Optimalität d. binären Suche

Geht es sogar noch schneller als bei der binären Suche?

sog. sortiertes Array a und Zahl x, finde den Index von x in a

* Angenommen, wir haben einen Algorithmus, der das Suchproblem für jedes sortierte Array der Länge n löst und dafür t Vergleiche der Form "$a[i] \geq x$?" braucht.

* **Beobachtung:** Das Verhalten des Algorithmus hängt nur von dem Ergebnis der Vergleiche ab!
  t Vergleiche $\Rightarrow \leq 2^t$ mögliche Ergebnisse

* Da das gesuchte Element an jede Stelle sein kann, gibt es aber $\geq n$ mögliche Rückgabewerte.
  (sogar $n+1$, falls x nicht in a ist)

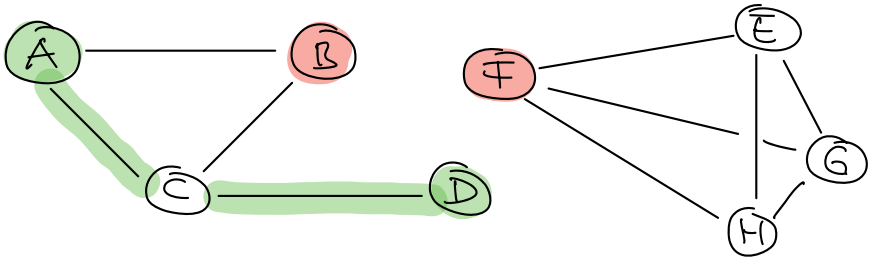
$\Rightarrow$ $\boxed{2^t \geq n, \text{ d.h. } t \geq \log_2(n)}$

denn falls $2^t < n$ dann gibt die Algo in mind. einem Fall das falsche Ergebnis aus!

Binäre Suche ist optimal ☺

Suche in Graphen a.k.a. Netzwerken

"Social Graph": Zwei Personen sind verbunden, falls sie befreundet sind:



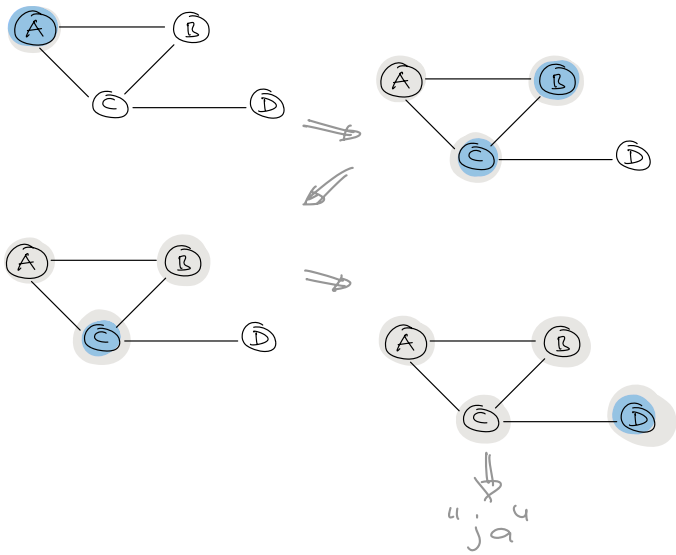
z.B. ja für A & D
nein für B & F

Frage: Gegeben 2 Personen P und Q, gibt es eine "Freundschaftskette" zwischen ihnen?

Idee: Starte bei P und Suche nach Q, angefangen bei den Nachbarn von P...
Herausforderung: Sich nicht im Kreis zu drehen (A → B → A → B → ...)!

Algorithmus "Freundschaftskette"
Eingabe: Der Graph & zwei Personen P & Q
Vorgehen:

```
todo ← {P}
seen ← {P}
while todo nicht leer:
    x ← entferne erstes Element aus todo
    if x = Q: return "ja"
    füge alle Freunde von x, die nicht in seen
        enthalten sind, zu todo und seen hinzu
return "nein"
```



Laufzeit: Anzahl Schleifendurchläufe $\leq$ Anzahl Personen ☺

("seen" wächst jeweils um eine Person)