(b) $\Rightarrow$ (a) Ergänzungseigenschaft $\Rightarrow$ Greedy ist optimal für jede Gewichtung

→ Voraussetzung: wenn Y_1 und Y_2 beide unabhängige Mengen, und $|Y_1| < |Y_2|$, dann gibt es ein $x \in Y_2 \setminus Y_1$ so dass $Y_1 \cup \{x\}$ unabhängig

→ Wir zeigen: Greedy findet für jede Gewichtung w_x der Elemente $x \in X$ eine optimale Lösung

→ <u>Beweis durch Widerspruch</u>: nehmen wir an, dass es eine Gewichtung w_x gibt, so dass Greedy eine nicht-optimale unabhängige Menge Y ausgibt.
Sei Y^* eine optimale Lösung.

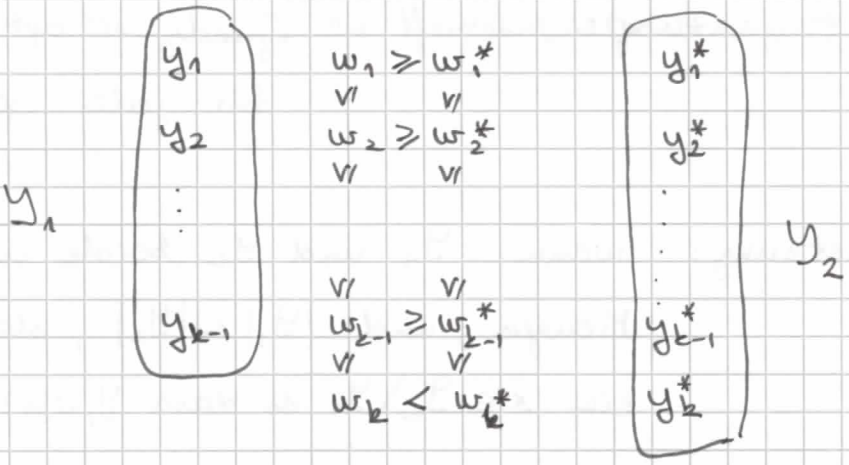
→ Wir können annehmen o.B.d.A., dass Y und Y^* nicht-vergrößerbar sind (sonst dürfte man sie vergrößern); deshalb gilt $|Y| = |Y^*| = m$ für irgendein m (sonst wäre die kleinere von Beiden aus der größeren ergänzbar)

→ (wie sich Y und Y^* schneiden, spielt keine Rolle)

→ Seien $Y = \{y_1, y_2, \ldots, y_m\}$
und $Y^* = \{y_1^*, y_2^*, \ldots, y_m^*\}$
so dass die entsprechenden Gewichte jeweils absteigend sortiert sind.

→ Da Y^* optimal ist, und Y nicht, gibt es Elemente y_i und y_i^* so dass $w_i < w_i^*$.
Seien y_k und y_k^* die ersten solche Elemente in der obigen Folge

H 12. Es gilt für die Gewichte



→ Nennen wir die markierten (eingerahmten) Mengen y_1 bzw. y_2

Die unabhängige Menge y_1 ist nicht vergrößerbar (ergänzbar) aus y_2, weil Greedy kein weiteres Element mit Gewicht größer als w_k findet, während in y_2 alle Gewichte größer sind als w_k.

Dies widerspricht die Ergänzungseigenschaft für y_1 und y_2

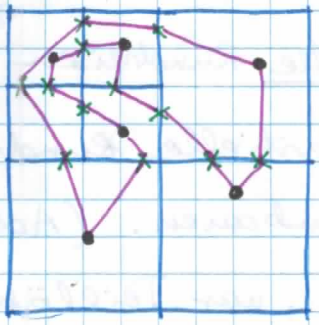
A4.

- jeder Knoten entspricht einem Quadrat; jeder Knoten im Baum B ist entweder ein Blatt oder hat vier Kinder (4-ärer Baum)

- der Algorithmus berechnet 'optimale' Teillösungen (partielle Rundreisen) zuerst für die Blätter, nachher für Knoten (Quadrate) auf höheren Ebenen, usw. (wir sehen später was 'optimal' bedeutet) → optimal unter Teillösungen, die bestimmte Randpunkt-Paare verbinden, und alle Orte im Quadrat besuchen

[die Tiefe des Baums B ist höchstens $\log n^2 = O(\log n)$
weil Quadrate der Länge 1 enthalten 0 oder 1 Punkt.
Die Anzahl seiner Knoten (der Quadrate) ist $O(n \cdot \log n)$ siehe später]

- Was ist eine partielle Rundreise, und wie werden diese zusammengebaut?

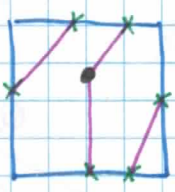


× Tür

• Punkt (Ort)

zusammenpassende partielle Rundreisen

→ wenn ein Quadrat im B einem Blatt entspricht, besteht eine partielle Rundreise aus geraden Segmenten zwischen Seiten des Quadrats (mit dem geringsten "Umweg" (ggf.) zum einzigen Ort im Quadrat → dazu später)



die partielle Rundreise hat keine Richtung(en)

→ wir möchten, dass wir solche partielle Rundreisen für die Teilquadrate aneinander kleben können, so dass sie Teillösungen (partielle Rundreisen) für ihr Vater-Quadrat bestimmen.

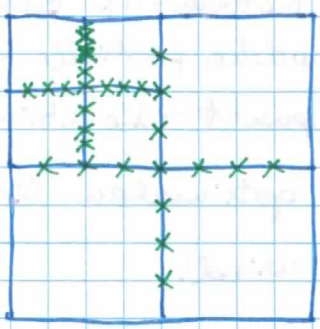
Aber: die Treffpunkte × der partiellen Rundreisen in den Teilquadraten müssen zusammenpassen, sonst bilden sie keine partielle Rundreise im Vater-Quadrat

⇒ wir müssen (die Menge der) möglichen Treffpunkte × am Anfang festlegen (damit es nicht unendlich viele Möglichkeiten gibt); diese festgelegten Treffpunkte nennen wir Türen.

Die Anzahl der Türen muss sorgfältig gewählt werden:

(Zu wenig Türen: zu lange Umwege in der gefundene Rundreise (zu "eckig"), und deshalb schlechte Approximation

Zu viele Türen: zu viele mögliche Partielle Rundreisen in der Tabelle der dynamischen Programmierung verursacht nicht-polynomielle Laufzeit)



— an beiden Trennlinien eines Teilquadrats legen wir m Türen im gleichen Abstand fest. $\left(m \approx \frac{\log_2 n}{\varepsilon}\right)$

die Anzahl m der Türen ist gleich für die langen wie für die kürzeren Trennlinien.

[die Distanz benachbarter Türen auf einer Trennlinie der Schicht i ist $\sim \frac{n^2}{2^i \cdot m}$]

A6.

- eine (komplette) Rundreise heißt legal wenn sie jede Trennlinie nur in Türen kreuzt.
  Unser Algorithmus berechnet mit dynamischer Programmierung eine minimale legale Rundreise
  (für den Approximationsfaktor muss man später zeigen, dass für eine optimale Rundreise eine legale Rundreise existiert die höchstens um Faktor $1+O(\varepsilon)$ länger ist)

- die minimale legale Rundreise wird bottom-up berechnet, angefangen mit den kleinsten Quadraten mit maximal einem Ort.
  Für jedes Quadrat wird für alle möglichen festgelegte Randpunkt-Paare (Tür-Paare)
  Treffpunkt-Kombinationen am Rand des Quadrats die minimale Länge einer legalen Rundreise berechnet und gespeichert.
  Beachte dass bis zum Ende der Bottom-up Berechnung (bzw. welche partielle Rundreise) nicht bekannt ist, welche Treffpunkt-Kombination am Rand des Quadrats in der optimalen legalen Rundreise letztendlich vorkommen wird!

  Was bedeutet eine Treffpunkt-Kombination am Rand eines Quadrats?? (bzw. Tür-Kombination)

Treffpunkt-Kombination = Besuchsmuster

Ein Besuchsmuster $\tilde{T}$ für ein Teilquadrat (für einen Knoten von B) ist eine ~~Vektor~~ Menge von Paaren von Türen am Rand dieses Quadrats:

$$\tilde{T} = \{\{T_1 T_1'\}, \{T_2 T_2'\}, \{T_3, T_3'\} \ldots \{T_r T_r'\}\}$$

die Reihenfolge der Paare und der Türen innerhalb eines Paars zählt nicht (ist irrelevant)

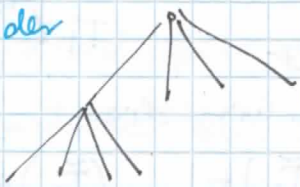
Die Bedeutung eines Besuchsmusters ist, dass eine Rundreise (die zu diesem speziellen Besuchsmuster passt / mit ihm konsistent ist) für jede $1 \leq i \leq r$ das Teilquadrat einmal über Tür T_i betritt, und dann über Tür T_i' verlässt, (oder andersrum).

> Dynamische Programmierung für die Berechnung einer minimalen legalen Rundreise

Wie für minimum-Gewicht Vertex Cover auf Bäumen, berechnet der Algorithmus für jeden Knoten von B (jedes Teilquadrat) die minimale Länge einer partiellen Rundreise für jedes mögliche Besuchsmuster:

sei $L_Q(\tilde{T})$ diese minimale Länge für Quadrat Q und eines seiner Besuchsmuster $\tilde{T}$

die Werte in der Tabelle



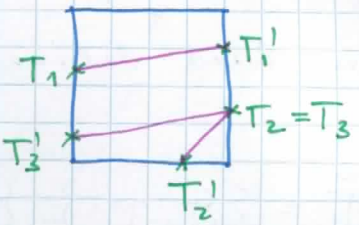
bottom-up Berechnung

A8.

Die Rekursion:

Basisfälle: (sind die Blätter in B)

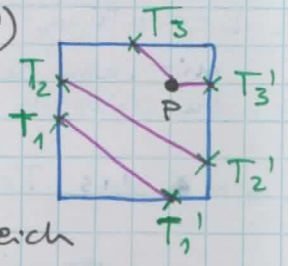
- für ein leeres Blatt-Quadrat Q ohne Eingabepunkt ist $L_Q(\tilde{T})$ einfach die Gesamtlänge der Segmente (Distanzen) $(T_i T_i')$ für jedes Tür-Paar im Besuchsmuster:



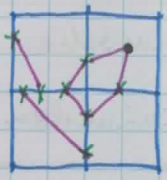
$$\tilde{T} = \{\{T_1 T_1'\}, \{T_2 T_2'\}, \{T_3 T_3'\}\}$$

$$L_Q(\tilde{T}) = \sum_{i=1}^{r} d(T_i T_i')$$

- für ein Blatt-Quadrat mit einem Eingabepunkt P ähnlich aber es wird für ein Tür-Paar statt $d(T_i T_i')$ ein Umweg zu P gerechnet so dass dieser Umweg (d.h. dieses Paar) die genaupte zusätzliche Länge (im Vergleich zu $\sum_i d(T_i T_i')$) verursacht.



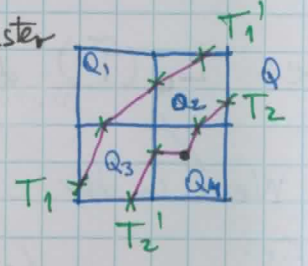
(nicht-passende Besuchsmuster)



"Rekursionsgleichung" (für innere Knoten)

- für ein Vater-Quadrat Q und ein Besuchsmuster $\tilde{T}$
  - betrachte alle Kombinationen von Besuchsmuster der 4 Kinder-Quadrate $Q_1\ Q_2\ Q_3\ Q_4$ $\left(\tilde{T}_1, \tilde{T}_2, \tilde{T}_3, \tilde{T}_4\right)$
  - diese nennen wir passend zueinander und zu $\tilde{T}$ falls die Türen all dieser Besuchsmuster zusammenpassen und so, dass eine (partielle) entsprechende Rundreise mit dem Besuchsmuster $\tilde{T}$ konsistent ist. In diesem Fall addieren wir die Längen

(Beachte dass passende Türen nicht reichen)



$$L_{Q_1}(\tilde{T}_1) + L_{Q_2}(\tilde{T}_2) + L_{Q_3}(\tilde{T}_3) + L_{Q_4}(\tilde{T}_4)$$

— Schließlich ist $L_Q(\tilde{T})$ für das Vater-Quadrat das Minimum dieser Gesamtlösungen über alle zu $\tilde{T}$ passenden $(\tilde{T}_1\ \tilde{T}_2\ \tilde{T}_3\ \tilde{T}_4)$

$$L_Q(\tilde{T}) = \min\left\{L_{Q_1}(\tilde{T}_1)+L_{Q_2}(\tilde{T}_2)+L_{Q_3}(\tilde{T}_3)+L_{Q_4}(\tilde{T}_4) \;\middle|\; (\tilde{T}_1,\tilde{T}_2,\tilde{T}_3,\tilde{T}_4) \overset{\text{alle}}{\text{passend zu}}\ \tilde{T}\right\}$$

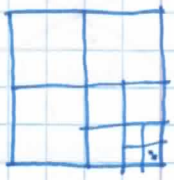
c.) Einige Details zur Laufzeitanalyse

— Die Anzahl aller möglichen Besuchsmuster $\tilde{T}$ für ein Quadrat ist $2^{O(m)} = n^{O(\frac{1}{\varepsilon})}$ (wir sehen später warum)

$\left(2^m = 2^{\frac{\log n}{\varepsilon}} = n^{\frac{1}{\varepsilon}}\right)$

— die Laufzeit für ein Quadrat = einen Baum-Knoten ist somit $\left[2^{O(m)}\right]^5 = 2^{O(m)}$, weil alle möglichen Kombinationen $(\tilde{T}, \tilde{T}_1, \tilde{T}_2, \tilde{T}_3, \tilde{T}_4)$ für sein eigenes Besuchsmuster und für seine 4 Teilquadrate betrachtet werden.

— B hat $O(n \cdot \log n)$ Knoten (zwei nahe Orte können für $O(\log n)$ leere Quadrate die Verantwortung tragen:



)

die Laufzeit für den ganzen Baum ist somit

$$O(n \log n) \cdot 2^{O(m)}$$

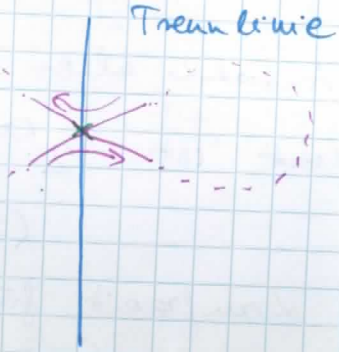
$$= \text{Poly}(n) \cdot n^{O(\frac{1}{\varepsilon})}$$

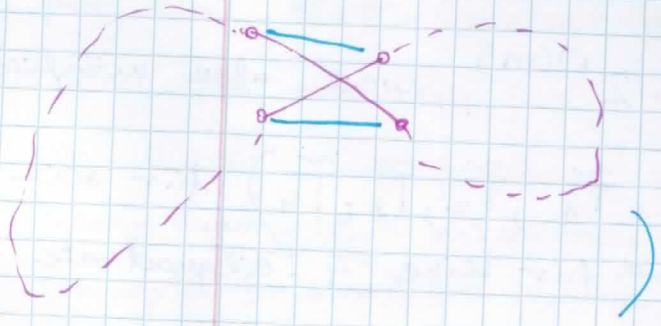
— Entscheidend für diese Analyse war die Anzahl der Besuchsmuster $2^{O(m)}$

- Wamm ist die Anzahl der Besuchsmuster $\tilde{T}$ für ein Quadrat $2^{O(m)}$?

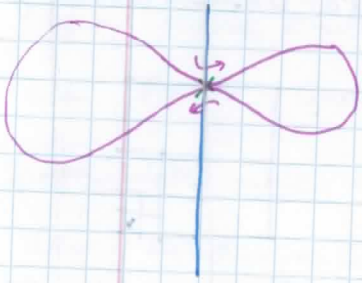
Behauptung 1. Es existiert eine kürzeste legale Rundreise die sich gar nicht kreuzt, und sich höchstens in Türen trifft.

(Wamm? jede Kreuzung kann aufgehoben werden so dass die Rundreise kürzer wird

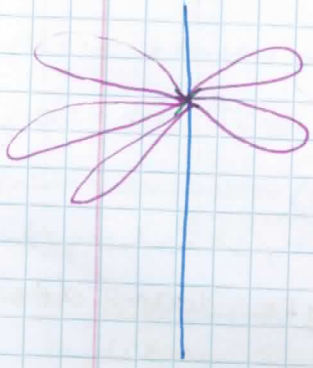




Behauptung 2: Eine kürzeste Rundreise besucht jede Tür höchstens zweimal.



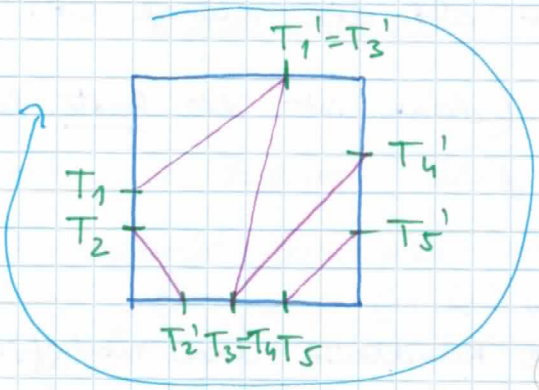
← das ist erlaubt



↙ das ist nicht erlaubt so eine Rundreise könnte gekürzt werden.

Wir können die Berechnung auf solche $\tilde{T}$ einschränken:

Korollar: Wenn wir die Tür-Paare in einem Besuchsmuster $\tilde{T}$ betrachten, die "Kanten" (Strecken) zwischen jedem Paar kreuzen sich nicht, (und jede Tür kommt maximal zweimal im Besuchsmuster vor).



$\tilde{T} = \{T_1, T_4'\}, \{T_2, T_2'\}\{T_3, T_3'\}$.

$3m$ $\binom{3m}{2}$ Paare

$2^{\binom{3m}{2}}$ Besuchsmuster $O(m^2)$

$(2^{\log n})^{\log n} = n^{\log n} = 2^{\log^2 n}$ → zu viel

Laufen wir am Rand des Quadrats (einmal) herum, und notieren wir die Türen, und markieren wir sie mit einer öffnenden bzw. schließenden Klammer beim Treffen der ersten bzw. zweiten Tür desselben Tür-Paares

$T_1'\ T_3'\ T_4'\ T_5'\ T_5\ T_4\ T_3\ T_2'\ T_2\ T_1$

$(\ (\ (\ (\ )\ )\ )\ (\ )\ )$

→ wir erhalten einen sog. wohlgeformten Klammerausdruck der Länge $\leq 8m$ (weil jede Trennlinie max m Türen hat, und eine Tür wird max. 2mal benutzt).

Thm: Die Anzahl der wohlgeformten Klammerausdrücke der Länge k ist die k-te Catalan-Zahl

$$C_k = \frac{1}{k+1}\binom{2k}{k} = 2^{O(k)}$$

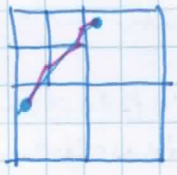
Grob: "deshalb" gibt es $2^{O(m)}$ "legale" Besuchsmuster für jedes Quadrat. (präzisere Analyse in Vazirani)

d.) Einige Details zum Approximationsfaktor.

Der Algorithmus berechnet eine optimale legale Rundreise

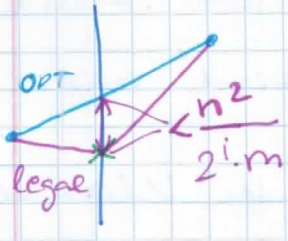
Wir brauchen: irgendeine legale Rundreise ist höchstens um $O(\varepsilon) \cdot OPT$ länger als die Länge OPT einer optimalen Rundreise (dann ist die beste legale Rundreise auch so)

– wir nehmen eine optimale Rundreise und modifizieren sie um eine legale Rundreise zu erhalten:



(wir ersetzen jede Strecke der optimalen Rundreise mit einem Polygonzug: wir verschieben jede Kreuzung mit einer Trennlinie in die nächstliegende Tür ×)

[die Verlängerung wegen einer Kreuzung mit einer Trennlinie der Schicht i ist höchstens $\frac{2n^2}{2^i \cdot m}$



an Trennlinien der niedrigen Schichten werden diese Verlängerungen länger; aber solche Trennlinien gibt es weniger!]

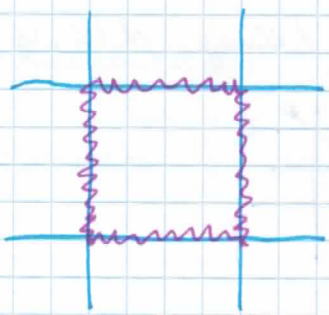
Dies kann viel zu lang werden, wenn die optimale Rundreise eine Trennlinie mit wenigen Türen zu oft kreuzt!

große Umwege zu einer Tür

[z.B. die Trennlinie der Schicht 0 m-mal kreuzen könnte $\sim \frac{n^2}{m} \cdot m = \Theta(n^2)$ Verlängerung verursachen und das ist [illegible] $\cdot OPT$!]

Beispiel

alle (vereinigten) Trennlinien der Schicht 1



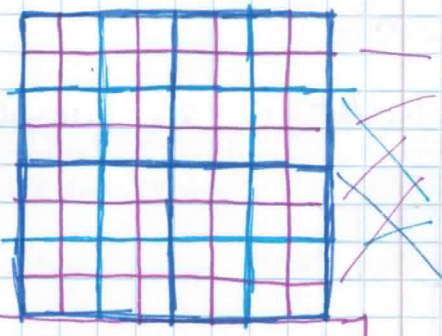
und eine optimale Rundreise die sie häufig kreuzt

ABER in diesem Beispiel haben wir offensichtlich Pech gehabt bei der Positionierung der Trennlinien!

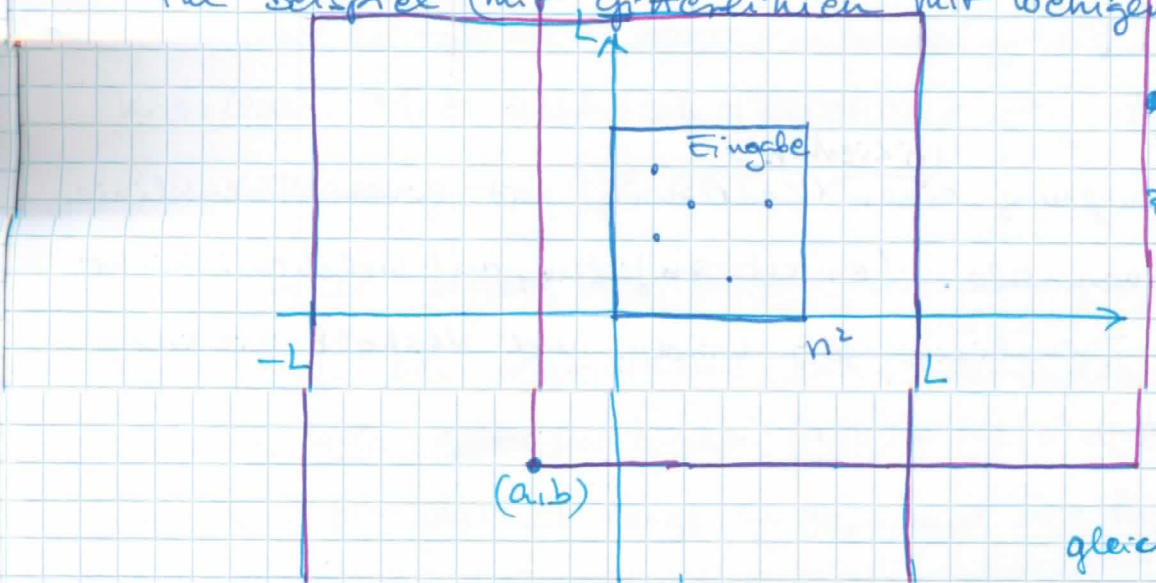
Wichtig: es gibt immer (viele) vorteilhafte Positionierungen der Trennlinien durch Verschiebung des ganzen Gitters.

Idee: [Die Gesamtanzahl aller Kreuzungen mit Trennlinien ist nicht mehr als 2-OPT (Bitte ausprobieren wieviele (ganzzahlig positionierte) Trennlinien eine Strecke der Länge 5 kreuzen kann)]

wenn das Gitter aller Trennlinien (mit ihren Schichten zusammen)



zufällig verschoben, bzw. am Anfang zufällig positioniert wird, ist unwahrscheinlich dass die Situation im Beispiel (mit Gitterlinien mit wenigen Tiefen) vorkommt.



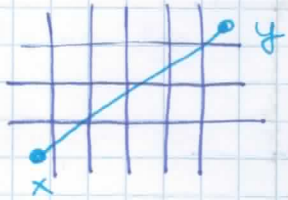
ein Gitter der Breite 2L wird zufällig positioniert (für $L > n^2$ L zweierpotenz) s.d. seine Ecke (a,b) zufällig in $[-L,0] \times [-L,0]$ gleichverteilt fällt.

A14.

für jede $(a,b) \in [-L,0] \times [-L,0]$ enthält es die ganze Eingabe

[Die präzise Analyse ist wie folgt:

— Eine Kante der Rundreise der Länge $d(x,y)$

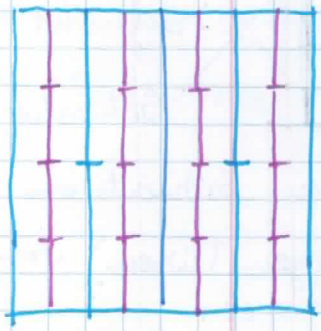


kreuzt maximal $d(x,y)$ horizontale und maximal $d(x,y)$ vertikale Trennlinien $\Longrightarrow$ die optimale Rundreise hat maximal $2OPT$ Kreuzungen mit Trennlinien insgesamt.

— Dank der zufälligen Positionierung des Gitters hat jede ganzzahlige Linie die Wahrscheinlichkeit

$$\frac{2^i}{2L} \approx \frac{2^i}{2n^2}$$

um eine (Vereinigung von) Trennlinie(n) der Schicht i zu werden:



Schicht 2 1 2 0 2 1 2

— Sei die Verlängerung einer fixierten Kreuzung mit einer Trennlinie eine Zufallsvariable. (es ist zufällig, auf welcher Schicht die betroffene Trennlinie sein wird, und deshalb wieviele Türen sie hat)

Da eine Linie mit Wahrscheinlichkeit $\frac{2^i}{2n^2}$ eine Trennlinie der Schicht i wird, ist die Verlängerung (höchstens) $\frac{2n^2}{2^i \cdot m}$ lang mit Wahrscheinlichkeit $\frac{2^i}{2 \cdot n^2}$. Der Erwartungswert der Verlängerung wegen einer Kreuzung ist somit

$$E\left(\text{Verlängerung einer konkreten Kreuzung der opt. Rundreise mit einer ganzzahligen Linie}\right) =$$

(max Anzahl der Schichten $\leftarrow 2\log n$)

$$= \sum_{i=0}^{2\log n} \underbrace{\frac{2n^2}{2^i \cdot m}}_{\text{Verlängerung mit dieser Wahrscheinlichkeit}} \cdot \underbrace{\frac{2^i}{2n^2}}_{\text{Wahrscheinlichkeit}} = \sum_{i=0}^{2\log n} \frac{1}{m} = \frac{2\log n}{m} = \frac{2\log n \cdot \varepsilon}{2\log n} = \varepsilon$$

$$E(\text{Verlängerung für alle Kreuzungen}) \leq \varepsilon \cdot \#\text{Kreuzungen} \leq$$

$$\leq \varepsilon \cdot 2 \cdot OPT$$

es folgt: Thm: Die erwartete Verlängerung einer konkreten Kreuzung mit einer Linie wegen Verschiebung in eine Tür ist $< \varepsilon$.

Theorem: Die erwartete Verlängerung der optimalen Rundreise dadurch dass man sie in eine legale Rundreise verwandelt ist $\leq 2 \cdot \varepsilon OPT$

(weil es maximal $2OPT$ Kreuzungen gibt)

Korollar: Mit Wahrscheinlichkeit $\geq \frac{1}{2}$ ist die Verlängerung $\leq 4 \cdot \varepsilon \cdot OPT$

(sonst wäre der Erwartungswert größer als $2\varepsilon OPT$)

Bemerkung: Der Algorithmus kann derandomisiert werden indem alle Positionen $(a,b) \in [-L,0] \times [-L,0]$ ausprobiert werden für das $2L$ lange Gitter, und die kürzeste Lösung gewählt wird.

A16.

Arora's PTAS (Zusammenfassung)

n Punkte in $\mathbb{R}^2$ sind gegeben

- verschiebe, skaliere und runde die Instanz, sd. $[0, n^2] \times [0, n^2]$ das kleinste umschließende Quadrat ist.
- sei $L > n^2$ Zweierpotenz und Q_0 ein Quadrat der Länge $2L$ mit Eckpunkt $(a, b) \in [-L, 0] \times [-L, 0]$ zufällig (gleichverteilt) gewählt.

- definiere den 4-ären Baum B der Teilquadrate von Q_0, und m Türen auf jeder Trennlinie

$$m > \frac{2 \cdot \log n}{\varepsilon}$$

- bestimme für jeden Knoten (Quadrat) Q im B bottom-up ~~für jedes~~ Besuchsmuster $\tilde{\tilde{T}}$

$L_Q(\tilde{\tilde{T}})$ = die kürzeste Länge einer legalen partiellen Rundreise konsistent mit diesem Besuchsmuster

~~xxxx~~ L_{Q_0} ist die optimale Länge einer legalen Rundreise

- bestimme die optimale legale Rundreise top-down

Theorem: - Arora's Algorithmus findet eine Rundreise mit erwarteter Länge $\leq (1 + 2\varepsilon)\,OPT$.

- Mit Wahrscheinlichkeit $\geq \frac{1}{2}$ hat die Rundreise Länge $\leq (1 + 4\varepsilon)\,OPT$.

- Die Laufzeit ist $O(n \cdot \log n) \cdot n^{O(\frac{1}{\varepsilon})}$

Der Algorithmus ist wegen enormer Laufzeit nicht praktikabel. Er sagt uns: es besteht die Hoffnung, dass einfachere Methoden mit guter Approximation gefunden werden.

BRANCH & BOUND

(für Optimierungsprobleme)

a) Einführung, Grobstruktur

Bei schwierigen Problemen ist der Lösungsraum groß (es gibt mindestens exponentiell-viele mögliche Lösungen).

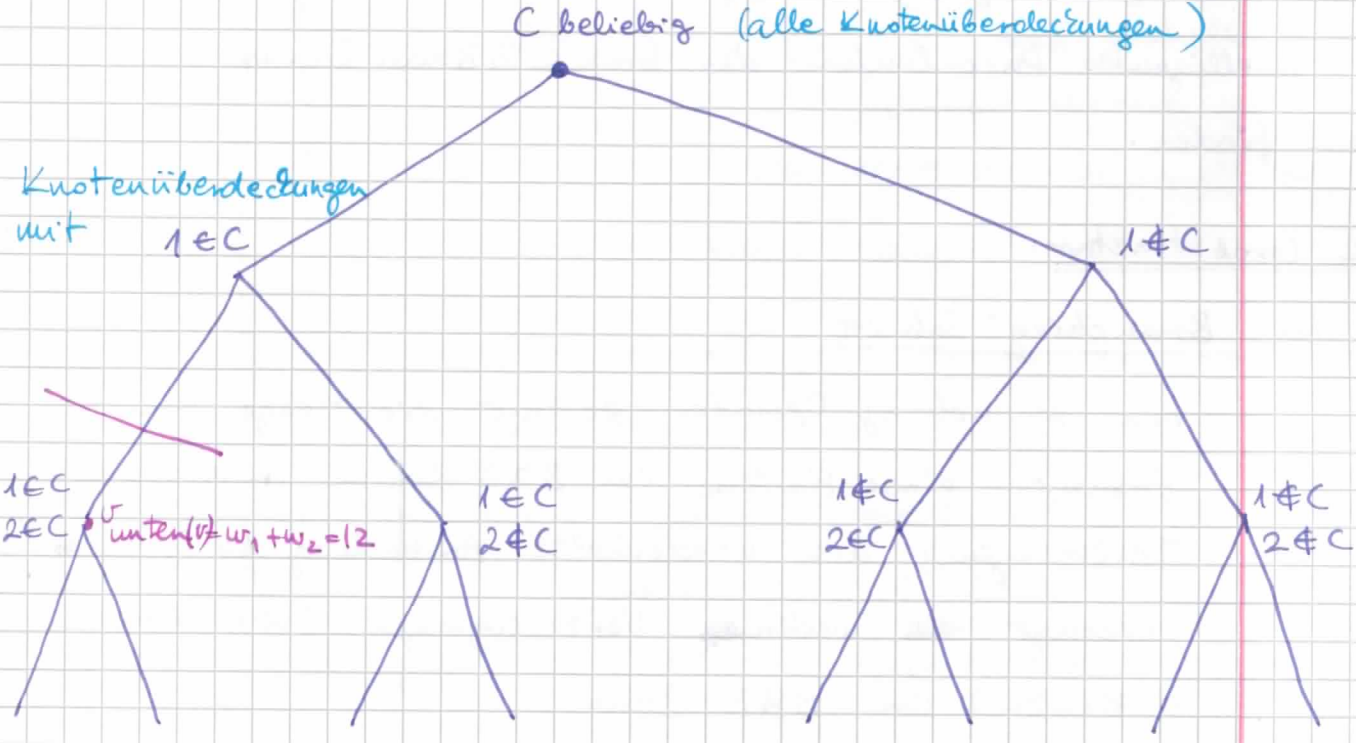
Theoretisch könnte man eine (erschöpfende Suche) vollständige Suche (die alle Lösungen betrachtet und eine optimale auswählt) gemäß eines „Suchbaums" (hier später: Branch & Bound Baum) durchführen, wie im folgenden einfachen (theoretischen!) Beispiel:

(Dummes) Beispiel 1.

VERTEX COVER

Sei $G(V,E)$ ein Graph mit $V = \{1, 2, 3, \ldots, n\}$ und Knotengewichten $w_1, w_2, \ldots w_n$

Finde eine Knotenüberdeckung $C \subseteq V$ mit minimalem Gewicht.



Zweige, wo sich keine Knotenüberdeckungen mehr befinden kann, können abgebrochen werden (z.B. hier wenn $\{1,2\} \in E$)

Tiefe des Baumes: n

Laufzeit $\Omega(2^n)$ im Worst-Case, wenn alle möglichen Lösungen geprüft werden!

(Warum? Schließlich entspricht jedes Blatt einer Lösung C. Die Anzahl der Blätter kann für die meisten Graphen $G(V, E)$, $\Omega(2^n)$ werden (d.h. die Zweige nicht früher abgeschnitten).)

Was ist nun Branch & Bound?

Angenommen, eine Heuristik fand am Anfang eine Knotenüberdeckung mit Gewicht 10. Seien außerdem $w_1 = 7$ und $w_2 = 5$ die Gewichte der Knoten 1 bzw. 2.

In diesem Fall braucht der Algorithmus im linkesten Teilbaum gar nicht weiterzusuchen, dort werden alle C mindestens Gewicht $7+5=12$ haben!

Ein Branch & Bound Algorithmus (Heuristik) versucht eine optimale (oder gute) Lösung durch eine intelligente Suche — intelligentes Durchlaufen des Branch & Bound Baums — zu finden.

Die Grobstruktur:

Im Branching Schritt

- ein Branching Operator zerlegt die Menge von Lösungen eines Knotens im B&B-Baum in disjunkte Teilmengen; die wiederholte Anwendung des Operators erzeugt weitere Verzweigungen (d.h. Kanten u. Knoten) im B&B-Baum.

Die Wurzel entspricht der Menge aller Lösungen.
Sei v ein Knoten mit Lösungsmenge $L(v)$ und Kindern $v_1, v_2 \ldots v_i$ im Baum. Dann ist $L(v_1), L(v_2), \ldots, L(v_i)$ eine Zerlegung (Partitionierung) von $L(v)$.

(Zb. die Menge

$$\{ C \text{ Knotenüberdeckung} \mid 1 \in C, 2 \notin C \}$$

kann zerlegt werden in die Lösungsmengen

$$\{ C \text{ Knotenüberdeckung} \mid \{1,3\} \subset C, 2 \notin C \}$$

und $\{ C \text{ Knotenüberdeckung} \mid 1 \in C, 2,3 \notin C \}$)

Bounding Schritt: (sei $\mathcal{B}$ der B&B-Baum)

- angenommen, für jeden Knoten $v \in \mathcal{B}$ gibt es eine (effizient berechenbare) untere Schranke unten(v) so dass

$$\text{unten}(v) \leq f(y)$$

für jede Lösung in $y \in L(v)$ gilt (f ist die Zielfunktion).

→ v kann verworfen werden, wenn

$$\text{unten}(v) \geq f(y_0)$$ für eine aktuelle beste bekannte Lösung y_0.

(das Blatt v wird deaktiviert)

Wie in unserem Beispiel mit

$$10 = f(y_0) \leq \text{unten}(v) = w_1 + w_2 = 12$$

($f(y_0)$ ist eine aktuelle globale obere Schranke für das Optimum, die unten(v) sind lokale untere Schranken für die Teilbäume (mit Wurzel v jeweils))

BB4.

Branch & Bound Algorithmus (für Minimierungsprobleme oBdA.)

→ anfänglich besteht der B&B Baum $\mathcal{B}$ nur aus der aktivierten Wurzel

→ eine Lösung y_0 wird mit Hilfe einer Heuristik (6. Approximationsalgorithmus) berechnet

→ WHILE es gibt aktivierte Blätter, DO

- wähle das erfolgversprechendste aktivierte Blatt v in $\mathcal{B}$
- wende den Branching-Operator auf v an, um die Kinder $v_1, v_2, \ldots, v_k$ zu erhalten
- FOR $i = 1$ TO k DO (berechne unten(v_i) ~~und eine Lösung $y_i \in L(v_i)$~~)
  IF unten$(v_i) \geq f(y_0)$ deaktiviere v_i
  (ggf. berechne eine Lösung $y_i \in L(v_i)$
  IF ~~[illegible]~~ $f(y_i) < f(y_0)$ setze $y_0 = y_i$
  und deaktiviere ggf. Blätter)

→ gib y_0 als Lösung aus

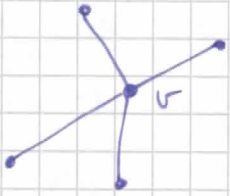
- Die Anfangslösung y_0, bzw y_i soll möglichst nah am Optimum (von $L(v_i)$) liegen, damit schlechte andere Zweige (Knoten) frühzeitig deaktiviert werden
- unten(v_i) soll möglichst nah am Optimum in $L(v_i)$ liegen damit v_i ggf. frühzeitig deaktiviert wird
- Was ist das „erfolgversprechendste Blatt" von $\mathcal{B}$?
  - best-fit-search: wähle Knoten mit der niedrigsten unteren Schranke
  - Tiefensuche: schont den Speicherplatzverbrauch

Bemerkung: Der B&B Baum $\mathcal{B}$ entspricht auch etwa dem Rekursionsbaum eines rekursiven Algorithmus (nicht polynomiell), wie im Fall der Dynamischen Programmierung. In der Dynamischen Programmierung führen die Überlappungen der Teilprobleme zu einem effizienten Algorithmus. Bei B&B hilft die frühzeitige Erkennung schlechter Lösungen, bzw. dass nicht jeder Zweig durchlaufen werden soll.

Lösungsmengen $\longleftrightarrow$ Teilprobleme

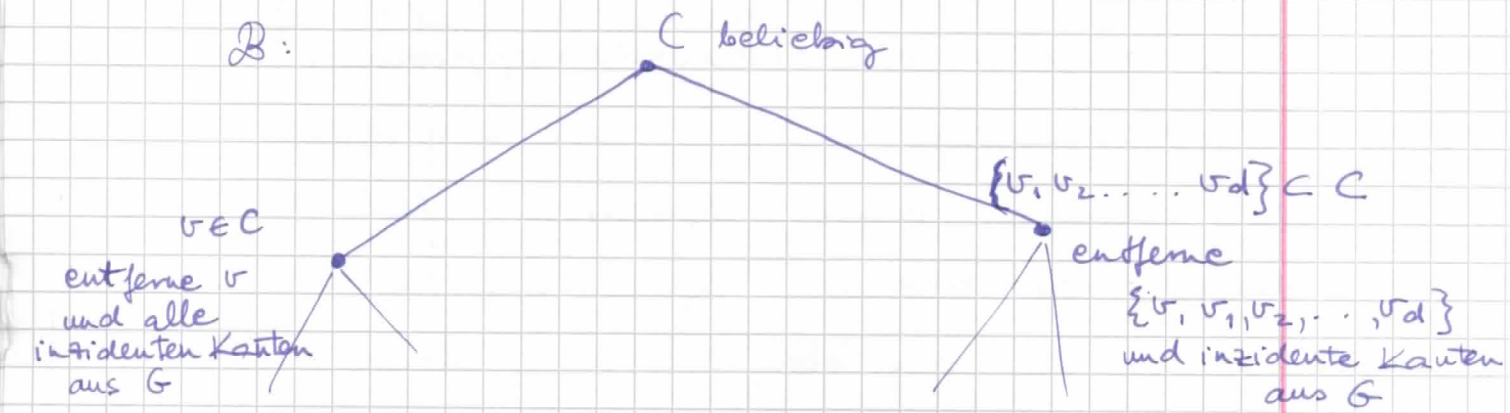
Beispiel 2. Ein besserer Branching-Operator für VERTEX COVER im ungerichteten Fall

- wähle einen Knoten $v \in V$ mit $\deg(v) \geq 3$ und Nachbarn $v_1 v_2 \dots v_d$ (d = d)



entweder $v \in C$ oder $\{v_1 v_2 \dots v_d\} \subset C$

$\mathcal{B}$:



- falls nur Knoten mit Grad ≤ 2 gibt, finde eine minimale Knotenüberdeckung

→ (ohne Bounding Schritte) hat dieser Algorithmus Worst-Case Laufzeit $O(1{,}38^n)$ (vielleicht auch mit...)

BB6.

b.) Branch & Bound für RUCKSACK

Eingabe: n Objekte mit Gewichten $g_1, g_2, \ldots, g_n$
und Werten $w_1, w_2, \ldots, w_n$
eine Gewichtsschranke G

Ausgabe: Finde eine Auswahl von Objekten $I \subseteq \{1, 2, \ldots, n\}$ mit maximalem Gesamtwert, so dass die Gewichtsschranke nicht überstiegen wird

Überlegungen:

- Welche Lösungsmengen (bzw. Teilprobleme) entsprechen den Knoten des B&B Baums?
  - $(\mathcal{F}, i)$ bezeichne alle Bepackungen so dass von den ersten i Objekten genau die Menge $\mathcal{F} \subseteq \{1, 2, 3, \ldots, i\}$ eingepackt wird
    (es gibt 2^{n-i} solche Teilmengen von $\{1, 2, \ldots n\}$)
    (Beachte: hier werden auch in jedem Knoten manche Objekte erzwungen, manche andere verboten)
- Wir haben hier ein Maximierungsproblem:
  in jedem Knoten von $\mathcal{B}$ werden wir eine obere Schranke $\text{oben}(v)$ berechnen; höchstens diesen Wert erreichen Bepackungen in $L(v)$.
- Wie finden wir eine obere Schranke $\text{oben}(v)$ für die Werte im Teilbaum von $v = (\mathcal{F}, i)$?
- Wie finden wir eine gute Anfangslösung y_0?

- Wie finden wir eine gute Lösung in $L(v) = L(\mathcal{F}, i)$?

Exkurs: Die Bestimmung einer oberen Schranke oben(v)

der Einfachheit halber sei oben(v) gesucht für das ganze Problem $v = (\mathcal{O}, 0)$

Wenn wir für RUCKSACK einen greedy Algorithmus entwerfen sollten, in welcher Reihenfolge würden wir die Objekte betrachten / in den Rucksack packen?

→ nach absteigendem Wert? (NEIN, siehe z.B. $w_i \sim 1$ aber $g_i = G$)

→ nach aufsteigendem Gewicht? (NEIN, siehe $g_i \sim 1$ aber $w_i = 0$)

→ Idee: sortieren wir die Objekte nach ihrem „Wert pro Kilo" $\frac{w_i}{g_i}$ und packen wir sie in absteigender Reihenfolge in den Rucksack:

$$\frac{w_1}{g_1} \geq \frac{w_2}{g_2} \geq \frac{w_3}{g_3} \geq \dots$$

Welchen Approximationsfaktor erreichen wir?

(∞ Approximation, schon mit $n=2$ Objekten; Warum?)

Fraktionale RUCKSACK: beliebige Bruchteile der Objekte dürfen eingepackt werden

LP Formulierung: (maximiere $\sum x_i w_i$

so dass $\sum x_i g_i \leq G$

$0 \leq x_i \leq 1$)

um eine obere Schranke zu bestimmen, relaxieren wir das Problem P (wir lassen auch nicht-Lösungen als Lösungen zu), so dass das relaxierte Problem P' effizient optimierbar ist. Das Maximum für P' kann nicht kleiner sein als für P, und ist somit eine obere Schranke.

Greedy Algorithmus für das fraktionale RUCKSACK

- sortiere die Objekte nach absteigendem Wert pro Kilo

$$\frac{w_1}{g_1} \geq \frac{w_2}{g_2} \geq \dots$$

- wenn $\sum_{i=1}^{k} g_i \leq G < \sum_{i=1}^{k+1} g_i$ dann

packe die Objekte $1, 2, \dots, k$ ein und noch ein Bruchteil x_{k+1} so dass

$$g_1 + g_2 + \dots + g_k + x_{k+1} \cdot g_{k+1} = G$$

Behauptung 1. Diese Lösung ist optimal für das fraktionale RUCKSACK.

(Wenn eine Lösung nicht die Form

$(1, 1, 1, \dots, 1, x, 0, 0, \dots, 0)$ hat, dann gibt es ein $x_k < 1$ und $x_{k+1} > 0$:

dann kann vom Objekt $k+1$ y Gewicht ausgeladen und vom Objekt k y Gewicht eingepackt werden ohne den Gesamtwert zu reduzieren.)

Behauptung 2: Der Maximumwert einer ganzzahligen Bepackung ist nicht größer als der Maximumwert einer fraktionalen Bepackung.

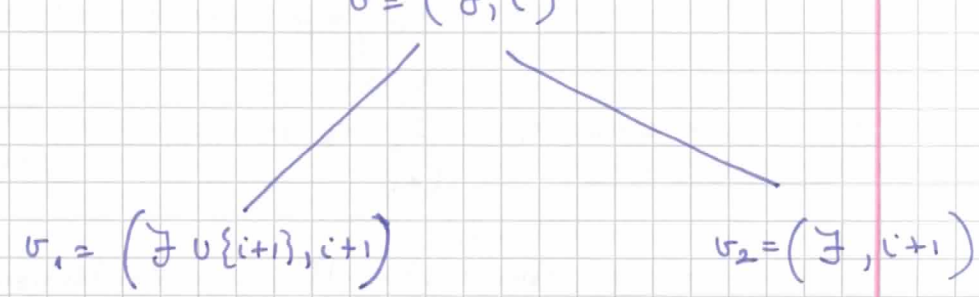
$$W_{max} \leq W_{max}^{frac}$$

(die beste ganzzahlige Bepackung ist auch eine fraktionale Bepackung, also höchstens so gut wie die beste fraktionale Bepackung)

$\Rightarrow$ die optimale fraktionale Lösung bestimmt also eine obere Schranke $\sum_{i=1}^{k} w_i + x_{k+1} w_{k+1} = W_{OPT}^{frac}$ für das Optimum von RUCKSACK.

Wir fassen zusammen (die Antworte auf unsere Fragen)

- jeder Knoten v von $\mathcal{B}$ entspricht einem Paar $v = (\mathcal{J}, i)$ für $1 \le i \le n$ und $\mathcal{J} \subseteq \{1,2,\dots,i\}$
  $L(v)$ ist die Menge aller Bepackungen, die von den ersten i Objekten genau die Teilmenge $\mathcal{J}$ enthalten
- für $v = (\mathcal{J}, i)$ erzeugt der Branching-Operator die Kinder



- um gute approximative Lösungen zu finden, wenden wir den approximativen dynamischen Programmier-Algorithmus an (PTAS mit Skalieren und Runden)

auf $\{i+1, i+2, \dots, n\}$ und $G - \sum_{k \in \mathcal{J}} g_k = G'$

- der (fraktionale) greedy Algorithmus bestimmt die obere Schranke $oben(v)$ für alle Lösungen in $L(v)$

$$oben(v) = \sum_{k \in \mathcal{J}} w_k + W^{frac,\,opt}_{\{i+1,\dots,n\},\,G'}$$

- als erfolgversprechendste Lösung $(\mathcal{J}, i)$ wird eine bisher wertvollste Bepackung gewählt

BB10.

Branch & Bound für RUCKSACK

→ anfänglich besteht $\mathcal{B}$ aus der aktivierten Wurzel $v_0 = (\emptyset, 0)$
und $oben(v_0) = W_{OPT}^{frac}$

→ eine Lösung I_0 mit Dynamischer Programmierung (PTAS)
wird berechnet mit Wert $W(I_0)$

→ WHILE es aktivierte Blätter in $\mathcal{B}$ gibt, DO

- wähle das Blatt $v = (\mathcal{J}, i)$ mit $\sum_{k \in \mathcal{J}} w_k$ maximal
(und i minimal)
- seien $v_1 = (\mathcal{J} \cup \{i+1\}, i+1)$ und $v_2 = (\mathcal{J}, i+1)$
$\mathcal{J}_1 = \mathcal{J} \cup \{i+1\}$ $\mathcal{J}_2 = \mathcal{J}$
- $oben(v_1) = \sum_{k \in \mathcal{J}} w_k + w_{i+1} + W_{\{i+2, \dots, n\}, G_{rem}}^{frac, OPT}$
- IF $oben(v_1) \leq W(I_0)$ deaktiviere v_1
ELSE sei I' Ergebnis des PTAS auf
$\{i+2, \dots, n\}$ und $G - \sum_{k \in \mathcal{J}_1} g_k$
und $I_1 = \mathcal{J} \cup \{i+1\} \cup I'$
IF $W(I_1) > W(I_0)$
sei $I_0 := I_1$
- $oben(v_2) = \sum_{k \in \mathcal{J}} w_k + W_{\{i+2, \dots, n\}}^{frac, OPT}$
- IF $oben(v_2) \leq W(I_0)$ - - - -
analog

→ gib $I_0 = (\mathcal{J}, n)$ mit $\sum_{k \in \mathcal{J}} w_k$ max aus