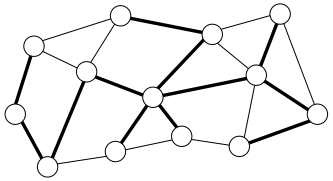
Verteilte Spannbaumkonstruktion?

- Gegeben: Zusammenhängendes Netz (mit bidir. Kanten)
- Alle Knotenidentitäten seien verschieden
- Bestimmung eines spannenden Baumes ist oft wichtig:



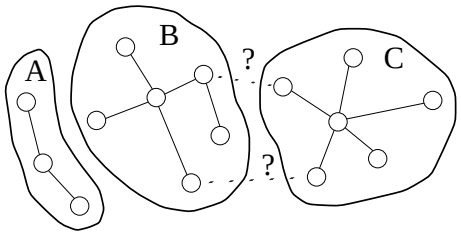
- z.B. um entlang dieses Baumes Information zu verteilen und einzusammeln
- Routing ist wesentlich einfacher, wenn Zyklen ausgeschlossen sind

---

Problem: - Wer initiiert die Spannbaumkonstruktion?
- Basteln alle Initiatoren am gleichen Baum?
(Problem: Zyklenbildung trotz Dezentralität vermeiden!)

- *Lösung 1*: Election, Gewinner startet Echo-Algorithmus
(bzw. Variante Echo-Election: Die Echos bilden bereits einen Spannbaum)

- *Lösung 2*: Dezentral werden Fragmente gebildet, die wachsen und sich nach und nach vereinigen



- Initial: Jeder Knoten ist ein Fragment
- Algorithmus von Gallager, Humblet, Spira (GHS) 1983 mit 2e + 5 n log n Nachrichten (im Detail nicht ganz einfach, hier nicht weiter behandelt)
- Problem: Sparsam mit Nachrichten umgehen!

Election und Spannbaumkonstruktion

Beh.: Election und Spannbaumkonstruktion sind in allgemeinen Netzen vergleichbar schwierige Probleme:

Präziser: Sei C_e die Nachrichtenkomplexität des Election-Problems, und C_t diejenige des Spannbaum-Problems:

(a) Es gilt für C_t: $C_t \leq C_e + 2e$ (wg. obiger "Lösung 1")

(b) Es gilt für C_e: $C_e \leq C_t + O(n)$ (wg. Kplx. Baumelection)

Interpretation:

(a) Hat man mittels Election einen "leader" bestimmt, lässt sich ein eindeutiger Spannbaum einfach ermitteln

(b) Hat man einen Spannbaum, dann lässt sich ein "leader" einfach (d.h. effizient) bestimmen

Hierbei wird 2e bzw. O(n) als "klein" gegenüber C_e und C_t angesehen

---

Aus der unteren Schranke $\Omega(e + n \log n)$ für die Nachrichtenkomplexität des Election-Problems folgt aus (b):

Das Spannbaum-Problem hat eine Nachrichtenkomplexität von mindestens $\Omega(e + n \log n)$

Denn sonst: Konstruiere den Spannbaum effizienter und löse mit zusätzlichen O(n) Nachrichten das Election-Problem!

--> Der genannte GHS-Algorithmus ist grössenordnungsmässig optimal!

Anonyme Netze

- Keine Knotenidentitäten
  - bzw.: alle (oder mehrere) *identische* Identitäten
  - bzw.: ggf. vorhandene Knotenidentitäten werden nicht benutzt

- Frage: Was geht dann noch? (Insbes. Symmetriebrechung!?)

- Es gilt: Falls Election in anonymen Netzen geht, dann können die Knoten individuelle Namen bekommen

...und *alles* geht wie gehabt!

  - 1. Idee: Leader gibt sich einen (neuen) Namen; restliche Knoten führen unter sich eine neue Election durch
  - 2. Idee: Leader fragen, wie man heissen soll; dieser denkt sich Namen aus
  - Konkretisierung bzw. Variante der 2. Idee:
    - es sei also die Existenz eines "Leaders" angenommen
    - dieser kann mit dem Echo-Algorithmus einen Spannbaum bilden (das klappt auch bei anonymen Knoten!)
    - die Echos melden ("rekursiv") die Grösse (= Anzahl der Knoten) jedes gebildeten Unterbaumes zurück
    - jeder Knoten merkt sich für jede "ausgehende" Kante die Grösse des daran hängenden Unterbaumes
    - nach Konstruktion des Baumes wird (ausgehend vom Initiator als Wurzel) der Baum durchlaufen - dabei wird jedem Unterbaum ein disjunktes Intervall natürlicher Zahlen passender Grösse zugeordnet

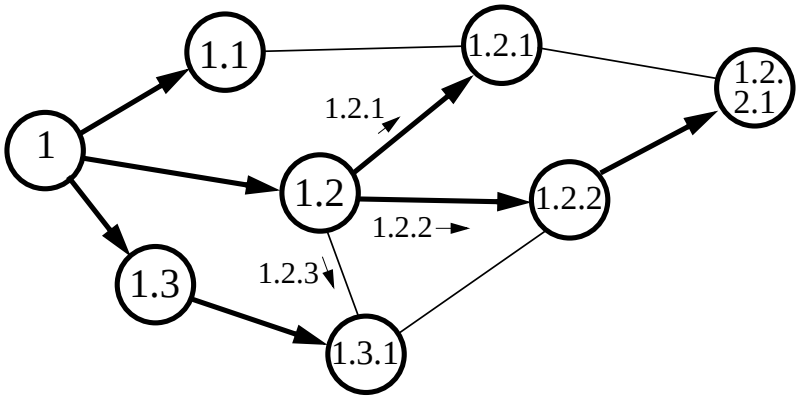
---

- Obiger Satz lässt vermuten, dass Election in anonymen Netzen *nicht* geht, sonst wäre Anonymität kein *grundsätzliches* Problem!

De-Anonymisierung

Es geht auch direkter ohne explizite Baumkonstruktion,
indem der Leader das Netz mit verschieden benannten Nachrichten
flutet und jeder Knoten die Identität der ersten Nachricht übernimmt,
die ihn erreicht - dazu numeriert ein Knoten seine Kanäle bzw. Nachrichten
durch und konkateniert seine eigene Identität zu dieser Nummer

==> Alle Nachrichten und damit alle Knoten heissen verschieden!



- Einziger (!) Initiator gibt sich selbst einen Namen “1”

- Jeder Prozess numeriert seine Ausgangskanäle 1,...,k

- Jede von Prozess X über Kanal j versendete Nachricht bekommt zusätzlich den Namen “X.j” dazugepackt

- Ein (noch) anonymer Prozess nimmt den Namen der ersten Nachricht als seinen Namen

Fragen:
- wann ist das Verfahren beendet?
- wie erfährt der Initiator dies?

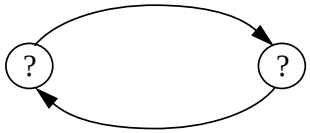
Election in anonymen Netzen?

- In anonymen Netzen geht manches (z.B. Election?) nicht mehr mit deterministischen Algorithmen
  --> randomisierte Verfahren helfen gelegentlich!

- Manches geht noch, wenn wenigsten die Knotenzahl bekannt ist (aber auch das hilft nicht immer!)

Satz: *Es gibt keinen stets terminierenden Election-Algorithmus für anonymen Ringe*

selbst wenn die Ringgrösse den Knoten bekannt ist



- hier für Ringgrösse 2
- jeder Knoten befindet sich (bzgl. des Election-Algorithmus) in einem bestimmten Zustand z

Bew.: - Betrachte *Konfigurationen* (hier Quadrupel aus den Zuständen der beiden Knoten und Kanäle)

- Kanalzustand = Nachrichten, die unterwegs sind

- Anfangkonfiguration des Algorithmus ist *symmetrisch*: (z,z,∅,∅), wegen Anonymität

- Alle lokalen Algorithmen sind definitionsgemäss identisch --> Knoten können sich jeweils gleich (quasi "zum gleichen Zeitpunkt") verhalten

- Konfigurationsfolge kann daher aus lauter symmetrischen Konfigurationen bestehen: (z,z,∅,∅) --> (z',z',k,k) --> ... --> etc.

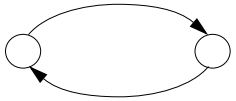
- Falls die Folge endlich ist, könnte der Endzustand symmetrisch sein --> keine Symmetriebrechung möglich!

Ein probabilistischer Election-Algorithmus

```
state := undecided;
while state = undecided do
{ mine := random(0,1);
  send <mine> to neighbor;
  receive <his>;
  if (mine,his) = (1,0) then state := win;
  if (mine,his) = (0,1) then state := lose;
}
```

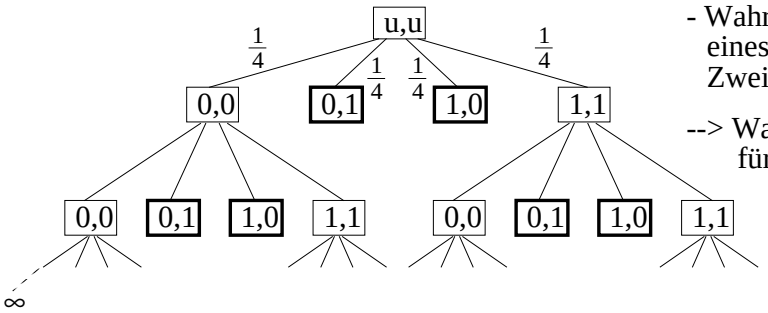
zufälliges Ergebnis 0 oder 1

- hier: *anonyme* Ringe der Grösse 2



- jeder Knoten führt den gleichen nebenstehenden Algorithmus aus
- Verallgemeinerung auf grössere Ringe?

- Geht es nicht auch mit einem *deterministischen* Algorithmus?
- Beachte: *"korrekt" ≠ "korrekt mit Wahrscheinlichkeit 1"*!
  - "korrekt" heisst: Algorithmus *hält stets* und liefert richtiges Ergebnis
  - obiger Algorithmus ist aber (in diesem Sinne) *nicht korrekt*, da es nicht terminierende Konfigurationsfolgen wie etwa
    - (0,0) --> (0,0) --> (0,0) --> (0,0) --> (0,0) --> (0,0) .. . oder
    - (0,0) --> (1,1) --> (0,0) --> (1,1) --> (0,0) --> (1,1)... etc. gibt!
  - alle diese Folgen haben aber die Dichte 0 (wenn "random" gut genug ist...)
    ==> Algorithmus terminiert mit Wahrscheinlichkeit 1
    ==> Algorithmus ist "korrekt mit Wahrscheinlichkeit 1"!



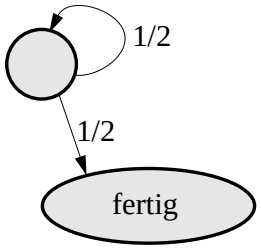
- Wahrscheinlichkeit eines unendlichen Zweiges = 0

--> Wahrscheinlichkeit für leader = 1

Mittlere Nachrichtenkomplexität?

- Beachte: Keine Schranke für *worst-case* Komplexität!

- Die mittlere ("erwartete") Nachrichtenkomplexität lässt sich mittels *Markow-Ketten* leicht ermitteln

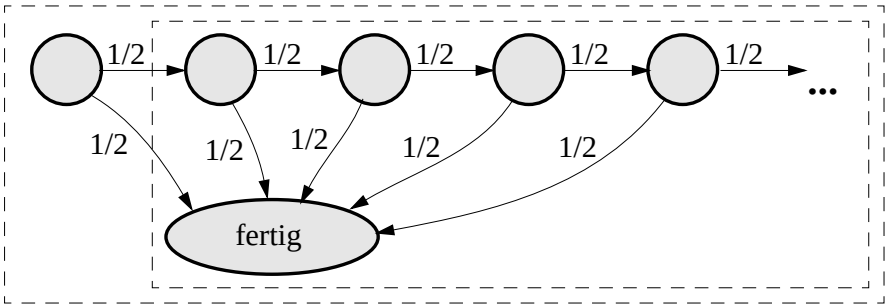


- wie hoch ist die erwartete Weglänge W, wenn mit den angegebenen Wahrscheinlichkeiten (hier jeweils 1/2) zum Folgezustand verzweigt wird?

- ist dieser Ansatz (gewichtete Summe) korrekt?

$$1\,\frac{1}{2} + 2\,\frac{1}{4} + 3\,\frac{1}{8} + 4\,\frac{1}{16} + \ldots + \quad = ?$$

- Durch "Aufbröseln" der Schleife ergibt sich diese Darstellung:



- hier ist man mit Wahrscheinlichkeit 1/2 nach einem Schritt fertig, oder es liegt (nach einem Schritt) wieder die gleiche Situation vor!

- daraus ergibt sich der *Rekursionsansatz* W = 1/2 + 1/2 (1+W)

- dies liefert W=2 als Lösung

- Bem.: Hinweise zur Varianz etc. findet man in entsprechenden Mathematik-Lehrbüchern

Probabilistische Algorithmen

- Der klassische "totale" Korrektheitsbegriff von Algorithmen kann auf zweierlei Weise abgeschwächt werden:

---

1. Sogenannte *Las Vegas-Algorithmen:*

- Abschwächung der Terminierungsforderung
- also: "Partiell korrekt und Terminierung mit *Wahrscheinlichkeit* 1"
- beachte: die (worst-case) Laufzeit solcher Algorithmen ist unbeschränkt!
- Beispiel: obiger Election-Algorithmus für anonyme Ringe

---

2. Sogenannte *Monte Carlo-Algorithmen:*

- Abschwächung der partiellen Korrektheit
- "terminiert stets, ist aber nur mit Wahrscheinlichkeit $p < 1$ partiell korrekt"
- also: $\exists$ *Restwahrscheinlichkeit* $\varepsilon = 1-p > 0$, dass das Ergebnis falsch ist!
- nur verwenden, wenn:
  - ε sehr *klein* ist (oft: als Parameter des Algorithmus, etwa abhängig von der Laufzeit und damit "beliebig klein" *wählbar*)
  - dadurch deutliche Vorteile erzielbar (Problem effizienter oder überhaupt erst lösbar)
- beachte den "Sonderfall" $p=1$ (also $\varepsilon=0$): ein solcher Monte Carlo-Algorithmus wäre *total korrekt* (hält stets und das Ergebnis ist dabei ("mit Wahrscheinlichkeit 1") korrekt)!

Las Vegas-Election-Algorithmus für anonyme Ringe bekannter Grösse

- 1981 von Itai/Rodeh: Basiert auf Chang/Roberts-Verfahren

- Prinzip:

  oder ein anderer Wert, z.B. 2n, n^2 oder einfach 2 ?

  - wähle eigene Identität id = random (1,...,n), mit n = Ringgrösse
  - message extinction wie gehabt
  - Nachrichten enthalten einen "hop counter": Zählt Anzahl besuchter Knoten
  - falls eine Nachricht mit eigener Identität empfangen wird:
    - prüfe, ob hop counter = n
      - *nein* --> ∃ anderen Knoten gleicher Identität (merken mittels Flag!)
      - *ja* --> gewonnen! (aber falls Flag gesetzt, gibt es andere Gewinner!)
  - falls es mehrere Gewinner gibt:

    FIFO-Kanäle notwendig?

    - nur diese führen eine *neue Election-Runde* durch
    - daher enthalten Nachrichten auch eine Rundenkennung (alte Nachrichten werden in der nächsten Runde einfach ignoriert)

- Ohne Beweis: *Erwartungswert bzgl. Rundenzahl* ≤ e (n/n-1)

  2.718281828...

  - vgl. mit vorherigem Algorithmus für Ringgrösse 2

- Berechnungsdauer ist im Prinzip aber unbegrenzt!

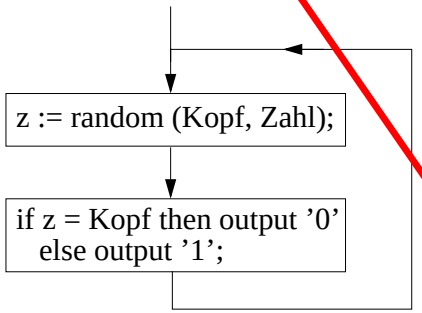
- Algorithmus ist partiell korrekt: *Wenn* er hält, dann mit genau einem leader!

- Verallgemeinerung auf *allgemeine Netze* mit Echo-Algorithmus (statt Ring) ist möglich:
  - wenn die durch Echos gemeldete Baumgrösse ≠ n ist, neue Runde starten etc.

Zufällige reellwertige Zahlen?

- Wenn man im Verfahren von Itai/Rodeh zufällige reelle Zahlen (z.B. zwischen 0 und 1) für die id wählen könnte...
  - wie hoch wäre dann die Wahrscheinlichkeit, dass zwei Prozesse sich für die gleiche Identität entscheiden?
  - wie hoch wäre dann die Rundenzahl bzw. die Nachrichtenkomplexität?

- Realisierung solcher Zufallszahlengeneratoren?



- liefert eine unendliche Folge von 0en und 1en
- Verwende diese als die Nachkommastellen von 0.⋯ im Dualsystem
- Zurückführung des Problems auf perfekten binären Zufallsentscheider

- Unendliche Folgen lassen sich aber nicht in endlicher Zeit generieren und mit endlich vielen Bits speichern...

- Wie wäre es statt dessen mit einer "lazy" Variante, die notwendige Nachkommastellen nur auf Anfrage produziert?
  - etwa: liefere zunächst nur die ersten 32 Nachkommastellen; wenn mehr benötigt werden, fordert man das Objekt "zufällige reellwertige Zahl zwischen 0 und 1" auf, weitere Stellen zu liefern
  - Denkübung: hilft das (grundsätzlich) bei unserem Problem?

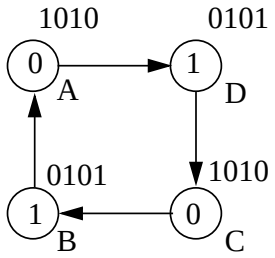
Genügt ein einziges Zufallsbit?

- 1993 haben S. Kurtz, C. Lin, S. Mahaney einen anderen Las-Vegas-Algorithmus für das Election-Problem vorgestellt, der hier *grob skizziert* wird:

für Ringe bekannter Grösse

  - jeder Knoten bestimmt *ein* zufälliges Bit und sendet es nach “rechts”
  - jeder Knoten reicht n-1 Mal ein empfangenes Bit nach “rechts” weiter
  - danach hat jeder Knoten *alle* n Bits (zyklisch verschoben!) gesehen
  - jeder Knoten prüft, ob bei “Ringshift” *seines* gesehenen Bitstrings dieser mit weniger als n shifts erhalten wird (nichttrivialer Automorphismus)
  - falls ja (selten!) --> gesamter Algorithmus wird wiederholt
  - falls nein: unter den n verschiedenen Bitstrings gibt es genau einen maximalen (bei Interpretation als Dualzahl)
  - der eindeutige Prozess, der diesen gesehen hat, ist der Leader

---



- symmetrische Lösung: alle Knoten führen den gleichen Algorithmus aus
- “common sense of orientation” wird vorausgesetzt
- in nebenstehendem Szenario ging es nicht gleich in der ersten Runde gut...

Beachte: die Laufzeit des Algorithmus ist prinzipiell unbegrenzt; wenn er hält, ist ein Leader allerdings eindeutig bestimmt

- Denkübungen (nicht ganz einfach!):
  - wie hoch ist die Best-case-Nachrichtenkomplexität?
  - macht es einen Unterschied, ob die Ringgrösse eine Primzahl ist?
  - kann man die Wahrscheinlichkeit abschätzen, dass eine einzige Runde (für eindeutiges Maximum) genügt?
  - wie hoch mag die *erwartete* Nachrichtenkomplexität sein?

Schätzung der Ringgrösse?

- Für den vorherigen Algorithmus ist die Kenntnis der Ringgrösse n entscheidend.

- Bei *unbekannter Ringgrösse* lässt sich diese aufgrund von zyklischen Wiederholungen im Bitstring mit wenigen Läufen mit hoher Wahrscheinlichkeit korrekt schätzen...
  - Wieviele Läufe sind für eine gewisse Sicherheit notwendig?
  - Konsequenzen, wenn man sich unerkanntermassen irrt?

Wir wollen das an dieser Stelle nicht weitertreiben...

Aus “philosophischer Sicht” ist allerdings interessant:

- Was ist an minimaler (struktureller) Information notwendig, um Symmetrie zu brechen? (Beispiele: Ringgrösse ist eine unbekannte Primzahl; obere Schranke für die Ringgrösse...)
- Unter welchen minimalen Voraussetzungen ist eine deterministische oder probabilistische Lösung möglich?
- Wie “sicher” und effizient können probabilistische Algorithmen für dieses Anwendungsproblem sein?

Satz (ohne Beweis): *Für anonyme Netze unbekannter Grösse existiert kein (det. oder prob. Las Vegas) Election-Algorithmus.*

- Daher stellt sich die Frage, ob die Grösse zumindest mit hoher Wahrscheinlichkeit korrekt abgeschätzt werden kann!