... \cap A_{n-1}] \quad (9)$$

Note 3. $1 - x \leq e^{-x}$

2.2 Unabhängigkeit

Definition 6. Zwei Ereignisse sind unabhängig, wenn

$$\Pr[A \cap B] = \Pr[A] \cdot \Pr[B] \tag{10}$$

Note 4. Ereignisse können stochastisch unabhängig sein, ohne dass sie physikalisch unabhängig sind.

The following was presented in the lecture on 21. March 2019.

Definition 7. Ereignisse sind unabhängig, wenn für alle Teilmengen $I \subseteq \{1, ..., n\}$ mit $I = \{i_1, ..., i_k\}$ gilt:

$$\Pr[A_{i_1} \cap ... \cap A_{i_k}] = \Pr[A_{i_1}] \cdot ... \cdot \Pr[A_{i_k}] \tag{11}$$

Lemma 2.1

Wenn A,B, C unabhängig sind dann gilt auch das $A \cap B$ und C oder $A \cup B$ und C unabhängig.

Note 5. In der Wahrscheinlichkeitstheorie werden Funktionen als $X : \Omega \to \mathbb{R}$ bezeichnet.

Note 6. $X \leq 5 \Leftrightarrow \{\omega \in \Omega : X(\omega) \leq 5\}$

Note 7. $W_x \Leftrightarrow \{X(\omega) : \omega \in \Omega\} \overset{\text{def}}{=}$ Wertebereich der Zufallsvariable

Definition 8. Erwartungswert: $\mathbb{E}[x] = \sum_{\omega \in \Omega} X(\omega) \cdot \Pr[w]$

The following was presented in the lecture on 28. March 2019.

Theorem 2.11

Sei X eine Zufallsvariable mit $W_X \subseteq \mathbb{N}_0$. Dann gilt:

$$\mathbb{E}[X] = \sum_{i=1}^{\infty} \Pr[X \geq i] \tag{12}$$

Definition 9. Eine Zufallsvariable heisst Gedächnisslos, falls gilt $\forall s, t$

$$\Pr[X \geq s + t \mid X > s] = \Pr[X \geq t] \tag{13}$$

Theorem 2.12

$X \sim \text{Geo(p)} \Rightarrow X$ Gedächnisslos
Bei Geometrischer Verteilung gilt: $\mathbb{E}[X] = \frac{1}{p}$

Algorithmus 2.1: Finde stabile Menge

Gegeben: Graph $G = (V, E)$
Ziel: Finde eine möglichst grosse stabile Menge

---

1 $\forall v \in V$: `Werfe Muenze mit Wahrscheinlichkeit p fuer Kopf, falls Zahl loesche v.`
2 $\forall e \in E$: `Falls beide Knoten noch da, loesche einen der beiden Knoten.`

$$\Rightarrow \mathbb{E}[S] \geq n \cdot p - m \cdot p^2 \tag{14}$$

2.3 Alle n Bilder sammeln

Es gibt n verschiedene Bilder in jeder Runde erhalten wir gleichwahrscheinlich eines der Bilder
Gesucht ist die Anzahl der Runden bis wir alle n Bilder besitzen.
Lösungsanzatz: Phase i: Runden während wir i-1 verschieden Bilder besitzen.

$$X_i \sim Geo\left(\frac{n-(i-1)}{n}\right) \tag{15}$$

$$\mathbb{E}[X] = \sum_{i=1}^{n} \mathbb{E}[X_i] = n \cdot (\ln(n) + \mathcal{O}(1)) \tag{16}$$

The following was presented in the lecture on 02. April 2019.

Theorem 2.13: Linearität des Erwartungswertes

$$\mathbb{E}[X] = a_1\mathbb{E}[X_1] + ... + a_n\mathbb{E}[X_n] + b \tag{17}$$

Bestimmung einer staiblen Menge

$$\mathbb{E}[S] \geq np - mp^2 \tag{18}$$

2.4 Poisson Verteilung

Grenzwert von der Binominalverteilung und modelliert unwahrscheinliche Ereignisse.

$$f_X(i) = \frac{e^{-\lambda} \cdot \lambda^i}{i!} \tag{19}$$

2.5 Varianz

$$\underbrace{Var(X)}_{\sigma^2} = \mathbb{E}[(X - \underbrace{\mathbb{E}[X]}_{\mu})^2] = \mathbb{E}[X^2] - (\mathbb{E}[X])^2 \tag{20}$$

$$\sigma = \sqrt{Var(x)} \tag{21}$$

The following was presented in the lecture on 04. April 2019.

$$\mathbb{E}[X+Y] = \mathbb{E}[X] + \mathbb{E}[Y] \qquad \text{gilt immer} \tag{22}$$
$$\mathbb{E}[X \cdot Y] = \mathbb{E}[X] \cdot \mathbb{E}[Y] \qquad \text{gilt nur fúr unabhängige X,Y} \tag{23}$$
$$Var[X+Y] = Var[X] + Var[Y] \qquad \text{gilt nur fúr unabhängige X,Y} \tag{24}$$
$$Var[X \cdot Y] \neq Var[X] \cdot Var[Y] \qquad \text{Ungleichheitszeichen gilt} \tag{25}$$

Theorem 2.14

X, Y, Z V unabhängig
$Z \stackrel{\text{def}}{=} X + Y$

$$f_z(z) = \sum_{x \in W_x} f_x(x) f_y(z-x) \tag{26}$$

The following was presented in the lecture on 09. April 2019.

Theorem 2.15: Waldsche Identität

und X seien zwei unabhängige Zufallsvariablen. mit

$$Z \stackrel{\text{def}}{=} \sum_{i=1}^{N} X_i \tag{27}$$

wobei X_1, X_2 unabhängige Kopien von X seiein. Dann gilt:

$$\mathbb{E}[Z] = \mathbb{E}[N] \cdot \mathbb{E}[X] \tag{28}$$

2.6 Abschätzen von Wahrscheinlichkeiten

2.6.1 Markov-Ungleichung

Falls gilt $\forall X \geq 0 \quad \forall t > 0$

$$\Pr[X \geq t] \leq \frac{\mathbb{E}[X]}{t} \tag{29}$$

2.6.2 Chebyshev-Ungleichung

$\forall X \quad \forall t > t$

$$\Pr[|X - \mathbb{E}[X]| \geq t] \leq \frac{\mathrm{Var}[X]}{t^2} \tag{30}$$

2.6.3 Chernoff-Ungleichung

Wichtig: Immer überprüfen, ob man sie anwenden darf. $\forall X$ mit $X = \sum_{i=1}^{n} X_i$ wobei X_i unabhängigen Bernoulli(p_i). $\forall 0 < \delta < 1$

$$\Pr[X \geq (1-\delta) \cdot \mathbb{E}[X]] \leq e^{-\frac{1}{3} \cdot \delta^2 \cdot \mathbb{E}[X]} \tag{31}$$

$$\Pr[X \leq (1-\delta) \cdot \mathbb{E}[X]] \leq e^{-\frac{1}{2} \cdot \delta^2 \cdot \mathbb{E}[X]} \tag{32}$$

The following was presented in the lecture on 11. April 2019.

2.7 Randomisierte Algorithmen

Algorithmus 2.2: Monte Carlo Algorithmen

Laufzeit immer beschränkt
Dürfen Fehler machen

Algorithmus 2.3: Las Vegas Algorithmus

Laufzeit immer beschränkt
Antwort ist entweder korrekt oder unbekannt
Zweite Definition:
Antwort immer korrekt, aber Laufzeit ist potentiell unbeschränkt.

2.8 Fehlerreduktion

Las Vegas Algorithmus mit beschränkter Laufzeit aber mit Fehlerwahrscheinlcihkeit $\epsilon > 0$ beliebig klein.

$$\Pr[A'[I] \; korrekt] \geq 1 - \delta \tag{33}$$

Das kann man erreichen indem man den Algorithmus n mal aufruft und falls es noch keine Antwort hat gibt er unbekannt zurück.

$$n = \frac{1}{\epsilon} \cdot \ln \delta^{-1} \tag{34}$$

Bei Monte Carlo Algorithmen gibt es zwei verschieden Typen, der erste Typ ist mit einseitigem Fehler.

$$n = \frac{1}{\epsilon} \cdot \ln \delta^{-1} \tag{35}$$

und mit zweiseitigen Fehler:
Es muss gelten:

$$\Pr[A(i) \; korrekt] \geq \frac{1}{2} + \epsilon \tag{36}$$

Und man möchte folgendes damit erreichen:

$$\Pr[A'(i)\ korrekt] \geq 1 - \delta \tag{37}$$

Man ruft den Algorithmus N mal auf und gibt die Mehrheit der Antworten aus.

The following was presented in the lecture on 16. April 2019.

Algorithmus 2.4: QuickSelect

```
1 p ← Uniform({l, l+1, ..., r})
2 t ← Partition(A, l, r, p)
3 if k = t then
4     return A[t]
5 else if k < t then
6     return QuickSelect(A,l,t-1,k)
7 else
8     return QuickSelect(A,t+1,r,k--t)
```

Laufzeit: $\mathcal{O}(n)$

Note 8. Um ein Las Vegas Algorithmus daraus zu machen lässt man die Ausführung nach Doppelter Erwarteter Laufzeit ab und gibt unbekannt zurück.

2.9 Target-Shooting

Theorem 2.16

Falls N gross ist so ist die Ausgabe des Target-Shooting mit Wahrscheinlichkeit mindestens $1 - \delta$ im Intervall

$$\left[((1-\epsilon)\frac{|S|}{|U|}, ((1+\epsilon)\frac{|S|}{|U|}\right] \tag{38}$$

2.10 Hashing und Zuordnungsverfahren

eine Hashfunktion bildet ein grosse Datenmenge auf eine kleine Zahl ab.
Die Wahrscheinlichkeit das irgendein Bin mehr als t Bälle enthählt ist höchstens:

$$\Pr[\sum_{i=1}^{m} Y_{i,t} \geq 1] \leq m \cdot 2^{-t} \tag{39}$$

$$\Rightarrow t \geq \log(m) + \log(\frac{1}{\delta}) \tag{40}$$

The following was presented in the lecture on 18. April 2019.

2.11 Minimum Cut

Definition 10. Kantenkontraktion: Ersetze 2 Knoten durch einen und der neue Knoten bekommt alle Kanten der alten Knoten.

Algorithmus 2.5: Kargers Algorithmus

```
CUT(G)
    while|V(G)| > 2 do
        e ← gleichverteilt zufaellige Kante in G
        G ← G/e
    return Groesse des eindeutigen Schnitts in G
```

Findet Min-Cut mit Wahrscheinlichkeit mindestens $\frac{1}{n^2}$
Wenn man $C \cdot n^2$ widerholt findet es die Kanten mit Wahrscheinichkeit $1 - e^{-C}$.
Laufzeit: $\mathcal{O}(C \cdot n^4)$ Laufzeit mit Bootstrapping: $\mathcal{O}(n^2 \cdot \log(n)^3)$

Lemma 2.2

Algorithmus findet Min-Cut, wenn er nur Kanten ausserhalb des Cuts kontrahiert.

2.12 Bootstrapping

Definition 11. Man bricht den ersten Algorithmus ab nachdem die Fehlerwahrscheinlichkeit gross wird. Nachdem wir diesem Algorithmus abgespeichert haben, lassen wir diesen kleineren Graphen mit einem anderen Algorithmus viele Male laufen, welcher weniger Fehler macht.

2.13 Primzahltest

Theorem 2.17: Kleiner Fermatischer Satz

n ist eine Primzahl und $a < n$ dann gilt $a^{n-1} \equiv 1(mod n)$

Carmicheal-Zahlen sind Zahlen bei welcher ein einfacher Primzahltest fehlschlägt.

Algorithmus 2.6: Miller-Rabin-Primzahltest

```
if n = 2:
    return Primzahl
else if n gerade oder n = 1:
    return keine Primzahl
Bestimme a ∈ {2,3,...,n-1} random,
Berechne k,d ∈ ℤ mit n-1 = d·2^k und d ungerade.
x ← a^d mod n
if x = 1 or x = n-1:
    return Primzahl
for i in range(k-1):
    x ← x^2 mod n
```

```
    if x = 1:
        return keine Primzahl
    if x = n − 1:
        return Primzahl
return keine Primzahl
```

The following was presented in the lecture on 30. April 2019.

2.14 Color-Coding

Definition 12. Bunter Pfad ist ein Pfad mit allen Kanten unterschiedlich gefärbt.

Example 1. Gegeben einen mit $k+1$ Farben gefärbten Graphen, gibt es einen bunten Pfad der Länge k?

Algorithmus 2.7

```
Bunte Pfade(G, k)
1. Initialisierung: ∀v ∈ V berechne P_i(v)
2. for i = 2 to k
        ∀v ∈ V :   P_i(v) = ∅
            ∀w ∈ N(v)
                ∀S ∈ P_{i−1}(w)
                    if c(v) ∉ S:
                        P_i(v) = P_i(v) ∪ {S ∪ {c(v)}}
3. if ∃v ∈ V mit P_k(v) ≠ ∅
    return Ja
else
    return Nein
```

Laufzeit: $\mathcal{O}\left(\sum_{i=1}^{k}\sum_{v\in V} deg(v)\cdot i\cdot\binom{k}{i+1}\right) = \mathcal{O}(m\cdot n^c\cdot\log(n))$

The following was presented in the lecture on 02. May 2019.

2.15 Lange Pfade

Um nun ein Langer Pfad zu finden setzt man $k = B+1$, und verwendet den bunte Pfade Algorithmus. Färbe G zufällig mit k Farben, wende Bunte-Pfade-Algorithmus an. Falls die Antwort ja ist, dann gib Ja zurück, sonst Nein.

$$\Pr[\text{Algorithmus färbt } P_0 \text{ nicht bunt}] = 1 - \frac{k!}{k^k} \leq 1 - e^{-k} \tag{41}$$

$$\Pr[\text{Algorithmus gibt richtige Antwort}] \geq \begin{cases} 1 & \text{korrekte Antwort Nein} \\ e^{-k} & \text{korrekte Antwort Ja} \end{cases} \tag{42}$$

2.16 Flüsse in Netzwerken

Definition 13. Netwerk $N = (V, A, s, t, c)$
V = Vertex, A = Arch (gerichtet), s=source, t=target, c=capacity
Flusserhaltung: $f : A \to \mathbb{R}_0^+$

$$\forall v \in V \backslash \{s, t\} : \quad \sum_{(x,v)\in A} f(x, v) = \sum_{(v,x)\in A)} f(v, x) \tag{43}$$

$$val(f) = netoutflow(s) = \sum_{(s,x)\in A} f(s, x) - \sum_{(x,s)\in A} f(x, s) \tag{44}$$

Lemma 2.3

Es gilt:

$$val(f) = netinflow(t) = \sum_{(x,t)\in A} f(x, t) - \sum_{(t,x)\in A} f(t, x) = val(f) \tag{45}$$

The following was presented in the lecture on 07. Mai 2017. Ziel: Gegeben ein gerichter Graph finde maximalen Fluss:

$$val(f) = \max_{\tilde{f} Fluss} val(\tilde{f}) \tag{46}$$

Definition 14. Ein s-t-Schnitt in einem Netzwerk N ist eine Partition $V = S \cup T$ mit $s \in S$ und $t \in T$

$$cap(S, T) \stackrel{\text{def}}{=} \sum_{(u,v)\in S\times T\cap A} c(u, v) \tag{47}$$

Lemma 2.4

$\forall$ s-t-Schnitte(S, T):

$$val(t) \leq cap(S, T) \tag{48}$$

Corollary 2.1

$$\max val(t) \leq \min cap(S, T) \tag{49}$$

Algorithmus 2.8: Ford-Fulkerson

```
1 f ← 0
2 while ∃ s-t-Pfad P in (V,A) do
3    Erhoehe den Fluss entlang P
4 return f
```

$$f'(e) \overset{\text{def}}{=} \begin{cases} f(e)+\epsilon & e \text{ auf } P \\ f(e)-\epsilon & e_{\text{Entgegengesetzt}} \text{ auf } P \\ f(e) & \text{sonst} \end{cases} \tag{50}$$

$$\epsilon = \min r_f(e) \tag{51}$$

The following was presented in the lecture on 09. May 2019.

Theorem 2.18: Maxflow-Mincut Theorem

$$\max val(t) = \min cap(S,T) \tag{52}$$

Definition 15. Restnetwerk: Kanten die im Netwerk noch nicht benutzt sind und ausgeschöpfte Kanten umgedreht.

Theorem 2.19

Ein Fluss ist genau dann maximal, wenn es im Restnetwerk keinen gerichten s-t-Pfad gibt.

Algorithmus 2.9: Edmonds-Karp

```
1 f ← 0
2 while ∃ s-t-Pfad P in (V,A) do
3    Waehle einen kuerzesten solchen Pfad P
4    Erhoehe den Fluss entlang P
5 return f
```

Laufzeit: $\mathcal{O}(n \cdot m^2)$

Note 9. Eine mögliche Anwendung von Flussdiagrammen ist maximaler Fluss $\hat{=}$ maximales Matching
Alle Kanten werden beim Matching von A nach B gerichtet und eine Source und Target angefügt mit Kantengewichten 1.

The following was presented in the lecture on 14. May 2019.

2.17 Flussproblem

Gibt es einen Subgraphen $G' = (A \cup B, E')$ mit $deg_{G'}(a) = k_a$ und $deg_{G'}(b) = k_b$?

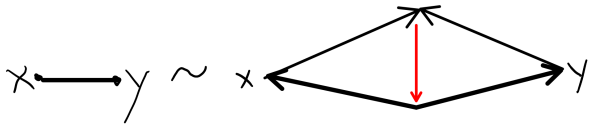
$$G' \text{exisitert} \Leftrightarrow \exists \text{Fluss t in N mit} val(t) = \sum_{a \in A} k_a = \sum_{b \in B} k_b \tag{53}$$

2.18 Intern Kanten-disjunkte Pfade in gerichteten Graphen

Theorem 2.20

Maximaler Fluss = maximale Anzahl kantendisjunkte Pfade

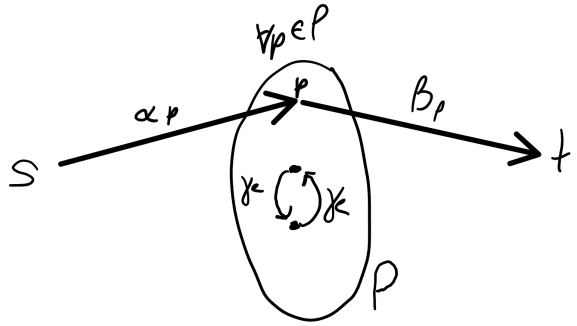
Bei ungerichteten Pfaden:



2.19 Bildsegmentierung

Gesucht Patitionierung von Vordergrund und Hintergrund. Jeder Pixel ist entweder 1 oder 0, ob er im Vordergrund ist oder nicht.

$$\max_{(A,B)} -Q + \sum_{p \in A} \alpha_p + \sum_{p \in B} \beta_p - \sum_{e \in E} \gamma_e = \min_{(A,B)} \sum_{p \in B} \alpha_p + \sum_{p \in A} \beta_p + \sum_{e \in E} \gamma_e \tag{54}$$



$$\min_{(A,B)} cap(\tilde{A}, \tilde{B})\ s-t-Schnitt = \min_{(A,B)} \sum_{p \in B} \alpha_p + \sum_{p \in A} \beta_p + \sum_{e \in E} \gamma_e \tag{55}$$

The following was presented in the lecture on 16. May 2019.

3 Geometrische Algorithmen

3.1 Kleinster umschiessender Kreis

Lemma 3.1

Für jede Punktemenge P gibt es genau einen kleinsten umschliessenden Kreis.

Algorithmus 3.1

```
do forever:
    choose |Q ⊆ P| with |Q| = 12 randomly.
    calculate C(Q)
    if P ⊆ C(Q):
        return C(Q)
    verdopple alle Punkte ausserhalb von C(Q)
```

Erwartete Laufzeit: $\mathcal{O}(n \log(n))$

Lemma 3.2

Gegeben: $n_1, ..., n_t \in \mathbb{N}$

$$N \stackrel{\text{def}}{=} \sum_{i=1}^{t} n^i \tag{56}$$

```
← Uniform({1, ..., N})
x ← 1
while k < Σ_{j=1}^{x} n_j
    x += 1
return x
```

Dann gilt $\Pr[x = i] = \frac{n_i}{N}$

Lemma 3.3

$$\mathbb{E}[\text{Anzahl Punkte ausserhalb von} C(R)] \leq \frac{3}{r+1} \cdot n \tag{57}$$

$$2^{\frac{k}{3}} \leq \mathbb{E}[X_k] \leq \left(1 + \frac{3}{r+1}\right)^k \cdot n \tag{58}$$

3.2 Konvexe Hülle

Wenn man eine Schnur um alle Punkte legt und sie anzieht, erhählt man eine konvexe Hülle.

3.3 Voronoi Diagram

Teile Graph in Flächen auf, sodass wenn man einen neuen Punkt hinzufügt sofort weiss, welcher Punkt der nčhste ist.

3.4 Schöner Graph Zeichnen

$$\text{Planarer Graph} \Leftrightarrow \text{Es gibt eine kreuzungsfreie Darstellung} \tag{59}$$

The following was presented in the lecture on 21. May 2019.

$$\mathbb{E}[T] \leq k_0 + \mathcal{O}(1) \leq 50 \cdot \ln(n) + \mathcal{O}(1) \tag{60}$$

Dies zeigt man, da falls der Algorithmus lange braucht, die Anzahl der Punkte sehr schnell wächst.

3.5 Manhatten Norm

$$||P|| = |P_x| + |P_y| \tag{61}$$

Definition 16. Eine Menge in $\mathbb{R}^2$ ist konvex wenn für beliebige zwei Punkte der Menge auch die Linie dazwischen vollständig enthalten ist.

Definition 17. Sei S eine Menge Punkte in $\mathbb{R}^2$. Dann ist die konvexe Hülle, conv(S), von S der Durchschnitt aller konvexen Mengen die S enthalten sind.

Definition 18. Eine Randkante ist eine Verbindung zwischen zwei Punkten die ganz aussen sind.

Note 10. Der grösste Abstand zweier Punkte kann man finden indem man alle Eckkanten der konvexen Hülle überprüft.

Note 11. Für die kleinste Distanz zwischen einem Punkt ausserhalb der konvexen Menge mit der konvexen Menge ist entweder ein Eckpunkt der konvexen Hülle oder ein Punkt auf einer Randkante

The following was presented in the lecture on 23. May 2019.

3.6 Finden einer konvexen Hülle

Ein ineffizienter Algorithmus ist es in $\mathcal{O}(n^2)$ kann man alle Kanten finden. In $\mathcal{O}(n)$ kann man prüfen, ob es eine Randkante ist. Das gibt eine Laufzeit von $\mathcal{O}(n^3)$
Bestimme mit Hilfe der Randkanten die Eckenfolge der konvexen Hülle in $\mathcal{O}(n)$

Algorithmus 3.2: JarvisWrap Finde konvexe Hülle

```
1 JarvisWrap(P)
2     h ← 0
3     p_now ← Find most left Point
4     do {
5         p_h ← p_now
6         p_now ← FindNext(p_now)
7         h+=1
8     } while(p_now ! = p_0)
9     return (p_0, p_1, ..., p_{h-1})
```

```
10
11 FindNext(p)
12     choose q ∈ P\{p} randomly
13     cand ← q
14     for all r ∈ P\{p, q}
15         if r is right von p.cand:
16             cand ← r
17     return cand
```

Laufzeit $\mathcal{O}(n \cdot h)$

Definition 19. Sei P eine Punktemenge in $\mathbb{R}^c$.

1. Eine Graph $G \in (P, E)$ heisst eben, genau dann wenn sich für alle Kanten $\{p, q\} \in E$ die Segmente pq höchstens in Endpunkten schneiden.
2. Entfernen wir alle Segmente / Kanten $\Rightarrow$ Gebiete
3. Ein Graph $T = (P, E)$ heisst Triangulierung genau dann wenn alle inneren Gebiete Dreiecke sind.

Algorithmus 3.3

```
1 Bestimme einen Graph der alle Knoten enthählt von Links nach Rechts.
2 Solange es Abkürzungen gibt:
3     Falls es eine Abkürzung gibt die unten durch geht, nimm die Abkürzung.
4
5 Mache das Gleiche noch von Rechts nach Links.
```

Lemma 3.4

P ist eine Punktemenge in $\mathbb{R}^2$, und $h \overset{\text{def}}{=}$ # Punkte auf $conv(P)$. Dann gilt der Algorithmus macht genau $2n - 2 - h$ viele Verbesserungsschritte. Der Algorithmus erzeugt immer eine Triangulierung.