In-class exercise
Greedy-Lösung für das Rucksack-Problem

Gegeben sind n Gegenstände mit jeweils ganzzahligem Gewicht $\mathrm{w}_1, w_2, \ldots, w_n$ und einem Wert $v_1, \ldots, v_n$ sowie einen Rucksack mit einem maximalen Tragevermögen von W.

Finde eine Auswahl $S \subset \{1, \ldots, n\}$ der Gegenstände, so dass

- Der Gesamtwert $\sum\limits_{i \in S} v_i$ maximal ist
- Das Gesamtgewicht vom Rucksack getragen werden kann, also $\sum\limits_{i \in S} w_i \leq W$

Greedy-Lösung für das Rucksack-Problem

Nehme an, dass sortiert nach *Wert-Dichte*:

$$\mathrm{q}_i := \frac{v_i}{w_i}$$

Stop and Think

Gegeben, dass man auch *Bruchteile* von Objekten mitnehmen kann, was ist die optimale Lösung?

Optimale Lösung

Sei k maximal, so dass $\underbrace{\sum_{i=0}^{k} w_i}_{=:\rho} \leq W$

Dann ist $\sum_{i=0}^{k} v_i + \frac{W-\rho}{w_{k+1}} v_{k+1}$ optimal.

Approximations-Algorithmus

A ist ein α-Approximations-Algorithmus, falls gilt

$$\forall \sigma : \frac{C_{\mathrm{OPT}}}{C_A} \leq \alpha$$

Greedy-Algorithmus A

Betrachte alle Objekte in absteigend sortierter Reihenfolge und wähle ein Objekt, falls es noch in den Rucksack passt.

Stop and Think

- Greedy-Algorithmus A ist beliebig schlecht
- Gebe eine Instanz an, bei der Greedy A um einen Faktor 700 schlechter als das Optimum ist

Greedy-Algorithmus A

Betrachte alle Objekte in absteigend sortierter Reihenfolge und wähle ein Objekt, falls es noch in den Rucksack passt.

$$\mathrm{v}_1 = 1, w_1 = 1$$
$$v_2 = 700, w_2 = 1000$$
$$\text{und } W = w_2$$

$$q_1 = 1, q_2 = \frac{7}{10}$$

Greedy-Algorithmus B

$$\text{Sei } k \text{ maximal, so dass } \sum_{i=0}^{k} w_i \leq W$$

$$\text{Nehme } \max\left\{v_{k+1}, \sum_{i=0}^{k} v_i\right\}$$

Stop and Think

Greedy-Algorithmus B hat Approximationsrate ≤ 2

$$\sum_{i=0}^{k} v_i + \underbrace{\frac{W - \rho}{w_{k+1}}}_{\leq 1} v_{k+1} \geq \mathrm{OPT}$$

$$\sum_{i=0}^{k} v_i + v_{k+1} \geq \mathrm{OPT}$$

$$\sum_{i=0}^{k} v_i \geq \frac{1}{2}\mathrm{OPT} \text{ oder } v_{k+1} \geq \frac{1}{2}\mathrm{OPT}$$

Stop and Think

Gebe ein Beispiel einer Instanz bei der die Approximationsrate von 2 beliebig nahe angenähert werden kann

Greedy-Algorithmus B

$$\text{Sei } k \text{ maximal, so dass } \sum_{i=0}^{k} w_i \leq W$$

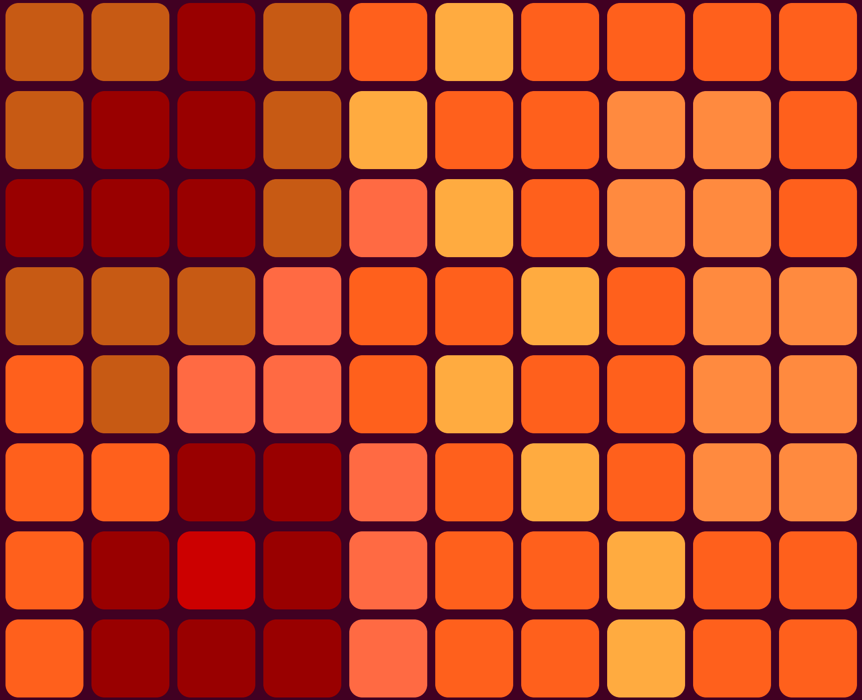
$$\text{Nehme } \max\left\{v_{k+1}, \sum_{i=0}^{k} v_i\right\}$$

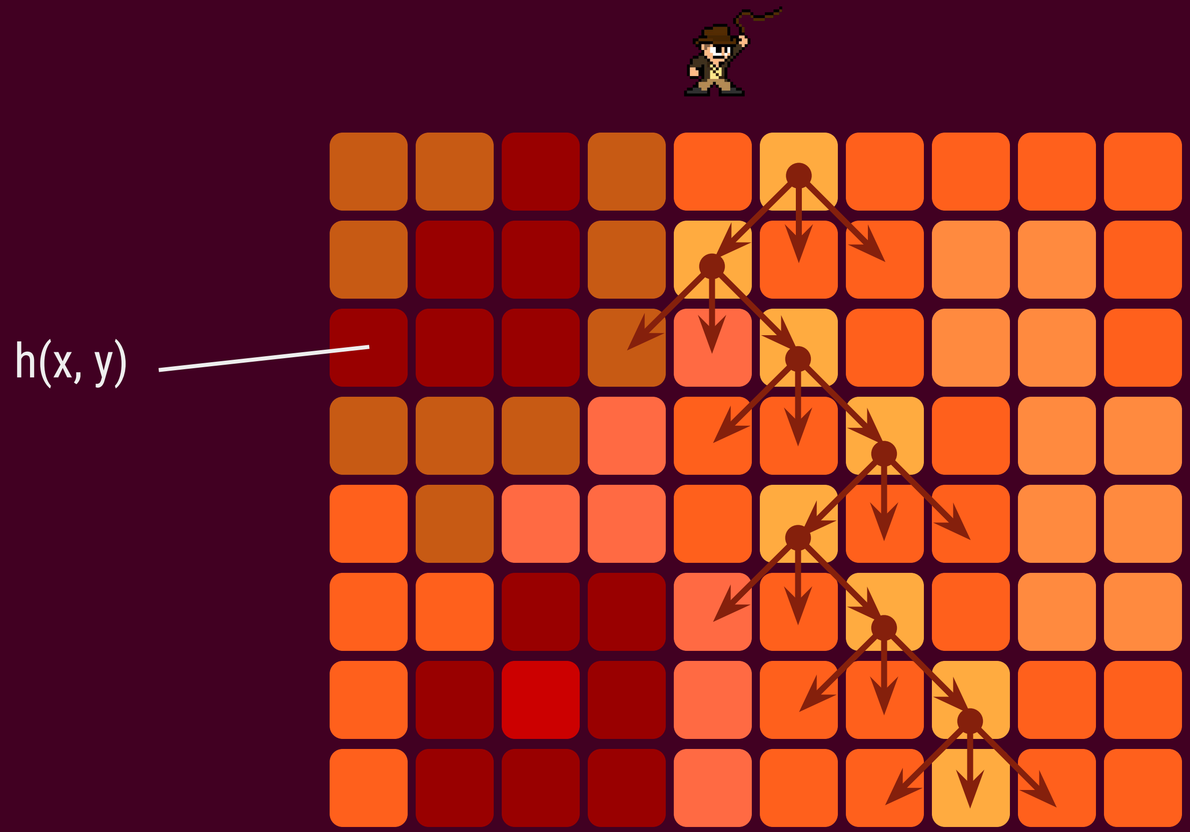
Greedy-Algorithmus B

$$v_1 = 1 + \varepsilon, w_1 = 1 + \varepsilon$$
$$v_2 = 1 + \varepsilon, w_2 = 1 + \varepsilon$$
$$v_3 = 2 - \varepsilon, w_3 = 2$$
$$W = 2$$

$$C_B = 1 + \varepsilon$$
$$C_{\mathrm{OPT}} = 2 - \varepsilon$$

Indiana Jones




h(x, y)

Indiana Jones

- Gegeben Hitze der Steinplatten - $h(x, y)$
- Finde Weg durch Feld mit minimaler Hitzesumme
- In jedem Schritt: ↙, ↓ , ↘

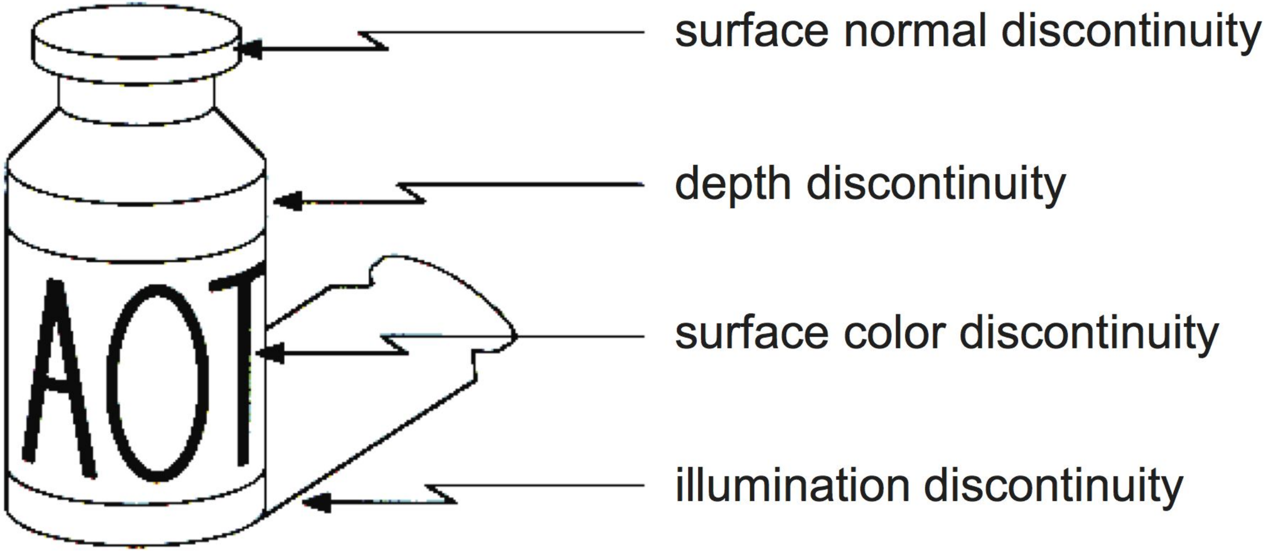
Content Aware Resizing

Content Aware Resizing

- Cropping entfernt wichtige Bereich
- Rescaling verzerrt
- Möchten ”unwichtige” Pixel entfernen
- Was zeichnet einen Pixel als wichtig aus? Was sind wichtige Bereiche im Bild?
  - Viele Möglichkeiten
  - z.B. Kanten sind wichtig, möchten Struktur bewahren

Content Aware Resizing

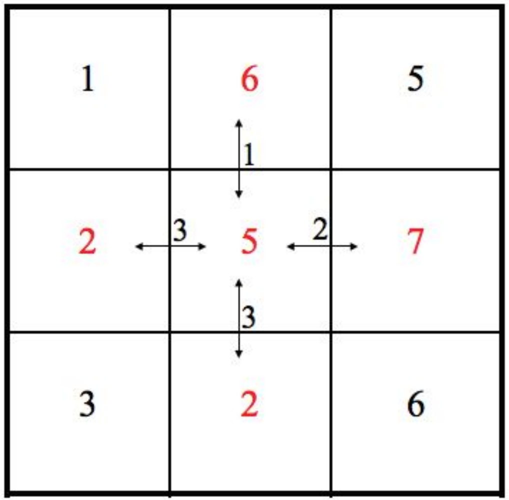
Kanten können aus einer Vielzahl von Gründen entstehen



Content Aware Resizing

Definiere eine Energie Funktion, die die Wichtigkeit eines Pixels beschreibt

z.B. wie verschieden ist ein Pixel von seiner Nachbarschaft?



Content Aware Resizing

Gradient Magnitude: $\sqrt{I_x^2 + I_y^2}$

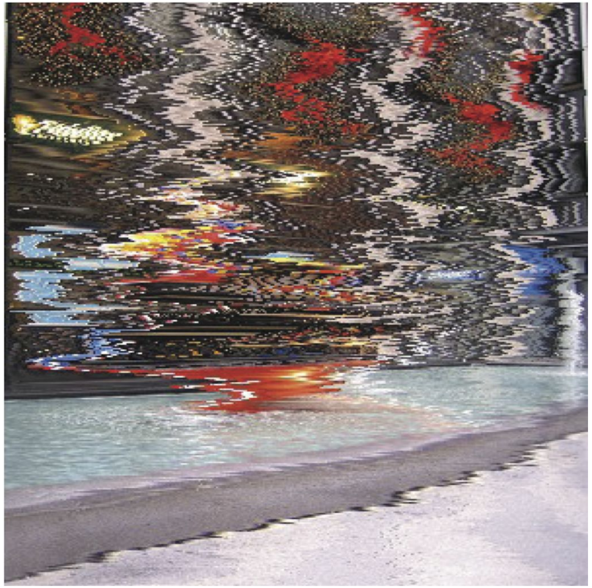
Typisch: $|I_x| + |I_y|$

Content Aware Resizing

Wie kann man garantieren, dass das Resultat “rechteckig” bleibt?

Content Aware Resizing

Entferne unwichtigsten Pixel in jeder Zeile, Starke Verzerrung

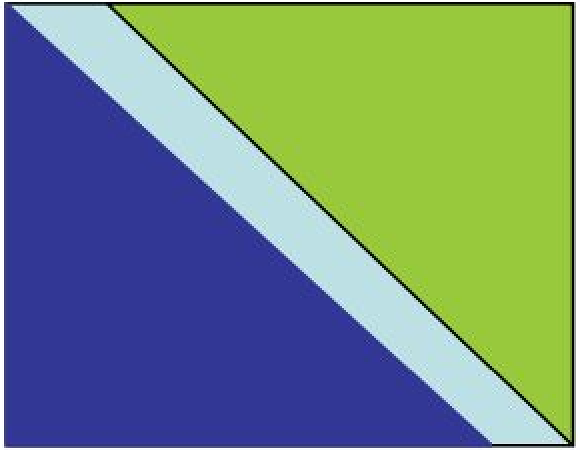


Content Aware Resizing

Entferne Spalte mit niedrigstem Gesamtwert, besser

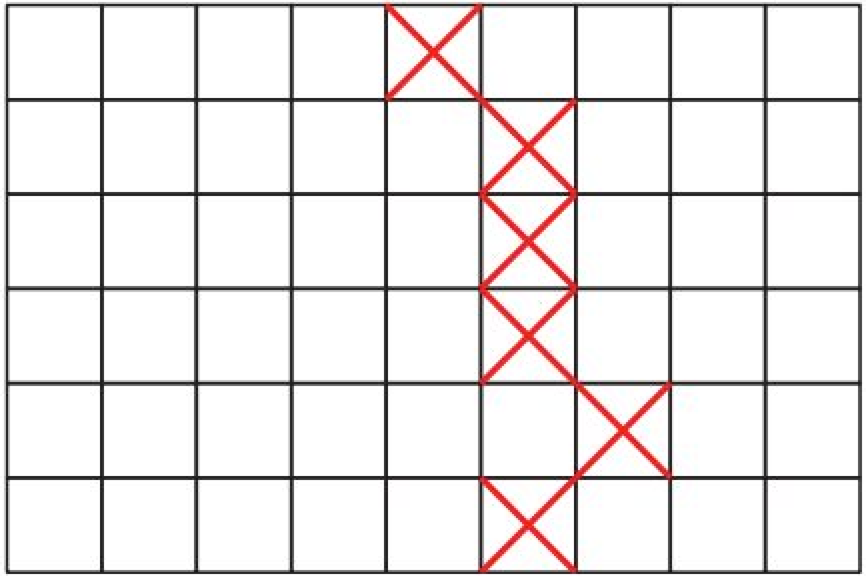
Content Aware Resizing

Nicht ideal, Was ist mit folgender Situation



Seam Carving

Idee: Entferne jeweils einen Seam (8-connected path)



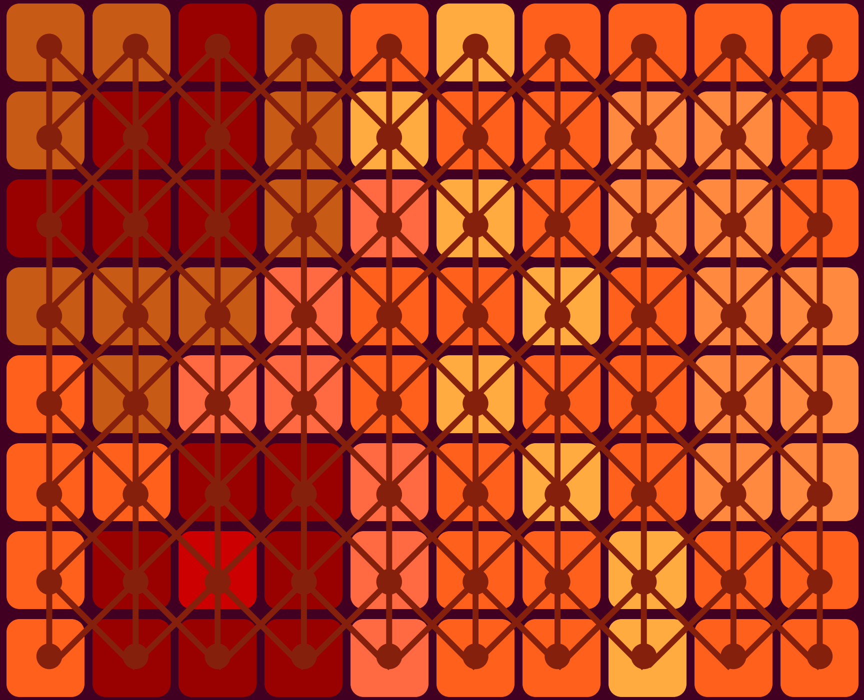
Seam Carving

Seam Carving

Indiana Jones / Seam Carving

- Wir suchen den günstigen Weg in einem DAG (directed acyclic graph)
- Dynamic Programming

Induktion



günstige Wege der Länge ≤ n-1

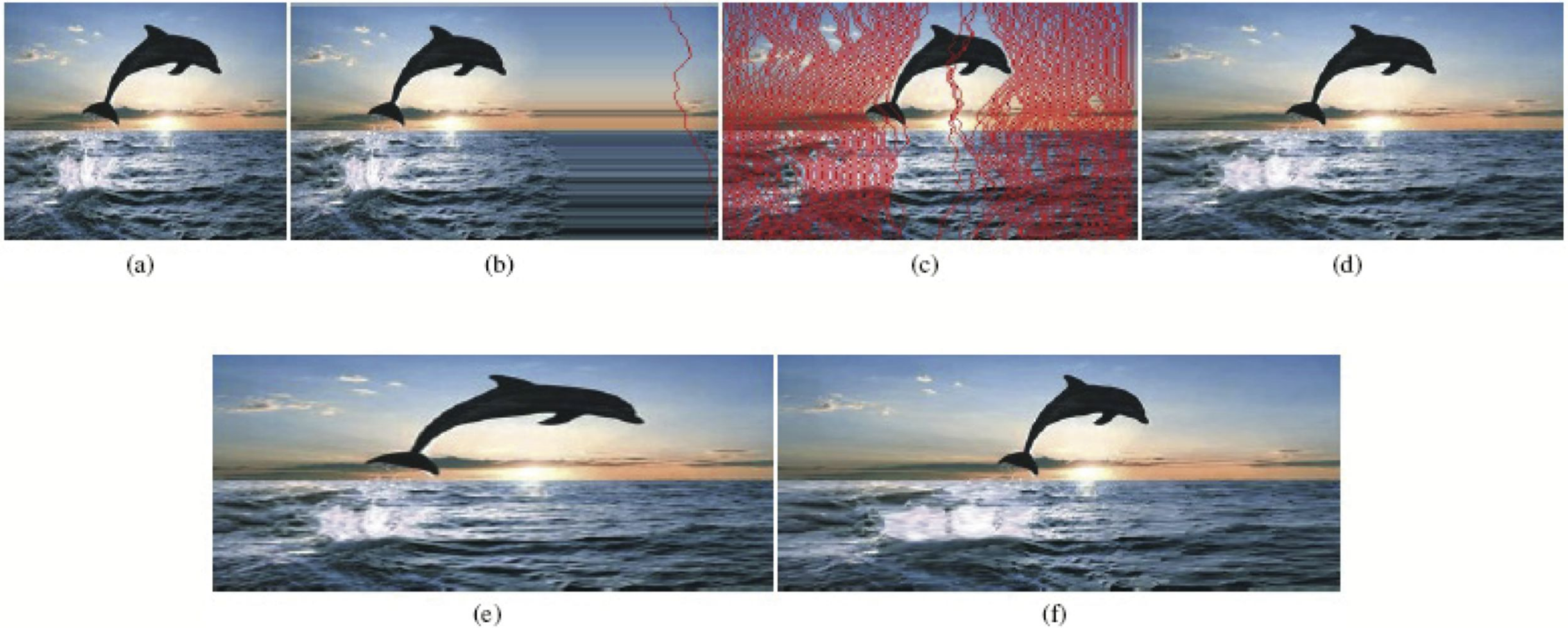
günstige Wege der Länge n

Rekursion

```
opt(x, y) = h(x, y) + min{
  opt(x-1, y-1),
  opt(x, y-1),
  opt(x+1, y-1)}
```

Content Enhancement

Vergrössern



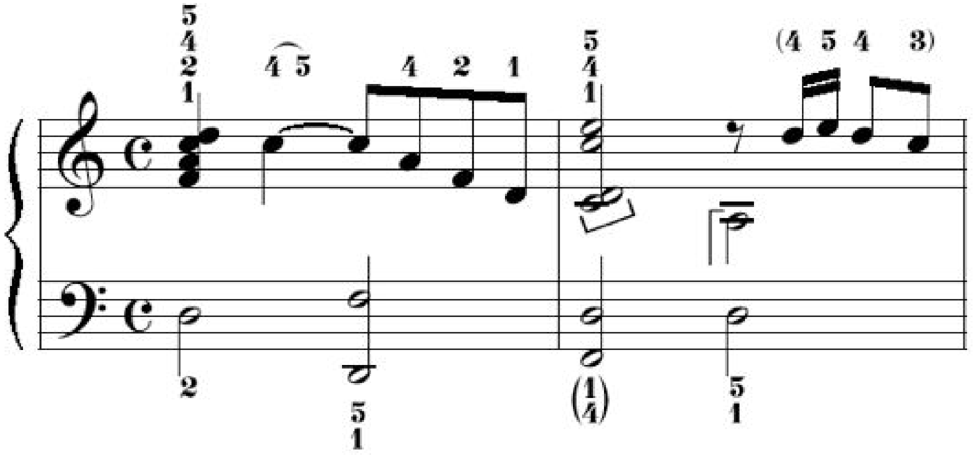
Assistance

Was ist wichtig?

Object Removal

Limitations

Fingersatz



Music Perception
Summer 1997, Vol. 14, No. 4, 341–382

An Ergonomic Model of Keyboard Fingering for Melodic Fragments

RICHARD PARNCUTT & JOHN A. SLOBODA
Keele University

ERIC F. CLARKE
Sheffield University

MATTI RAEKALLIO
Sibelius Academy of Music, Helsinski

PETER DESAIN
NICI, University of Nijmegen

3.2.1 Stretch rule

Assign 2 points for each semitone that an interval exceeds Maxcomf or is less than MinComf. (Parncutt et al.,1997,P349)

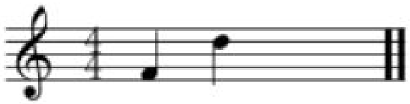


Fig. 3.2.1 Example of Strech Rule and Large Span Rule

3.2.6 Weak-Finger Rule

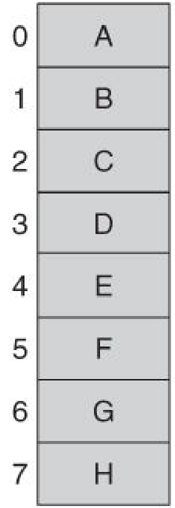
Assign 1 point every time finger 4 or finger 5 is used.(Parncutt et al.,1997,P 357)

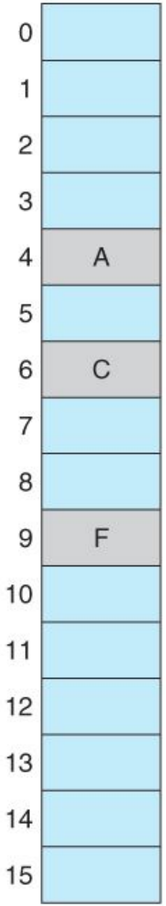
Table 2
Combined Effect of the Stretch, Small-Span, and Large-Span Rules for All Pairs of Right-Hand Fingers and All Intervals

Finger	Span (semitones)																				
Pair	–5	–4	–3	–2	–1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1-2	10	7	4	3	2		0	0	0	0	0	1	2	3	6	9					
1-3		11	8	5	4		2	1	0	0	0	0	0	1	2	3	6	9			
1-4			12	9	6		4	3	2	1	0	0	0	0	0	1	2	3	6	9	
1-5					12		6	5	4	3	2	1	0	0	0	0	1	2	3	6	9
2-3							0	0	2	6	10										
2-4							4	2	0	0	2	6	10								
2-5								6	4	2	0	0	2	4	8	12					
3-4							0	0	4	8											
3-5							4	2	0	0	2	6	10								
4-5							0	0	2	6	10										

Das Paging-Problem

Cache





physical memory

Online Algorithmus

- Approximationsalgorithmen: optimale Lösungen approximieren
- Online Algorithmen versuchen wir möglichst gute Lösungen zu finden wenn ein Algorithmus nicht alle *Informationen* zu Beginn hat.

Online Algorithmus

Competitive Ratio

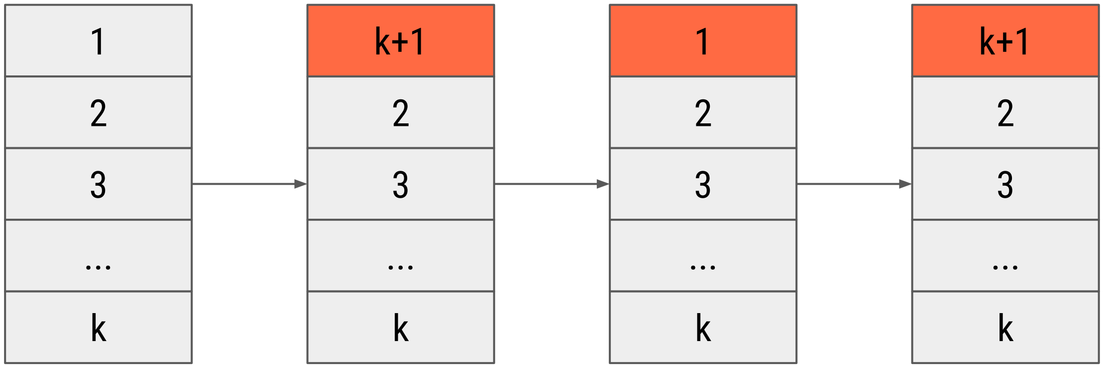
$$\forall \sigma : \frac{C_{\text{OPT}}}{C_A} \leq \alpha$$

LIFO

Last-In - First Out - ersetze Seite aus Cache, die zuletzt aufgenommen wurde

LIFO ist nicht kompetitiv

- Zu Beginn: 1, …, k im Cache
- Wähle abwechselnd Seite k+1 und 1



FIFO

First-In - First Out - ersetze Seite aus Cache, die am längsten im Cache ist

FIFO ist k-kompetitiv

- Phase: FIFO macht k Seitenfehler
- In der ersten Phase: OPT und FIFO gleich
- Betrachte andere Phase **P**
  - FIFO macht k Seitenfehler
  - Zeige, dass OPT mindestens einen macht
- Sei **s** diejenige Seite die als letzte vor Beginn von **P** abgefragt wurde
  - **s** ist sicher in Caches von OPT und FIFO
  - Cache von OPT und FIFO in ≤ k - 1 Seiten unterschiedlich
  - Wir zeigen: k verschiedene Seiten werden in **P** angefragt
  - → OPT hat mindestens einen Seitenfehler

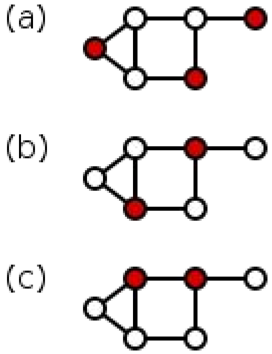
FIFO ist k-kompetitiv

- Eine Anfrage auf Seite **s** verursacht keinen Seitenfehler
    - Sie ist zu Beginn der Phase am jüngsten
    - Wird erst nach k-1 anderen Seitenfehlern ersetzt
- Kein Seitenfehler tritt auf, wenn eine Seite mehrmals angefragt wird
- D.h. k verschiedene Seiten werden angefragt, bis **s** ersetzt wird und Phase endet

Branch And Bound

Minimum Dominating Set:

- Jeder Knoten hat einen Nachbarn im Set
- Grösse ist minimal
- NP-vollständig



Branch And Bound

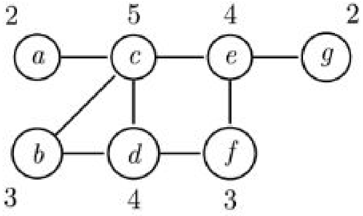
Untere Schranke

- Ein Knoten v heisst **dominiert** falls v in IN ist, oder einen Nachbarn darin hat
- δ_v sei Anzahl Knoten die zusätzlich dominiert werden wenn v in IN aufgenommen wird
- δ_v = 0 wenn v in IN
- δ_{max} = $\max \delta_v$ für v in $V \backslash OUT$
- Sei D die Menge noch nicht dominierter Knoten
    - Benötigen mindestens $|\mathrm{D}|/\delta_{max}$ zusätzliche Knoten
- $|\mathrm{IN}| + |\mathrm{D}|/\delta_{max}$

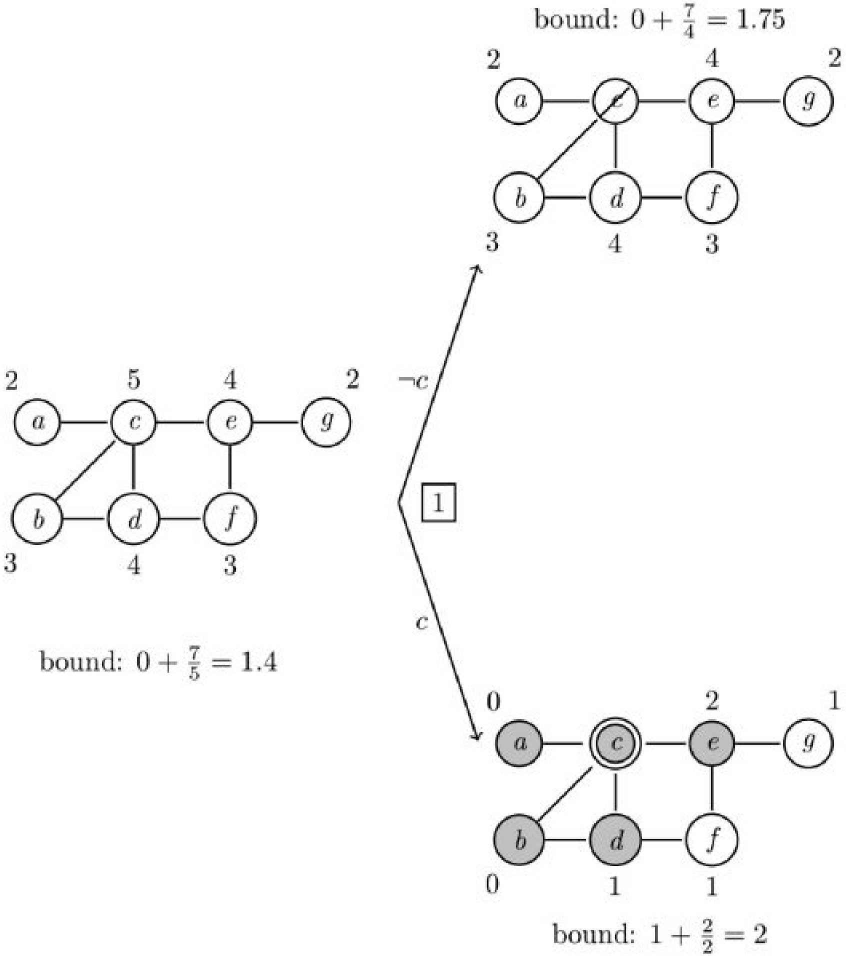
Branch And Bound

Regel

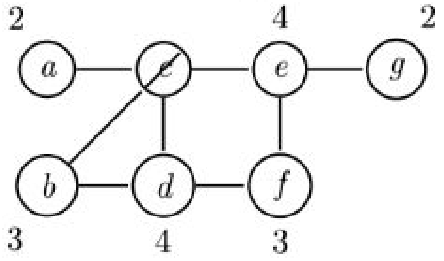
Wähle w in $V \backslash OUT$ mit: $\delta_w = \max \delta_v$

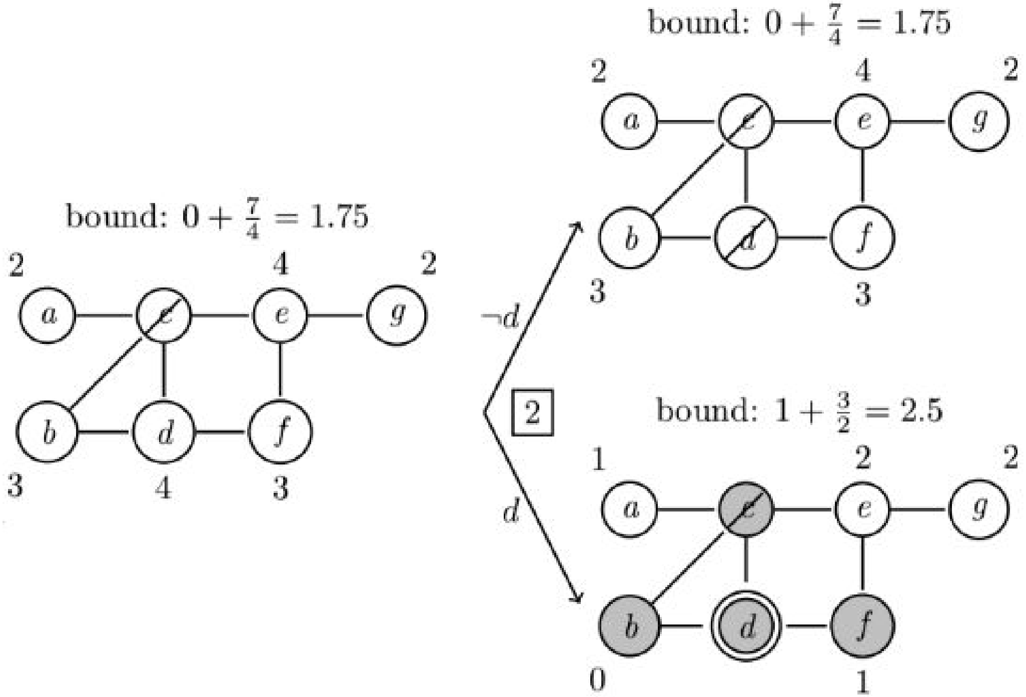


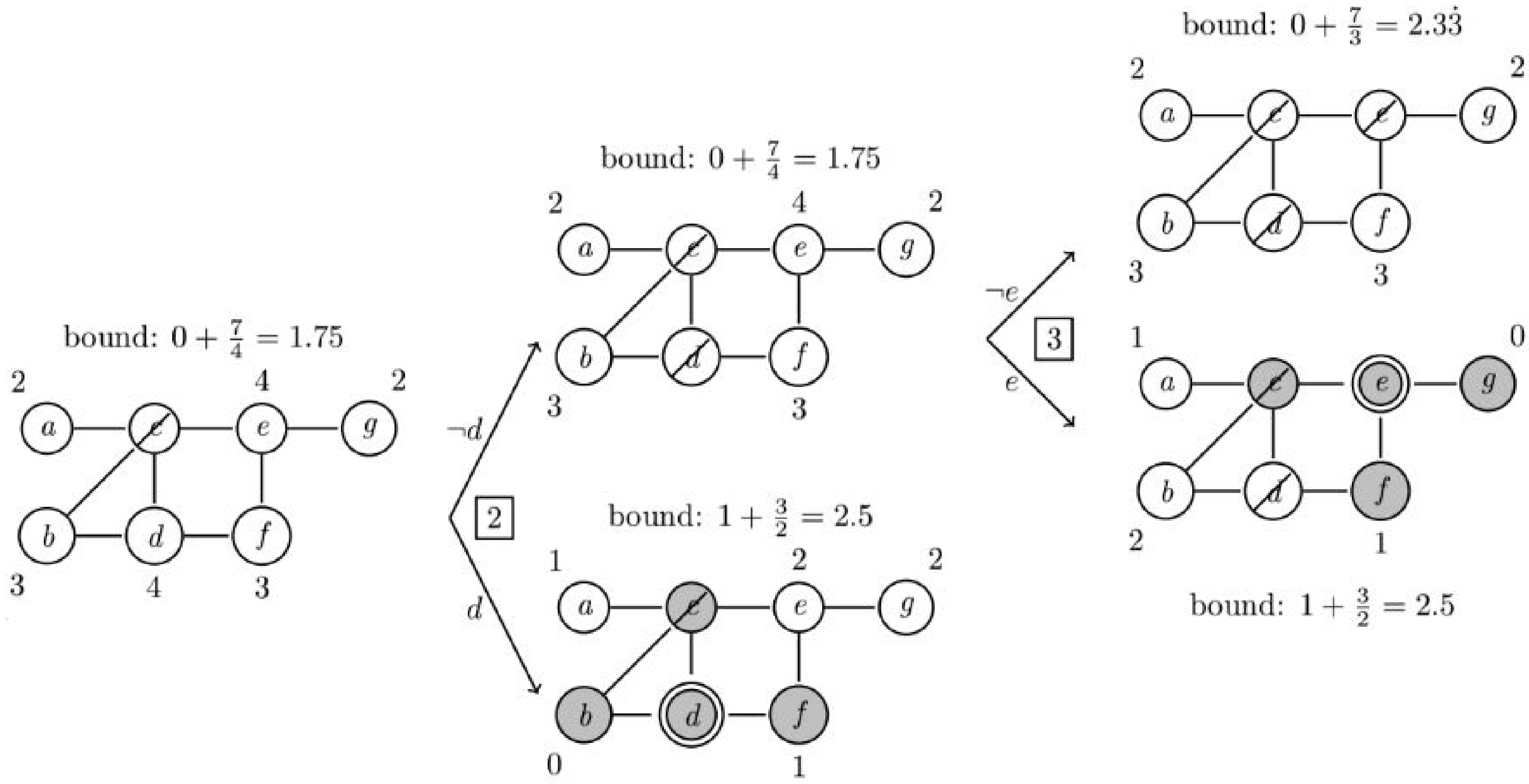
bound: $0 + \frac{7}{5} = 1.4$


bound: $0 + \frac{7}{5} = 1.4$
bound: $0 + \frac{7}{4} = 1.75$
bound: $1 + \frac{2}{2} = 2$
$\neg c$
c
1

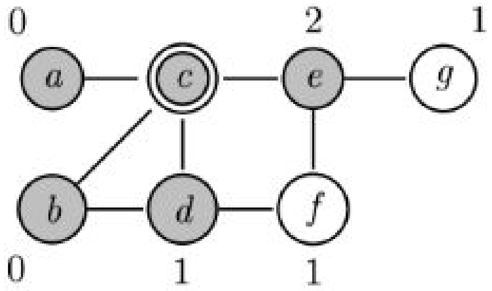
bound: $0 + \frac{7}{4} = 1.75$



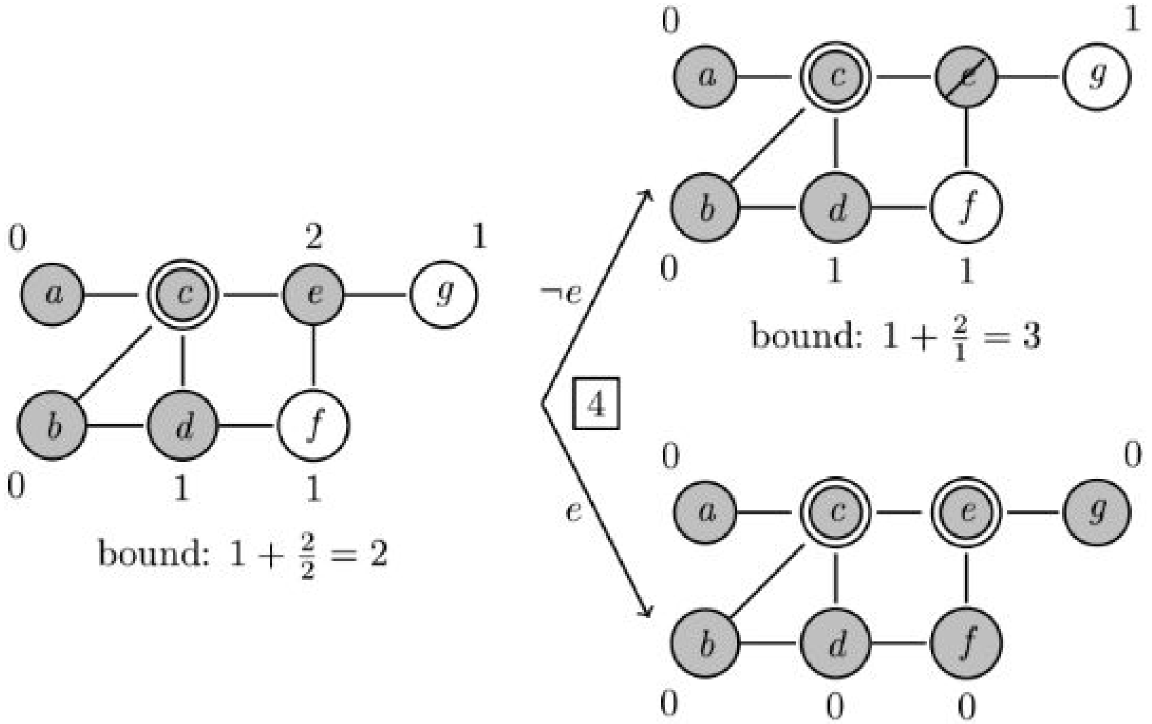

bound: $0 + \frac{7}{4} = 1.75$
$\neg d$
$\boxed{2}$
d
bound: $0 + \frac{7}{4} = 1.75$
bound: $1 + \frac{3}{2} = 2.5$



bound: $0 + \frac{7}{4} = 1.75$
bound: $0 + \frac{7}{4} = 1.75$
bound: $1 + \frac{3}{2} = 2.5$
bound: $0 + \frac{7}{3} = 2.3\dot{3}$
bound: $1 + \frac{3}{2} = 2.5$
$\neg d$
d
2
$\neg e$
e
3



bound: $1 + \frac{2}{2} = 2$



Neue obere Schranke ist 2 und kein Blatt hat eine kleinere untere Schranke.

Stop And Think

Minimum Dominating Set auf einem Baum

ACM ICPC Wordfinal

- Algorithmischer Programmierwettbewerb
- https://vis.ethz.ch/de/current/acmicpc
- Live Finale am 19. Mai: http://icpclive.com/