Gerüste

*Sei G(V,E) ein ungerichteter Graph. Ein Baum H(W, F) heißt ein **Gerüst** von G, wenn H ein Teilgraph von G ist und alle Ecken von G enthält (wenn also gilt $F \subseteq E$ und $W = V$). Wenn H ein Gerüst von G ist sagt man auch: „G wird von H aufgespannt“.*

Ein Graph G besitzt genau dann ein Gerüst, wenn er zusammenhängend ist.

(Satz von Caley) *Der vollständige Graph K_n mit n Ecken hat $\tau(K^n) = n^{n-2}$ verschiedene (nicht notwendig nichtisomorphe) Gerüste.*

***Algorithmus von Kruskal**: Eingabe: Eine Menge E der Kanten mit ihren Längen. Ausgabe: Eine Teilmenge F der Kantenmenge.*

1. *Nummeriere die Kanten $e_1, \dots, e_{|E|}$ nach steigender Länge. Setze $F := \emptyset$.*
2. *Für $i := 1, \dots, |E|$:*
   Falls $F \cup \{e_i\}$ nicht die Kantenmenge eines Kreises in G enthält, setze $F := F \cup \{e_i\}$.

Gerüste

Praktische Durchführung des Tests auf einen Kreis

- *Vor Beginn des Algorithmus werden alle Ecken von 1 bis $|V|$ durchnummeriert.*
- *Eine Kante darf genau dann zu F hinzugefügt werden, wenn ihre Endecken unterschiedlich nummeriert sind.*
- *Wenn eine Kante zu F hinzugefügt wird, seien α und β mit $\alpha < \beta$ die Nummern ihrer Endecken. Vor dem Test der nächsten Kante wird die Nummerierung aller bisher mit β nummerierten Ecken auf α gesetzt.*

Am Ende des Algorithmus tragen dann alle Ecken die Nummer 1.

***Greedy-Algorithmus**: Eingabe: Eine (endliche) Menge E. Ausgabe: Eine Teilmenge F der Kantenmenge.*

1. *Ordne die Elemente von E nach steigender [fallender] Bewertung. Setze $F := \emptyset$.*
2. *Für jedes Element $e \in E$ (in der gerade festgelegten Reihenfolge):*
   Falls $F \cup \{e_i\}$ die Greedy (gierige)-Bedingung erfüllt, setze $F := F \cup \{e\}$.

Gerüste

Praktische Durchführung des Tests auf einen Kreis

Ein Disjoint Set (Union-Find-Datenstruktur) hält eine Sammlung $S = \{S_1, \ldots, S_k\}$ von disjunkten Mengen.

- *Die Mengen sind dynamisch, d.h. Sie können sich mit der Zeit ändern.*
- *Jede Menge S_i wird von einem Stellvertreter identifiziert, der Element dieser Menge ist.*

Operationen:

- ***Make-Set(x)**: Erstellt eine neue Menge, dessen einziges Element und Vertreter x ist. (Initialisiert die Struktur)*
- ***Find-Set(x)**: Gibt den eindeutigen Repräsentanten der Menge S_i zurück, die x als Element enthält.*
- ***Union(x,y)**: Vereinigt S_i mit Element x und S_j mit Element y in eine neue Menge S_k und entfernt S_i und S_j aus der Sammlung S.*
  Es gilt dann Find-Set(x) == Find-Set(y).
  Vorher gilt Find-Set(x) =/= Find-Set(y).

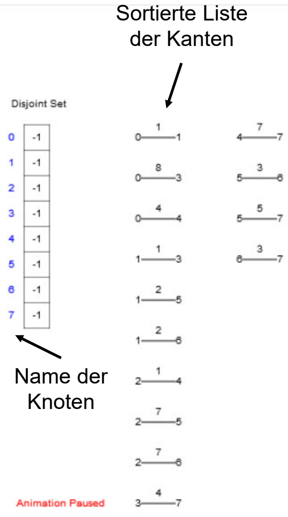
Gerüste

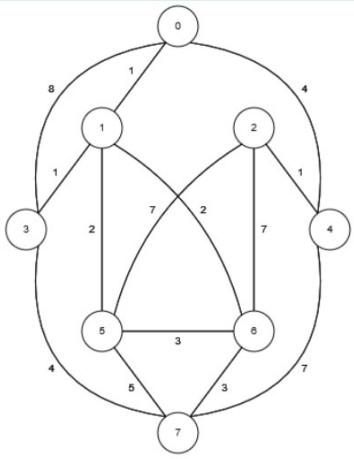
Implementierung der Disjoint Set als Baum

Bei der Wurzel steht die Anzahl der enthaltenen Elemente als negative Zahl. Bei den anderen Elementen steht die Wurzel bzw. der „link zur Wurzel“.

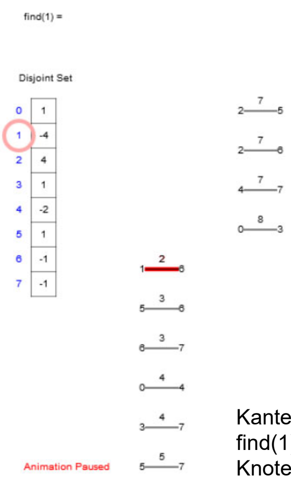
- ***Make-Set(x)**: Trägt -1 bei jedem Element als Wurzel ein (Initialisierung: jede Menge Senthät genau ein Element)*
- ***Find-Set(x)**: Gibt den Repräsentanten zurück. Falls x den Eintrag k < 0 hat, ist x der Repräsentant, sonst ist der Eintrag die Wurzel (bzw. der link zur Wurzel). In jedem Set gibt es nur ein Element mit negativem Wert!*
- ***Union(x,y)**: Sei o.B.d.A. a der Repräsentant der Menge S_i mit Element x und b der Repräsentant der Menge S_j mit Element y und der Eintrag bei x sei kleiner als der Eintrag bei y. Dann ist x der Repräsentant der neunen Menge S_k und Erhält als Eintrag die Summe der Einträge der beiden Wurzeln x und y. Alle Elemente in S_j erhalten als Eintrag x bzw. ein link auf x.*

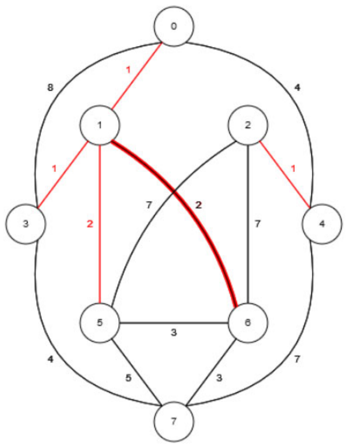
Gerüste





Gerüste





Kante 1-5 mit Kosten 2 wird behandelt:
find(1) = 1 find(5) = 1
Knoten in selber Menge, Kante wird nicht Aufgenommen (da sonst ein Kreis entsteht!)

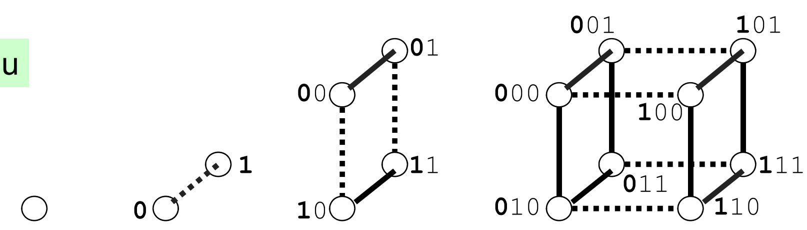
Hypercube-Routing

Hier spezielle ParallelrechnerTopologie Hyperwürfel (Hypercube):

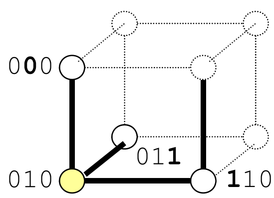
- Ein (Boolescher) **Hyperwürfel** besteht aus 2^n Knoten.
- Jeder Knoten wird mit einem nBitstring nummeriert.
- Zwei Knoten $(i_0 \ldots i_{n-1}), (j_0 \ldots j_{n-1}) \in \{0,1\}^n$ sind genau dann **miteinander verbunden**, wenn sie sich **in genau einem Bit unterscheiden**.

Routing im Hypercube

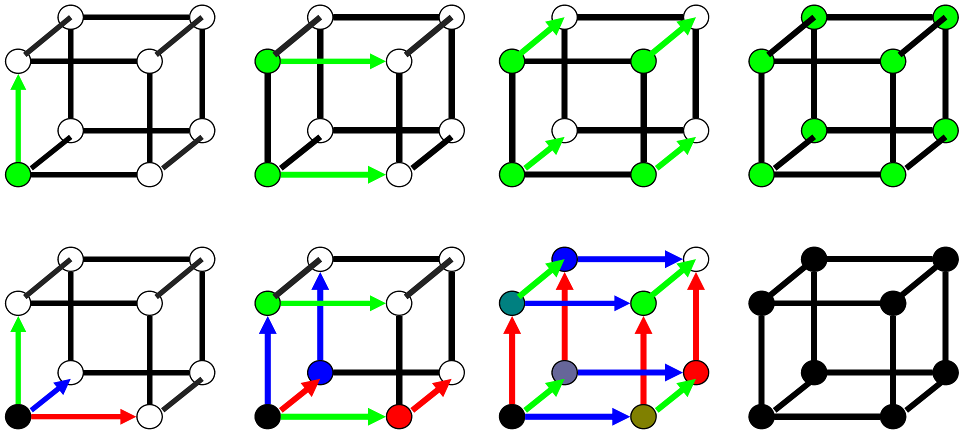
Aufbau



Adressierung

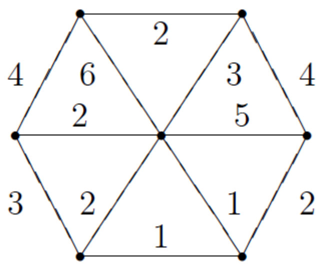


Routing im Hypercube



Aufgaben

1. Bestimmen Sie ein Minimalgerüst des Graphen



Aufgaben

2. Die Fluggesellschaft Graphic-Airlines macht außerhalb der Saison auf allen ihren Strecken Verluste. Sie möchte den Betrieb einschränken, indem sie manche Strecken vorübergehend einstellt. Es soll aber noch jeder Flughafen von jedem anderen aus erreichbar sein, notfalls mit Umsteigen. Aus der Landkarte kann man die Verluste für die einzelnen Strecken entnehmen. Sie sollen nun zwei Lösungen erarbeiten. Identifizieren Sie dazu das jeweilige Problem und verwenden einen geeigneten Algorithmus, um das Problem zu lösen. Vergleichen Sie abschließend die beiden Lösungen miteinander und machen Sie der Fluggesellschaft einen Vorschlag. Hier nun die jeweiligen Anforderungen für eine der beiden gesuchten Lösungen:
   a) Die Gesamtkosten für den Betrieb des Netzes sollen möglichst niedrig sein.
   b) In Eulerstadt ist die Zentrale. Alle Flughäfen sollen von dort aus mit möglichst geringen Verlusten erreichbar sein.

Aufgaben