

Vorlesung

Effiziente Algorithmen und Komplexitätstheorie

Sommersemester 2008

Ingo Wegener

Lineare Optimierung

LP Minimiere $Z(x) = c^T x$
 unter $Ax \leq b$
 mit $x \in \mathbb{R}_0^+$

Wir haben

- Simplex-Algorithmus (Algorithmus 14.29)
- partiell korrekt
- korrekt für nichtdegenerierte Probleme
- **schlecht** im Worst Case
- empirisch **gut**

Erinnerung

$$\begin{bmatrix} p & q \\ r & s \end{bmatrix} \xrightarrow{\text{Basiswechsel um Pivotelement } p} \begin{bmatrix} \frac{1}{p} & \frac{q}{p} \\ -\frac{r}{p} & s - \frac{rq}{p} \end{bmatrix}$$

Dualität

gesehen

- LP mit n Variablen und m Nebenbedingungen
 ↗ LP mit m Variablen und n Nebenbedingungen
- Koeffizienten durch „Umsortieren“

Definition 14.34

Zu einem **primalem Problem** $Z(x) = c^T x \rightarrow \min$ unter $Ax \geq b$ und $x \geq 0$ heißt das Problem $Z'(y) = b^T y \rightarrow \max$ unter $A^T y \leq c$ und $y \geq 0$ das **duale Problem**.

Theorem 14.35

Sei P ein primales Problem der linearen Optimierung, sei P' das zu P duale Problem, sei P'' das zu P' duale Problem. Dann sind P und P'' äquivalent.

Beweis von Theorem 14.35

Wir haben P gegeben durch $Z(x) = c^T x \rightarrow \min$
unter $Ax \geq b$ und $x \geq 0$

Wir haben P' gegeben durch $Z'(y) = b^T y \rightarrow \max$
unter $A^T y \leq c$ und $y \geq 0$

äquivalent zu $-Z'(y) = (-b)^T y \rightarrow \min$
unter $(-A)^T y \geq -c$ und $y \geq 0$

dazu dual P'' gegeben durch $Z''(z) = (-c)^T z \rightarrow \max$
unter $((-A)^T)^T z \leq -b$ und $z \geq 0$

äquivalent zu $-Z''(z) = c^T z \rightarrow \min$
unter $Az \geq b$ und $z \geq 0$

offenbar äquivalent zu P



Beziehung zwischen Zielfunktionswerten

Theorem 14.36

Sei x zulässig für das lineare Minimierungsproblem P mit Zielfunktion Z , sei y zulässig für das duale Maximierungsproblem P' mit Zielfunktion Z' . Dann gilt $Z(x) \geq Z'(y)$.

Beweis.

klar $Z(x) = c^T x$ (Definition Zielfunktion)

klar $c^T x \geq (A^T y)^T x$ (Nebenbedingung P')

klar $(A^T y)^T x = y^T Ax$ (elementares Rechnen)

klar $y^T Ax \geq y^T b$ (Nebenbedingung P)

klar $Z'(y) = y^T b$ (Definition Zielfunktion)

zusammen $Z(x) \geq Z'(y)$



Einmal in Ruhe hinsehen...

Theorem 14.36

Sei x zulässig für das lineare Minimierungsproblem P mit Zielfunktion Z , sei y zulässig für das duale Maximierungsproblem P' mit Zielfunktion Z' . Dann gilt $Z(x) \geq Z'(y)$.

Beobachtung zulässiger Punkt y für P'
definiert untere Schranke für Z

also $M_{P'} \neq \emptyset \Rightarrow \inf_{x \in M_P} Z(x) > -\infty$

anders gewendet $Z \rightarrow -\infty \Rightarrow M_{P'} = \emptyset$

Das Dualitätstheorem

Theorem 14.37 (Dualitätstheorem)

Sei P ein primales Problem der linearen Optimierung mit Zielfunktion Z , sei P' das zu P duale Problem mit Zielfunktion Z' . Es gilt:

- ① P hat eine Lösung $\Leftrightarrow P'$ hat eine Lösung
- ② x Lösung von P und y Lösung von $P' \Rightarrow Z(x) = Z'(y)$
- ③ Aus dem Lösungssimplextableau von P kann die Lösung von P' abgelesen werden und umgekehrt.

Beweisstrategie

- Betrachte Simplextableaus von P und P' parallel.
- Führe Simplex-Algorithmus für P aus.
- Vollziehe jeden Basiswechsel parallel auch für P' .
- Beobachte die Beziehungen zwischen den beiden Simplextableaus.

Zum Beweis des Dualitätstheorems

Betrachte Basiswechsel um $-a_{i,k}$

$$\begin{aligned}
 a''_{i,k} &= \frac{1}{a_{i,k}} & a''_{l,j} &= a_{l,j} - a_{i,j} \frac{a_{l,k}}{a_{i,k}} \quad (\neq i, k) \\
 a''_{i,j} &= -\frac{a_{i,j}}{a_{i,k}} \quad (\neq k) & b''_l &= -b_l + b_i \frac{a_{l,k}}{a_{i,k}} \quad (\neq i) \\
 b''_i &= \frac{b_i}{a_{i,k}} & c''_j &= -c_j + a_{i,j} \frac{c_k}{a_{i,k}} \quad (\neq k) \\
 a''_{l,k} &= \frac{a_{l,k}}{a_{i,k}} \quad (\neq i) & \Delta'' &= \Delta - b_i \frac{c_k}{a_{i,k}} \\
 c''_k &= -\frac{c_k}{a_{i,k}}
 \end{aligned}$$

$$\begin{aligned}
 a'_{i,k} &= -\frac{1}{a_{i,k}} \\
 a'_{i,j} &= \frac{a_{i,j}}{a_{i,k}} \quad (\neq k) \\
 b'_i &= -\frac{b_i}{a_{i,k}} \\
 a'_{l,k} &= -\frac{a_{l,k}}{a_{i,k}} \quad (\neq i) \\
 c'_k &= \frac{c_k}{a_{i,k}} \\
 a'_{l,j} &= -a_{l,j} + a_{l,k} \frac{a_{i,j}}{a_{i,k}} \quad (\neq i) \\
 b'_l &= b_l - a_{l,k} \frac{b_i}{a_{i,k}} \quad (\neq i) \\
 c'_j &= c_j - c_k \frac{a_{i,j}}{a_{i,k}} \quad (\neq k) \\
 \delta' &= \delta + c_k \frac{b_i}{a_{i,k}}
 \end{aligned}$$

y_1	\dots	y_i	\dots	y_m		
$a_{1,1}$	\dots	$a_{i,1}$	\dots	$a_{m,1}$	$-c_1$	$-v_1$
\vdots	\dots	\vdots	\dots	\vdots	\vdots	\vdots
$a_{1,k}$	\dots	$a_{i,k}$	\dots	$a_{m,k}$	$-c_k$	$-v_k$
\vdots	\dots	\vdots	\dots	\vdots	\vdots	\vdots
$a_{1,n}$	\dots	$a_{i,n}$	\dots	$a_{m,n}$	$-c_n$	$-v_n$
$-b_1$	\dots	$-b_i$	\dots	$-b_m$	$\Delta = 0$	$-Z'$

$$\begin{bmatrix} p & q \\ r & s \end{bmatrix} \rightsquigarrow \begin{bmatrix} \frac{1}{p} & \frac{q}{p} \\ -\frac{r}{p} & s - \frac{rq}{p} \end{bmatrix}$$

Beweis des Dualitätstheorems

Beobachtung — Es bleiben erhalten

- Beziehungen zwischen BV und N-BV
- $A'' = -A'^T$
- letzte Spalte in $A' = -$ letzte Zeile in A''
- letzte Zeile in $A' = -$ letzte Spalte in A''
- $\Delta'' = -\delta$

Das gilt für alle Basiswechsel.

Simplex-Algo. für P hält mit opt. Basispunkt $x^* = 0, u^* = -b^*$.

- (1) Basispunkt zulässig $\Rightarrow -b^* \geq 0$
- (2) Basispunkt optimal $\Rightarrow c^* \geq 0$
- (3) Wert δ^*

Was gilt für Basispunkt $y^* = 0, v^* = c^*$ für P' ?

- (1) $c^* \geq 0 \Rightarrow$ zulässig
- (2) $-b^* \geq 0 \Rightarrow$ optimal
- (3) Wert $-\Delta^* = \delta^*$



Folgerung aus dem Dualitätstheorem

Korollar 14.38

Sei P ein lineares Optimierungsproblem, sei P' sein duales Problem. Es gilt stets eine der vier folgenden Aussagen.

- ① P und P' haben beide Lösungen, die Lösungen haben den gleichen Wert.
- ② Z ist auf M_P nach unten unbeschränkt und $M_{P'} = \emptyset$.
- ③ Z' ist auf $M_{P'}$ nach unten unbeschränkt und $M_P = \emptyset$.
- ④ $M_P = \emptyset$ und $M_{P'} = \emptyset$.

Anwendung

Beobachtung erster Basispunkt zu P **unzulässig** und $c \geq 0$
 \Rightarrow erster Basispunkt zu P' **zulässig**

also Lösen von P' **spart** erste Phase des Simplex-Algorithmus

Auf dem Weg zu Approximationsalgorithmen

Zusammenfassung lineare Optimierung

primales Problem P

minimiere $c^T x$ unter

$$Ax \geq b$$

$$x \geq 0$$

duales Problem D

maximiere $y^T b$ unter

$$y^T A \leq c$$

$$y \geq 0$$

Theorem 14.36: x, y zulässig $\Rightarrow c^T x \geq y^T b$

Theorem 14.37: primal hat Lösung \Leftrightarrow dual hat Lösung
optimale Lösungen haben gleichen Wert

jetzt Entwicklung eines **Approximationsalgorithmus**
an einem **konkreten Beispiel**

Vertex Cover

Problem gewichtetes Vertex Cover

Eingabe ungerichteter gewichteter Graph $G = (V, E, c)$
Knotengewichte $c: V \rightarrow \mathbb{N}$

Ausgabe $V' \subseteq V$ mit $\forall e \in E: e \cap V' \neq \emptyset$ und $\sum_{v \in V'} c(v)$ minimal

Fakt NP-schwierig

Beobachtung leicht als ganzzahliges LP formulierbar

Minimiere $\sum_{v \in V} c(v) \cdot x_v$ unter

$\forall e \in E: \sum_{v \in e} x_v \geq 1$

$\forall v \in V: x_v \in \{0, 1\}$

Ganzzahliges LP als Ausgangspunkt

Formulierung als ganzzahliges LP

Minimiere $\sum_{v \in V} c(v) \cdot x_v$ unter

$$\forall e \in E: \sum_{v \in e} x_v \geq 1$$

$$\forall v \in V: x_v \in \{0, 1\}$$

Relaxierung P

Minimiere $\sum_{v \in V} c(v) \cdot x_v$ unter

$$\forall e \in E: \sum_{v \in e} x_v \geq 1$$

$$\forall v \in V: x_v \geq 0$$

Erinnerung randomisiertes Runden

Duales Problem D dazu

Maximiere $\sum_{e \in E} y_e$ unter

$$\forall v \in V: \sum_{e=\{u,v\} \in E} y_e \leq c(v)$$

$$\forall e \in E: y_e \geq 0$$

Duales Problem und Approximation

Sei y zulässige Lösung von D . $\text{Wert}(y) = \sum_{e \in E} y_e$

klar $\sum_{e \in E} y_e \leq \underbrace{\text{OPT}_P}_{\text{Theorem 14.36}} \leq \underbrace{\text{OPT}_{VC}}_{\text{Relaxierung}}$

betrachte Vertex Cover V' mit $\sum_{v \in V'} c(v) \leq \rho \cdot \text{Wert}(y)$

$$\sum_{v \in V'} c(v) \leq \rho \cdot \text{Wert}(y) \leq \rho \cdot \text{OPT}_P \leq \rho \cdot \text{OPT}_{VC}$$

also $\frac{\sum_{v \in V'} c(v)}{\text{OPT}_{VC}} \leq \rho$ V' ist ρ -Approximation

„Kochrezept“ für Approximation

- 1 Formuliere Relaxierung P und duales Problem D .
- 2 Starte mit Lösungen $x = 0$ und $y = 0$.
Beobachtung $y = 0$ ist zulässig. $x = 0$ ist meist unzulässig.
- 3 Solange primale Lösung x unzulässig
 - 1 Erhöhe ein y_i , bis eine duale Bedingung exakt erfüllt
 $(\sum_i y_i a_{i,j} = c_j)$, dabei y weiter zulässig
 - 2 Wähle solche dualen Bedingungen und erhöhe entsprechende primale Variable ganzzahlig.
- 4 In der Analyse, beweise $c^T x \leq \rho \cdot y^T b$ für ein möglichst kleines ρ .

Algorithmus für Vertex Cover (Algorithmus 14.39)

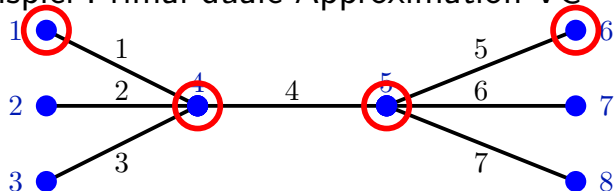
Primal-dualer Approximationsalgorithmus für Vertex Cover

Eingabe ungerichteter Graph $G = (V, E)$, Knotenkosten $c: V \rightarrow \mathbb{N}$

Ausgabe Vertex Cover M

1. $x := 0, y := 0$
2. Wenn $E \neq \emptyset$
3. Wähle $E' \subseteq E$ beliebig.
4. Erhöhe y_e für alle $e \in E'$ gleichförmig bis $\sum_{e=\{u,v\} \in E} y_e = c(v)$ für ein $v \in V$.
5. $S := \left\{ v \in V \mid \sum_{e=\{u,v\} \in E} y_e = c(v) \right\}$
6. Für alle $v \in S$
7. $x_v := 1$
8. $E := E \setminus \{\{u, v\} \in E\}$
9. $M := \{v \in V \mid x_v = 1\}$

Beispiel Primal-duale Approximation VC



primales Problem

Min. $\sum_{i=1}^8 x_i$ unter

$$x_1 + x_4 \geq 1$$

$$x_2 + x_4 \geq 1$$

$$x_3 + x_4 \geq 1$$

$$x_4 + x_5 \geq 1$$

$$x_5 + x_6 \geq 1$$

$$x_5 + x_7 \geq 1$$

$$x_5 + x_8 \geq 1$$

$$x_1, \dots, x_8 \geq 0$$

duales Problem

Max. $\sum_{i=1}^7 y_i$ unter

$$y_1 \leq 1$$

$$y_2 \leq 1$$

$$y_3 \leq 1$$

$$y_1 + y_2 + y_3 + y_4 \leq 1$$

$$y_4 + y_5 + y_6 + y_7 \leq 1$$

$$y_5 \leq 1$$

$$y_6 \leq 1$$

$$y_7 \leq 1$$

$$y_1, \dots, y_7 \geq 0$$

$$E = \{1, \dots, 7\}$$

$$E' = \{1\}$$

$$y_1 = 1, S = \{1, 4\}$$

$$x_1 = 1, x_4 = 1$$

$$E = \{5, 6, 7\}$$

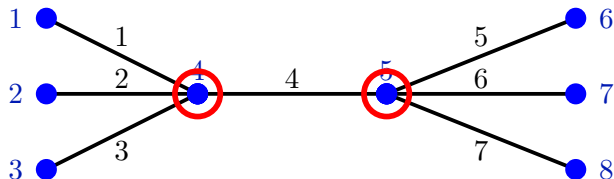
$$E' = \{5\}$$

$$y_5 = 1, S = \{5, 6\}$$

$$x_5 = 1, x_6 = 1$$

$$E = \emptyset$$

Beispiel Primal-duale Approximation VC



primales Problem

Min. $\sum_{i=1}^8 x_i$ unter

$$x_1 + x_4 \geq 1$$

$$x_2 + x_4 \geq 1$$

$$x_3 + x_4 \geq 1$$

$$x_4 + x_5 \geq 1$$

$$x_5 + x_6 \geq 1$$

$$x_5 + x_7 \geq 1$$

$$x_5 + x_8 \geq 1$$

$$x_1, \dots, x_8 \geq 0$$

duales Problem

Max. $\sum_{i=1}^7 y_i$ unter

$$y_1 \leq 1$$

$$y_2 \leq 1$$

$$y_3 \leq 1$$

$$y_1 + y_2 + y_3 + y_4 \leq 1$$

$$y_4 + y_5 + y_6 + y_7 \leq 1$$

$$y_5 \leq 1$$

$$y_6 \leq 1$$

$$y_7 \leq 1$$

$$y_1, \dots, y_7 \geq 0$$

$$E = \{1, \dots, 7\}$$

$$E' = \{1, 4, 7\}$$

$$y_1 = y_4 = y_7 = \frac{1}{2}$$

$$S = \{4, 5\}$$

$$x_4 = 1, x_5 = 1$$

$$E = \emptyset$$

Über die primal-dualen Approximation von VC

zur Zulässigkeit von x und y

klar initial y zulässig, x unzulässig

nötig am Ende x zulässig

Lemma 14.40

Algorithmus 14.39 berechnet x und y so, dass x zulässig ist für das lineare Programm P (Relaxierung von Vertex Cover) und y zulässig ist für dessen duales Problem P' .

Beweis von Lemma 14.40

Beobachtung Kante $\{u, v\}$ wird entfernt
 $\Leftrightarrow x_u = 1$ oder $x_v = 1$

Beobachtung Algorithmus terminiert
 $\Leftrightarrow E = \emptyset$

also x am Ende zulässig für P

Beobachtung y initial zulässig für P'

Beobachtung y_e explizit nur bis Grenze Zulässigkeit erhöht
außerdem bei Erreichen der Grenze wird e entfernt

also y stets zulässig für P'



Über Algorithmus 14.39

Theorem 14.41

Der primal-duale Approximationsalgorithmus für Vertex Cover ist eine 2-Approximation.

Beweis.

Beobachtung polynomielle Laufzeit ✓

aus Lemma 14.40 M ist Vertex Cover ✓

zu zeigen Approximationsgüte

Betrachte M und zugehöriges x am Ende des Algorithmus

laut Algorithmus
$$\sum_{v \in M} c(v) = \sum_{v \in M} \left(\sum_{e=\{u,v\}} y_e \right)$$

Beobachtung andere Summationsreihenfolge

$$\rightsquigarrow \sum_{v \in M} c(v) = \sum_{e \in E} \left(\left(\sum_{v \in M \cap e} 1 \right) \cdot y_e \right)$$

Beweis der Approximationsgüte

Wir haben

$$\begin{aligned} \sum_{v \in M} c(v) &= \sum_{v \in M} \left(\sum_{e=\{u,v\}} y_e \right) \\ &= \sum_{e \in E} \left(\left(\sum_{v \in M \cap e} 1 \right) \cdot y_e \right) \end{aligned}$$

klar $\forall e \in E: |e| = 2$

also
$$\sum_{v \in M} c(v) \leq 2 \cdot \sum_{e \in E} y_e$$

also wie gesehen
2-Approximation



Ein Thema aus DAP 2...

Erinnerung Wörterbuchproblem

Aufgabe Verwalte **Schlüssel** (mit Daten).
Operationen Insert, Search, Delete

Erinnerung Schlüssel können geordnet und ungeordnet sein

Erinnerung „Lösungen“ z. B.

- lineare Listen (**einfach**, **langsam**, deterministisch)
- AVL-Bäume (**schnell**, **kompliziert**, deterministisch,
nur für geordnete Schlüsseluniversen)
- Skip-Listen (**schnell**, **eher unkompliziert**, randomisiert,
nur für geordnete Schlüsseluniversen)
- Hashing (**schnell** im Average Case, **langsam** im Worst Case,
einfach, randomisiert)

Hashing

Erinnerung an DAP 2

- **Wörterbuch**, Operationen Search, Insert, Delete (Delete u. U. **problematisch**)
- Daten (eigentlich Schlüssel) aus **Universum** \mathcal{U} ($S \subseteq \{\mathcal{U}\}$, $|S| = n$)
- **Hashfunktion** $h: \mathcal{U} \rightarrow \{0, 1, \dots, M - 1\}$ ($|\mathcal{U}| \gg M$)
- **Kollision** $x \neq y \in \mathcal{U}$ mit $h(x) = h(y)$
- Geburtstagsparadoxon **Kollisionen unvermeidlich**
- **offenes Hashing** Liste an jeder Arrayposition
- **geschlossenes Hashing** im Kollisionsfall andere Arrayposition suchen
- **Kollisionsstrategien** bei geschlossenem Hashing: lineares Sondieren, quadratisches Sondieren, multiplikatives Sondieren, doppeltes Hashing
- zentrale Kenngröße **Lastfaktor** $\alpha = n/M$

Performanz von Hashing

im Durchschnitt **sehr gut**

aber im Worst Case **sehr schlecht**

Erinnerung **Warum ist das so?** (hier für offenes Hashing)

Notation

- $B_i = \{x \in S \mid h(x) = i\}$
- $b_i = |B_i|$

im Worst Case $\forall x, y \in S: h(x) = h(y)$

\rightsquigarrow Zeit $\Theta(n)$ bei offenem Hashing

im Average Case **Uniformitätsannahme**

Die Hashwerte $h(x)$ ($x \in S$) sind rein zufällig.

klar Uniformitätsannahme gerechtfertigt bei zufälligen Daten

Erwartete Suchzeit bei offenem Hashing

erwarte Zeit für **erfolglose** Suche bei offenem Hashing
 $= \Theta(1 + E(b_i))$

Uniformitätsannahme $\Rightarrow E(b_i) = \frac{n}{M} = \alpha$

also erwartete Zeit für erfolglose Suche $\Theta(1 + \alpha)$

erwarte Zeit für **erfolgreiche** Suche bei offenem Hashing

Annahme nach jedem Schlüssel $x \in S$ mit gleicher W'keit suchen

$$\begin{aligned} \text{erwartete Zeit} &= \Theta(1 + E(\text{Elemente} \in B_{h(x)} \text{ vor } x)) \\ &= \Theta(1 + E(\text{nach } x \text{ eingefügte Elemente} \in B_{h(x)})) \end{aligned}$$

Erwartete Zeit erfolgreiche Suche

Definition $X_{i,j} = \begin{cases} 1 & \text{falls } h(x_i) = h(x_j) \\ 0 & \text{sonst} \end{cases}$

Uniformitätsannahme $\Rightarrow \forall i \neq j: \text{Prob}(X_{i,j} = 1) = \frac{1}{M}$

also E (nach x eingefügte Elemente $\in B_{h(x)}$)

$$\begin{aligned}
 &= E \left(\sum_{i=1}^n \frac{1}{n} \cdot \sum_{j=i+1}^n X_{i,j} \right) \\
 &= \frac{1}{n} \sum_{i=1}^n \sum_{j=i+1}^n E(X_{i,j}) = \frac{1}{n} \sum_{i=1}^n \sum_{j=i+1}^n \frac{1}{M} \\
 &= \frac{1}{Mn} \sum_{i=1}^n (n - i + 1) = \frac{1}{Mn} \sum_{i=1}^n i = \frac{n(n+1)}{2Mn} = \frac{n+1}{2M} = \Theta(\alpha)
 \end{aligned}$$

also erwartete Suchzeit $\Theta(1 + \alpha)$

Wappnen gegen den Worst Case

klar mit $n = O(M)$ (also $\alpha = O(1)$) Zeit $\Theta(1) \rightsquigarrow$ sehr anziehend
(vgl. Zeit $\Theta(\log n)$ für AVL-Bäume)

Wunsch Vermeidung des Worst Case

Idee „Verschiebung“ des Zufalls

Idee Wähle Hashfunktion h zufällig unabhängig von S .

„Konkrete Daten können eine uns unbekannte Struktur haben. Diese Struktur kann dazu führen, dass es sehr viele Primärkollisionen gibt. Ein Ausweg aus diesem Dilemma besteht darin, eine Klasse von Hashfunktionen zu betrachten und für jede Anwendung eine Hashfunktion zufällig aus der Klasse zu wählen. [...] Wir wollen diesen Aspekt hier nicht vertiefen.“ (DAP 2-Skript)

Zufällige Hashfunktionen

Annahme \mathcal{U} endlich (also S endlich)

triviale Idee Wähle Hashfunktion aus $\{h: \mathcal{U} \rightarrow \{0, 1, \dots, M-1\}\}$ zufällig gleichverteilt.

Problem speichern von $|\mathcal{U}|$ erforderlich \rightsquigarrow **nicht praktikabel**

Annahme $\mathcal{U} = \{0, 1, \dots, u-1\}$

Definition 15.2

Für $p \geq |\mathcal{U}|$ prim ist $\mathcal{H}_l = \{h_{a,b} \mid 0 < a < p, 0 \leq b < p\}$ eine **Klasse von Hashfunktionen** mit $h_{a,b}(x) := ((a \cdot x + b) \bmod p) \bmod m$.

Wozu Klassen von Hashfunktionen?

erwartete Suchzeit für $h \in \mathcal{H}$ zufällig gleichverteilt gewählt?

klar weiterhin $= 1 + \mathbb{E}(b_{h(x)}) = 1 + \sum_{y \in S \setminus \{x\}} \text{Prob}(h(y) = h(x))$

zentraler Unterschied **vorhin** W'keit über Wahl von y
jetzt W'keit über Wahl von h

Was wünschen wir uns?

Definition 15.1

Eine Hashklasse $\mathcal{H} \subseteq \{h: \mathcal{U} \rightarrow \{0, 1, \dots, M-1\}\}$ heißt **c -universell**, wenn bei zufälliger Wahl von $h \in \mathcal{H}$ für beliebige Schlüssel $x \neq y \in \mathcal{U}$ gilt:

$$\text{Prob}(h(x) = h(y)) \leq \frac{c}{M}$$

Suchzeiten bei c -universellen Hashklassen

Theorem 15.4

Die erwartete Suchzeit bei zufälliger Wahl von h aus einer c -universellen Hashklasse \mathcal{H} beträgt bei erfolgreicher und erfolgloser Suche $\Theta(1 + c \cdot \alpha)$.

Beweis.

Rechnung wie eben mit W'keit c/M statt $1/M$
 $\rightsquigarrow \Theta(1 + c \cdot \alpha)$ □

Gibt es überhaupt c -universelle Hashklassen mit kleinem c ?

Theorem 15.3

\mathcal{H}_l ist 1-universell.

Beweis des Theorems

Seien $x \neq y \in \mathcal{U} = \{0, 1, \dots, u-1\}$ beliebig.

Seien $a \in \{1, \dots, p-1\}$, $b \in \{0, \dots, p-1\}$ rein zufällig.

Beobachtung p prim $\rightsquigarrow \{1, \dots, p-1\} = \mathbb{Z}_p^*$

also $x' := (ax + b) \bmod p$ und $y' := (ay + b) \bmod p$ verschieden
(chinesischer Restklassensatz)

also $h(x) = h(y) \Leftrightarrow x' \bmod M \equiv y' \bmod M$ mit $x' \neq y'$

klar $\forall i \in \mathcal{U}: |\{j \in \mathcal{U} \mid i \equiv j \bmod M\}| \leq \left\lceil \frac{|\mathcal{U}|}{M} \right\rceil$

also $\text{Prob}(h(x) = h(y)) \leq \frac{\lceil |\mathcal{U}|/M \rceil - 1}{|\mathcal{U}| - 1} \leq \frac{(|\mathcal{U}| - 1)/M}{|\mathcal{U}| - 1} = \frac{1}{M}$ □

Zur Implementierung

Einwand alle Überlegungen setzen voraus, dass $|S| = n$ vorab bekannt

Idee dynamische Anpassung der Hashtabellengröße

- Fixiere vorab gewünschten Lastfaktor α' .
- Führe Buch über aktuellen Lastfaktor α .
- Wenn $\alpha > \alpha'$ gilt,
 - Erzeuge neue Hashtabelle doppelter Größe.
 - Wähle zufällig neue Hashfunktion.
 - Trage alle Einträge aus alter Tabelle in neue Tabelle gemäß neuer Hashfunktion. (**Bemerkung** vorheriges Search unnötig)
 - Lösche alte Hashtabelle.

Beobachtungen bei $\leq n$ Einträgen zu jeder Zeit

- Gesamtplatzbedarf $O(n)$
- Gesamtzeitbedarf fürs Umkopieren $O(n)$
- erwarteter Gesamtzeitbedarf für $k \geq n$ Operationen $O(k)$

Perfektes Hashing

bis jetzt alle Aussagen „im Erwartungswert“

Kann man bessere Garantien für die Performanz bekommen?

perfektes Hashing Worst-Case Zugriffszeit $O(1)$

Anmerkung Worst-Case-Zeit $O(1)$ je Operation verschieden von
amortisierter Worst-Case-Zeit $O(1)$ je Operation

hier jetzt Worst-Case-Zeit $O(1)$ je Operation
für statisches Wörterbuch (z. B. CD-ROM)

Auf dem Weg zum statisch perfekten Hashing

Wie erreichen wir Worst-Case-Zeit $O(1)$ je Search?

triviale Beobachtung wenn unter h kollisionsfrei,
dann sicher Zeit $O(1)$

Wie erreichen wir Kollisionsfreiheit?

triviale Beobachtung Wähle M groß genug.

Lemma 15.5

Beim Hashen von n Schlüsseln in eine Hashtabelle der Größe $M := n^2$ mit einer uniform zufällig gewählten Hashfunktion $h \in \mathcal{H}_l$ gibt es mit Wahrscheinlichkeit $\geq 1/2$ keine Kollision.

Beweis von Lemma 15.5

klar Kollision $\Leftrightarrow h(x) = h(y)$ für $x \neq y \in S$

klar Anzahl solcher Paare = $\binom{n}{2}$

also für jedes Paar Kollisionsw'keit $1/M$ (Theorem 15.3)

also erw. Anzahl Kollisionen $E(K) = \binom{n}{2} \cdot \frac{1}{M}$
 $= \frac{n(n-1)}{2n^2} < \frac{1}{2}$

klar $K \geq 0$ also Markow-Ungleichung anwendbar

damit $\text{Prob}(K \geq 2 \cdot 1/2) = \text{Prob}(K \geq 1) < 1/2$

also $\text{Prob}(\text{keine Kollision}) = 1 - \text{Prob}(K \geq 1) > 1/2$ □

Lemma 15.5 kritisch hinterfragt

klar quadratischer Platz **völlig inakzeptabel**

Geht es nicht auch mit $M \ll n^2$?

Beobachtung Prob (keine Kollision) = $\left(1 - \frac{1}{M}\right)^{\binom{n}{2}}$
 $= \left(1 - \frac{1}{M}\right)^{M \cdot \binom{n}{2}/M} \leq e^{-\binom{n}{2}/M} = e^{-\Theta(n^2/M)}$

Fazit Hashtabellengrößen $M = o(n^2 / \log n)$ **chancenlos**

also bessere **Idee** benötigt