- Successive Shortest Path Algorithmus
- Matchings & Edmonds-Algorithm

<u>MinCostFlow-Problem</u>: (G, c, u, b)

Kosten in Residualgraphen G_f zu f:

VK: c

RK: $-c$

<u>1. Optimalitätskriterium:</u>

Sei f zul. Fluss in (G, c, u, b).

f ist kostenoptimal $\Longleftrightarrow$ G_f enthält keine Kreise neg. Gewichts

$\Rightarrow$ führt zum CycleCanceling-Algorithmus

Idee ähnlich zu Ford-Fulkerson-Alg für MaxFlow:

- starte mit zul. Fluss
- stelle sukzessive Optimalität her

Laufzeit: ganzz. Daten $\Rightarrow O(nm^2 C U)$

Verbesserung zu MinMeanCycleCancelling: $O(nm^2 \log n)$

---

2. Optimalitätskriterium

Def.: **Potenzial**: $\pi : V \to \mathbb{R}$

reduzierte Kosten bzgl. Potenzials π:

$$c^\pi(i,j) := c(i,j) - \pi(i) + \pi(j)$$

Bem.: reduzierte Kosten in G_f:

VK (i,j): $c^\pi(i,j) = c^\pi(i,j)$

RK (j,i): $c^\pi(j,i) = -c^\pi(i,j)$

Satz: Sei f zulässiger Fluss in (G,c,u,b).

f kostenoptimal $\Leftrightarrow$ $\exists$ Potenzial π mit

$$c^\pi(i,j) \geq 0 \quad \forall (i,j) \in E(G_f)$$

Idee für Algorithmus:

- Starte mit "Pseudofluss" f, für den ein optimales Potenzial existiert

$\underbrace{}$

d.h. eines mit $c^{\pi}(i,j) \geq 0 \; \forall E(G_f)$

- Mache f sukzessive zu einem zul. Fluss in (G,c,u,b) ohne die Optimalitätsbedingung zu verletzen.

Def.: Pseudofluss: $f: E \to \mathbb{R}^+$, $f(e) \leq u(e)$

Exzess/Defizit von Knoten v:

$$e(v) := b(v) - \sum_{e \in \delta^+(v)} f(e) + \sum_{e \in \delta^-(v)} f(e)$$

also falls $e(v) = 0 \; \forall v \in V$, so ist f ein zulässiger Fluss in (G,c,u,b).

Lemma: Sei f ein Pseudofluss in (G, c, u, b) und π ein optimales Potenzial zu f.
Seien $d(v)$ die kürzesten Weglängen von einem Knoten s zu v bzgl. c^{π} in G_f.

$\Rightarrow$ $\pi' := \pi - d$ ist auch ein optimales Potenzial zu f und

$$c^{\pi'}(i,j) = 0 \quad \forall (i,j) \text{ auf einem kürzesten}$$
s-v-Weg.

Korollar: Augmentieren eines optimalen Pseudoflusses entlang von kürzesten Weg P erhält die Optimalität.
(G_f verändert sich nur entlang des Weges P und

$c^{\pi'}(i,j) = -c^{\pi'}(j,i) = 0 \quad \forall\, (i,j) \in P)$

- **while** es gibt Knoten v mit $b'(v) \neq b(v)$ ($e(v) \neq 0$) **do**
  - wähle Knoten s mit $b(s) - b'(s) > 0$
  - **if** es gibt einen Knoten t mit $b(t) - b'(t) < 0$ der von s in G_f erreichbar ist
    - **then** wähle einen solchen Knoten t
    - **else** **return** // es gibt keinen b-Fluss
  - berechne in G_f einen s,t-Weg W mit minimalen Kosten
  - setze $\gamma := \min\{\min_{e \in E(W)} u_f(e),\ b(s) - b'(s),\ |b(t) - b'(t)|\}$
  - augmentiere f entlang W um γ
  - $b'(s) := b'(s) + \gamma,\ b'(t) := b'(t) - \gamma$

<u>Bem.:</u> Vorsicht bei neg. Zykel bzgl. c in G: Augmentiere zunächst entlang negativer Zykel, bis auf jedem wenigstens eine Kante gesättigt ist.

Optimale Matchings

Eine Anwendung:

Chinese Postman Problem

<u>Instanz:</u> unger. Graph $G=(V,E)$

Aufgabe: Finde Tour, die alle Kanten wenigstens einmal benutzt und möglichst kurz ist.
(z.B. Briefträger, Müllabfuhr)

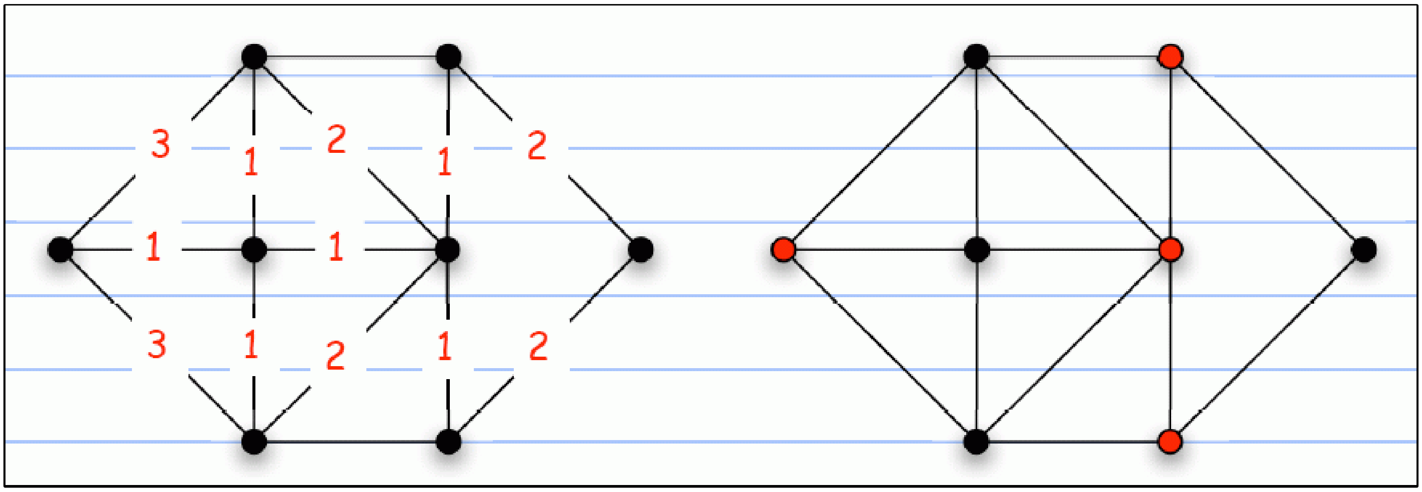
Falls G eulersch (also alle Knoten geraden Grad), so ist jede Euler-Tour optimal ✓

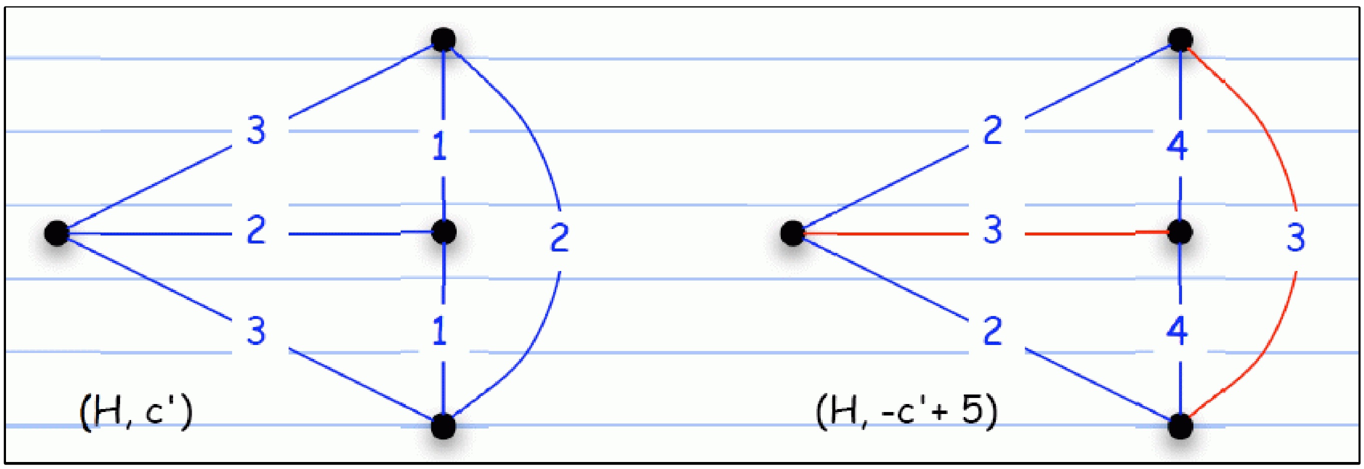
Falls G nicht eulersch: jede Tour benutzt Kanten mehrfach: minimiere Kosten dieser Kanten.
mehrfach benutzte Kanten sind Wege zwischen Knoten ungeraden Grades!
⇒ stelle diese Wege durch Kanten zwischen Knoten ungeraden Grads dar, deren Gewicht die entsprechenden kürzesten Weglängen sind!
⇒ Finde kostenminimales Matching auf diesen Kanten!

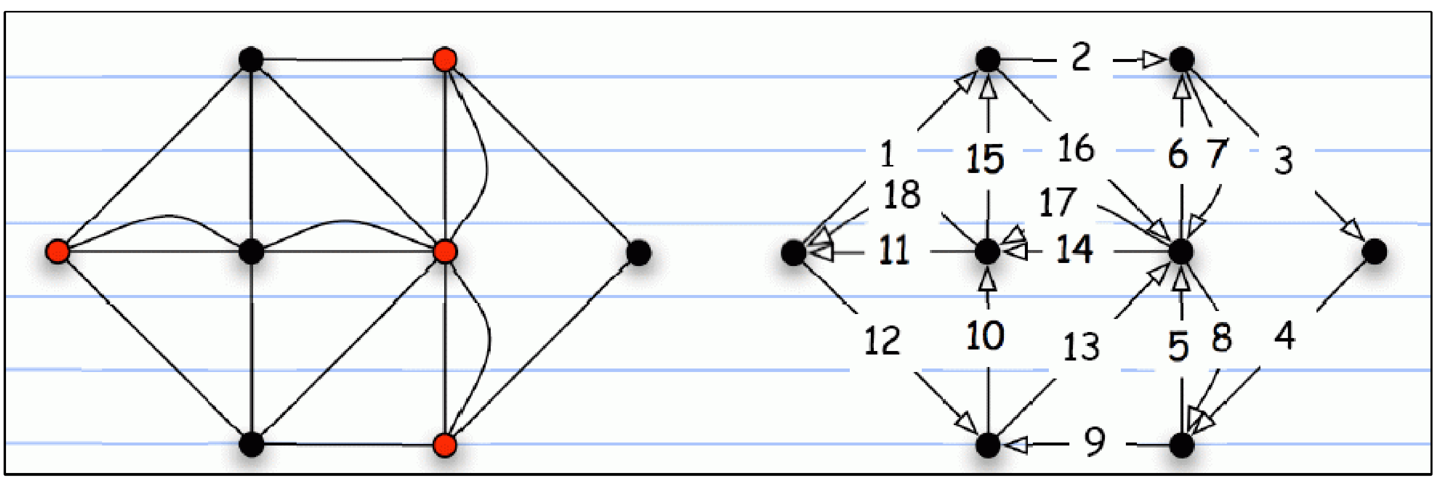
- Input
  - Ungerichteter Graph G, Kantengewichte $c(e) \geq 0$
- Output
  - Eine Tour minimaler Länge, die jede Kante mindestens einmal durchläuft
- Methode
  - <u>if</u> es gibt keine Knoten mit ungeradem Grad
    - <u>then</u> <u>return</u> Euler Tour von G
  - ermittle alle Knoten mit ungeradem Grad \\ gerade Anzahl
  - betrachte vollständigen Graphen H auf diesen Knoten
  - Kante $e \in E(H)$ bekommt Kantenbewertung $c'(e)$:= Länge eines kürzesten Weges in G zwischen den Endpunkten von e
    // kann wegen $c(e) \geq 0$ min Kürzeste-Wege Algorithmen berechnet werden
  - berechne ein maximal gewichtetes Matching M in H bzgl. Kantengewichten $-c'(e) + N$ (N große Konstante

- für jede Kante $e \in M$ nehme einen zugehörigen kürzesten Weg W mit Länge $c'(e)$ und füge die Kanten aus W zu G hinzu (Mehrfachkanten möglich)

  // der entstehende Graph G' ist Eulersch mit minimalen zusätzlichen Kosten







---

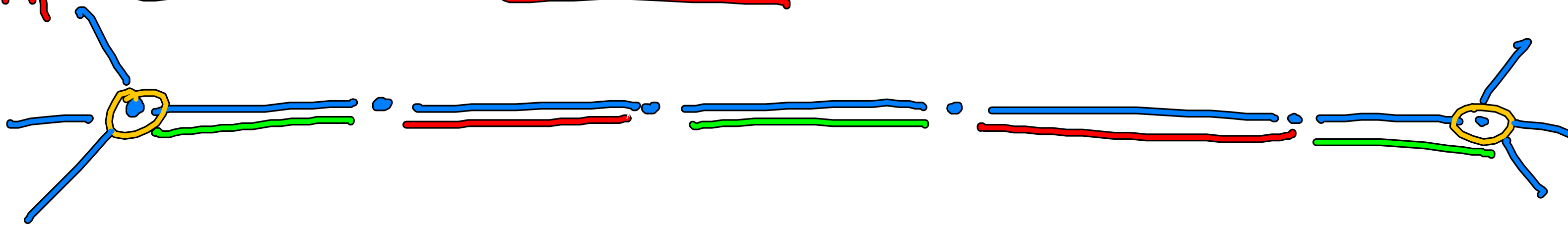
zunächst: **Kardinalitätsmaximale Matchings**

Satz: Sei G ungr., M Matching

M maximal $\iff$ $\nexists$ M-augmentierende Weg in G gilt

$\hookrightarrow$ M-alternierende Weg mit exponierten Endknoten

$|M| = 2$



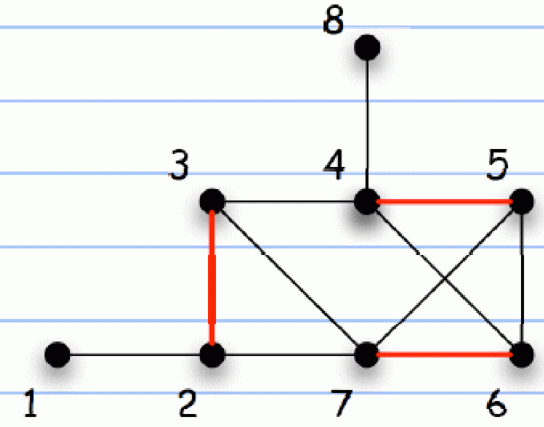
$|M'| = 3 > |M|$

$\Longrightarrow$ Algo:
- Starte mit leerem Matching M
- finde M-augmentieren Weg

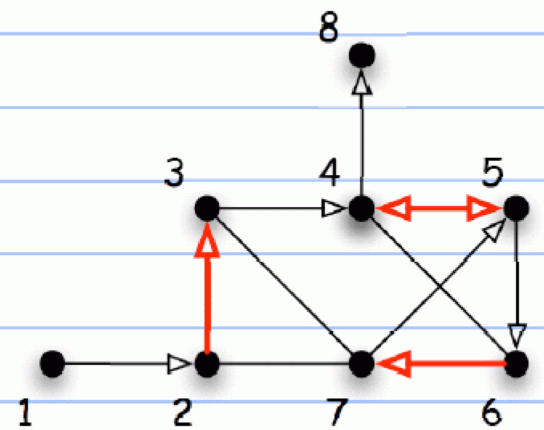
Frage: Wie findet man M-augmentierende Wege?

- bipartite Graph: leicht! (Aufgabe 47)
- allgemeine Graphen: schwieriger...

Bsp.:

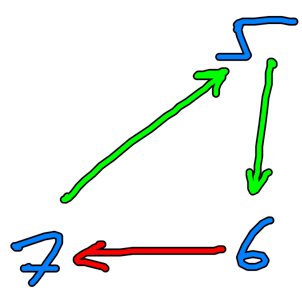


Erster Versuch eines augmentierenden Weges: 1-2-3-4-5-6-7-5-4-8

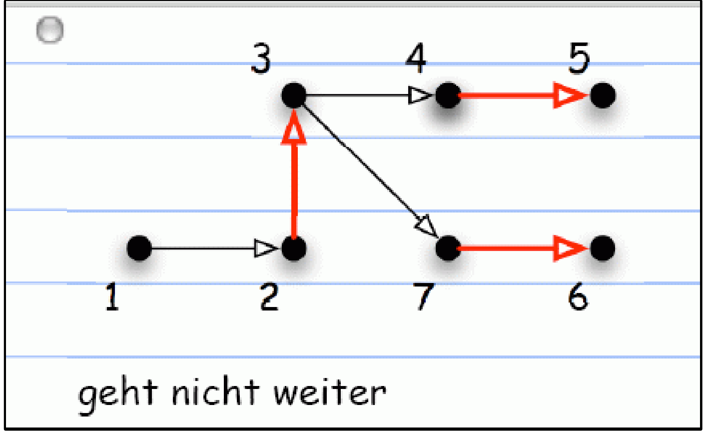


Ist nicht elementar, also nicht M-augmentierend. Er enthält ungeraden Kreis

Problem: ungerader Kreis

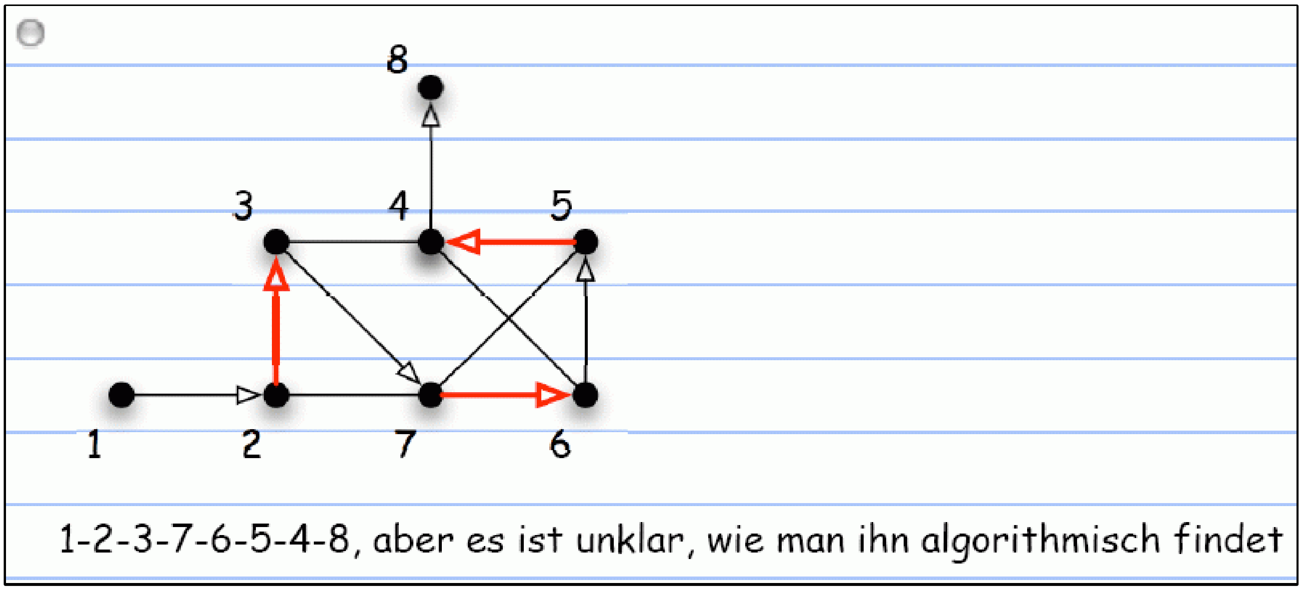


2. Versuch: BFS:



geht nicht weiter

ABER:



1-2-3-7-6-5-4-8, aber es ist unklar, wie man ihn algorithmisch findet

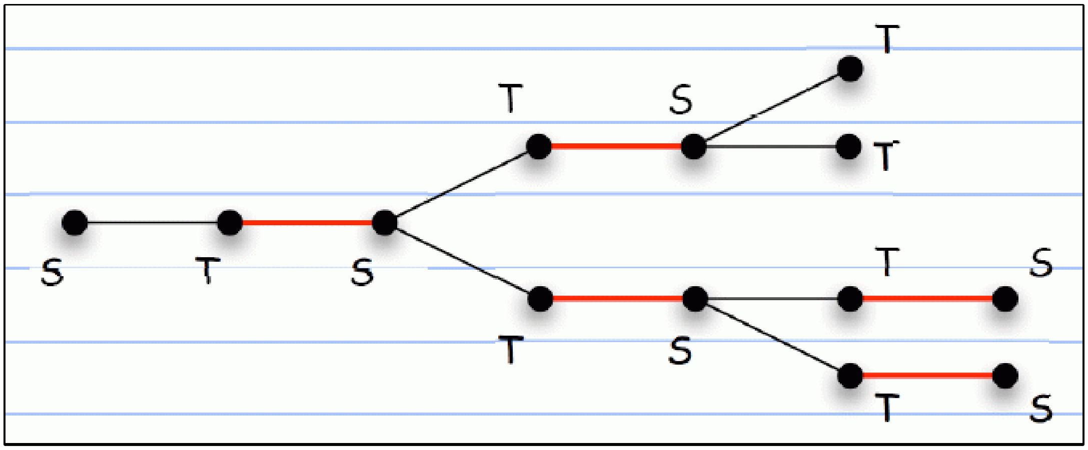
Lösung durch Edmonds 1965:

„Paths, Trees & Blossoms"

Idee: Finde einen M-alternierenden Baum
mit
- exponierter Wurzel
- exponiertem Blatt

- Falls man ein exponiertes Blatt findet ✓

• Falls nicht, benenne Knoten mit S, T wie folgt:

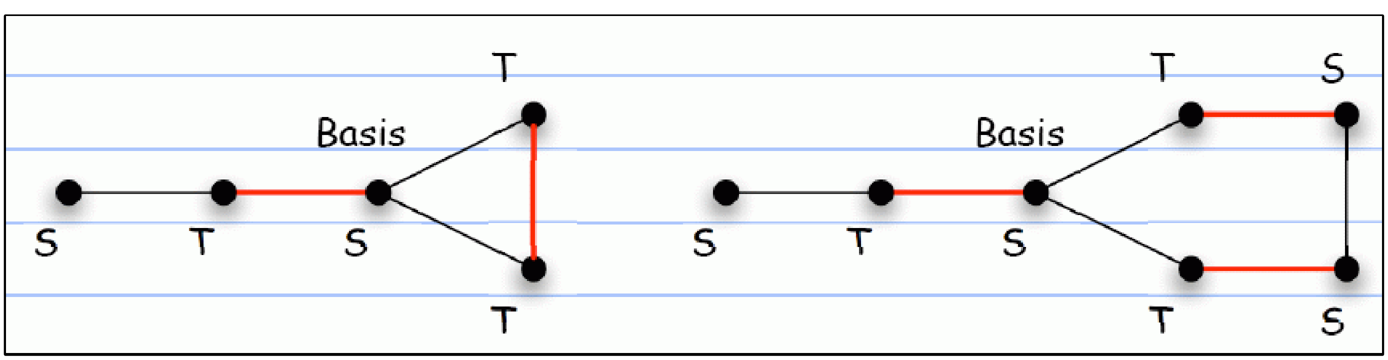


Def.: Eine <u>Blüte</u> ist ein elementarer Kreis B in G, der entsteht durch

- $e \in M$ zwischen zwei T-Knoten
- $e \notin M$ zwischen zwei S-Knoten

Die <u>Basis</u> einer Blüte B ist der exponierte Knoten auf dem zugeh. Kreis B.

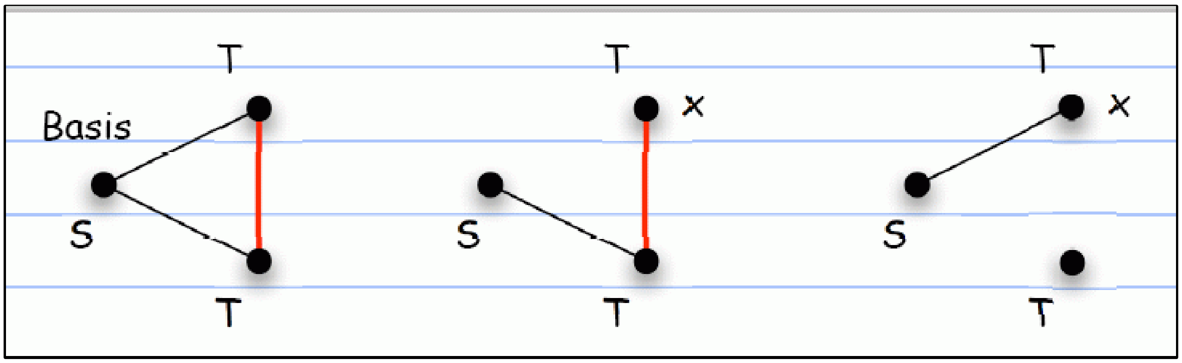
Der <u>Stengel</u> einer Blüte ist der Weg von der Wurzel zur Basis.



Für Blüte B gilt:

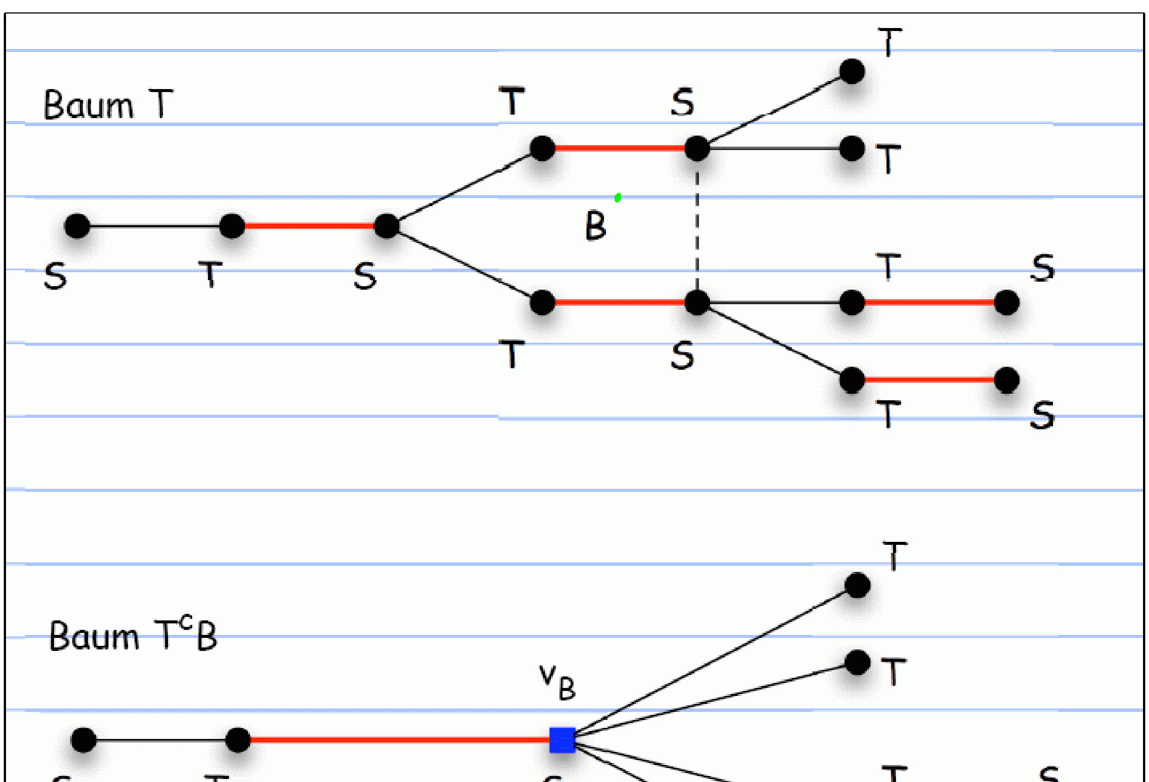
• zu jedem Knoten auf dem Kreis der Blüte gibt es zwei Wege von der Basis aus:

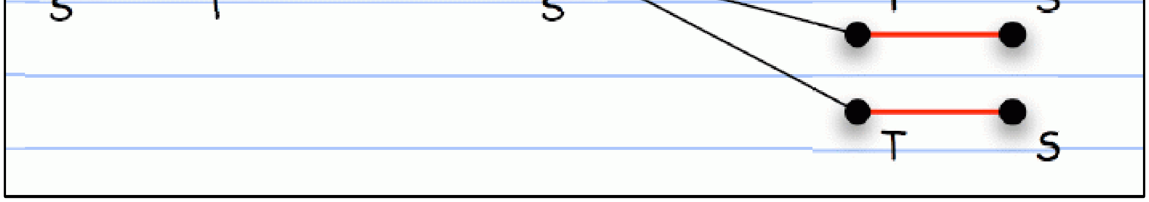
  • 1, der mit Matching-Kante endet
  • 1, der mit Nicht-Matching-Kante endet



⇒ Blüten kodieren "Alternative" bei der Suche nach M-augmentierenden Wegen.

Kontraktion einer Blüte B: $G \rightarrow G^cB$





Gebe v_B den Label S.

Nach Kontraktion 2 Möglichkeiten:

1) Wir können den Baum fortsetzen (es gibt Knoten, die noch nicht im Baum sind und zu B "alternierend-adjazent" sind)

2) Es gibt keine zu B "alternierend-adjaz." Knoten, die noch nicht im Baum sind.

<u>Im Fall 1:</u> weitermachen: Es gibt einen Weg durch die Blüte, der jeden Weg von der Wurzel zu einem Blatt M-alternierend macht (die eine Alternative, die in der Blüte kodiert ist)

<u>Im Fall 2:</u> Durch Wege im Baum kann das Matching nicht augmentiert werden.

$\Rightarrow$ Lösche Baum & inzidente Kanten aus G und starte neuen Baum.

Def.: "Ungarischer Baum": Maximaler (d.h. nicht erweiterbarer) M-alternierender Baum, der keine Blüte oder exponierten Knoten enthält.

- Input
  - Ungerichteter Graph G
- Output
  - Ein Matching M maximaler Kardinalität
- Methode
  - Initialisierung
    - setze $G' := G$ und $M' := M_0$ (M_0 beliebiges Matching, $M_0 = \emptyset$ möglich)
  - Hauptschleife
    - <u>repeat</u>
      - konstruiere M'-alternierenden Baum T in G' // zum Beispiel mit Breitensuche
      - <u>case</u>
        - T enthält M'-augmentierenden Weg
          - konstruiere zugehörigen M-augmentierenden Weg W in G gemäß Lemma 7.5
          - augmentiere M entlang W
          - Setze $G' := G$ und $M' := M$ // beginne wieder in G mit größerem Matching
        - T enthält Blüte B
          - kontrahiere B
          - setze $G' := G^{c}B$ und $M' := M' - E(B)$
        - T ist ungarisch
          - setze $G' := G' - V(T)$ // $V(T)$ und inzidente Kanten wegnehmen
      - end case
    - <u>until</u> $G' = \emptyset$ <u>or</u> es gibt keine exponierten Knoten
  - Konstruktion des Matching M aus M' // M' ist maximal in G'
    - konstruiere M rückwärts aus M' durch Expansion der kontrahierten Blüten (Schachtelung mö
    - <u>return</u> M

Formaler Korrektheitsbeweis siehe z.B. Möhring-Skript.