Ideen und Konzepte der Informatik

Online Algorithmen

... was ist es wert, die Zukunft zu kennen.
...oder: die Chance auf der besten Praline.

Antonios Antoniadis

Bisher...

Für alle bisher betrachteten Algorithmen (Sortieren, kürzeste Wege, ...) war die Eingabe immer komplett bekannt.

Bisher...
Für alle bisher betrachteten Algorithmen (Sortieren, kürzeste Wege, ...) war die Eingabe immer komplett bekannt.

Aber...
In der Realität ist die Eingabe im Voraus meist nicht vollständig bekannt.

- Entscheidungen müssen getroffen werden, ohne zu wissen, „was kommt“.
- Beispiele:
  - Börse (Aktienkauf und Verkauf)
  - Wann tanken (Spritpreis)?

In der Regel:

- Es gibt keine Information über die zukünftigen Daten.
- Daten/Informationen werden schrittweise verfügbar.
- Die Daten müssen sofort verarbeitet werden und Entscheidungen, die getroffen werden, sind nicht umkehrbar.

Ungewissheit

„Das Leben ist wie eine Schachtel Pralinen. Man weiß nie, was man kriegt“.

Forrest Gump

Ungewissheit ist schwer auszuhalten

Wie kann man Daten halbwegs gut verarbeiten, wenn nur ein Teil der Eingabe bekannt ist?

Ungewissheit ist schwer auszuhalten

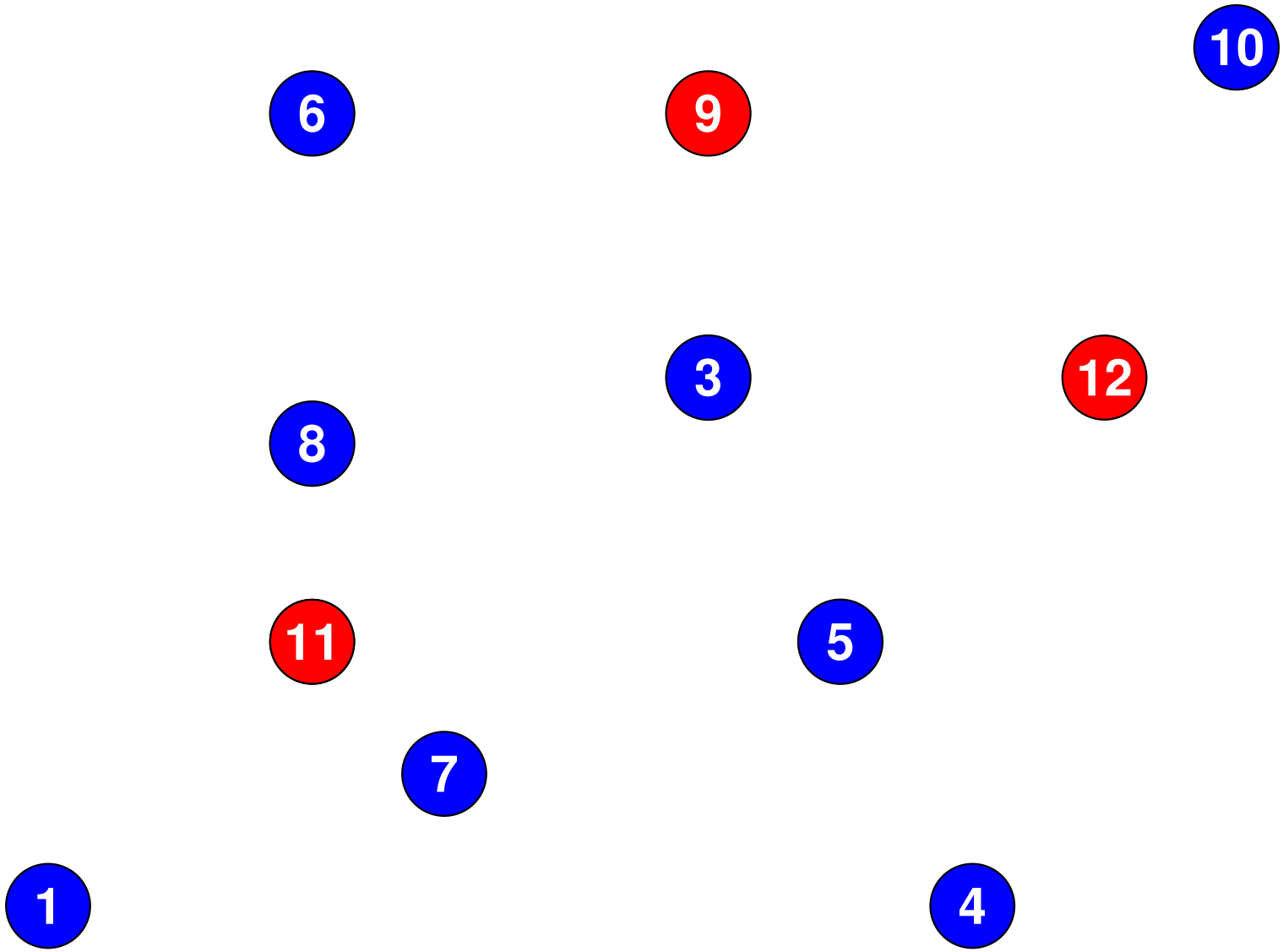
Wie kann man Daten halbwegs gut verarbeiten, wenn nur ein Teil der Eingabe bekannt ist?

Erstaunlicherweise ist es oft möglich, trotz Ungewissheit sehr gute Entscheidungen zu treffen!

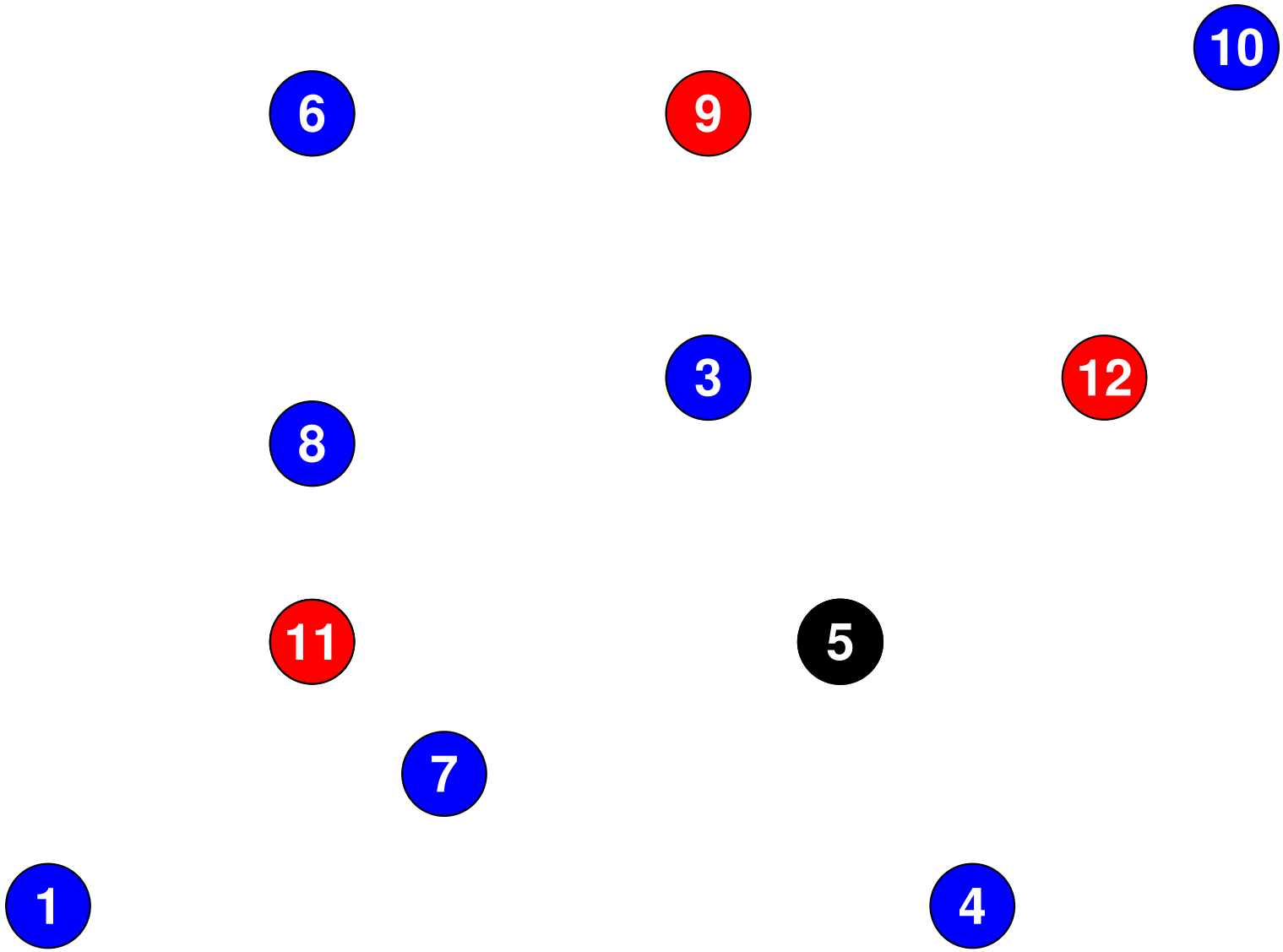
Das Feuerwehr-Problem

- Es gibt n Standorte und $k < n$ viele Feuerwehrautos, die jeweils auf einem von diesen Standorten positioniert sind.
- Über die Zeit kommen Anrufe in der Feuerwehrzentrale an, die jeweils ein Feuer auf einem der n Standorte melden.
- Falls ein Auto dort positioniert ist, müssen Sie nichts tun.
- Falls aber kein Auto dort positioniert ist, müssen Sie ein anderes Auto hinschicken. Dieses wechselt somit den Standort.
- Wir möchten die Anzahl der Standortwechsel minimieren.

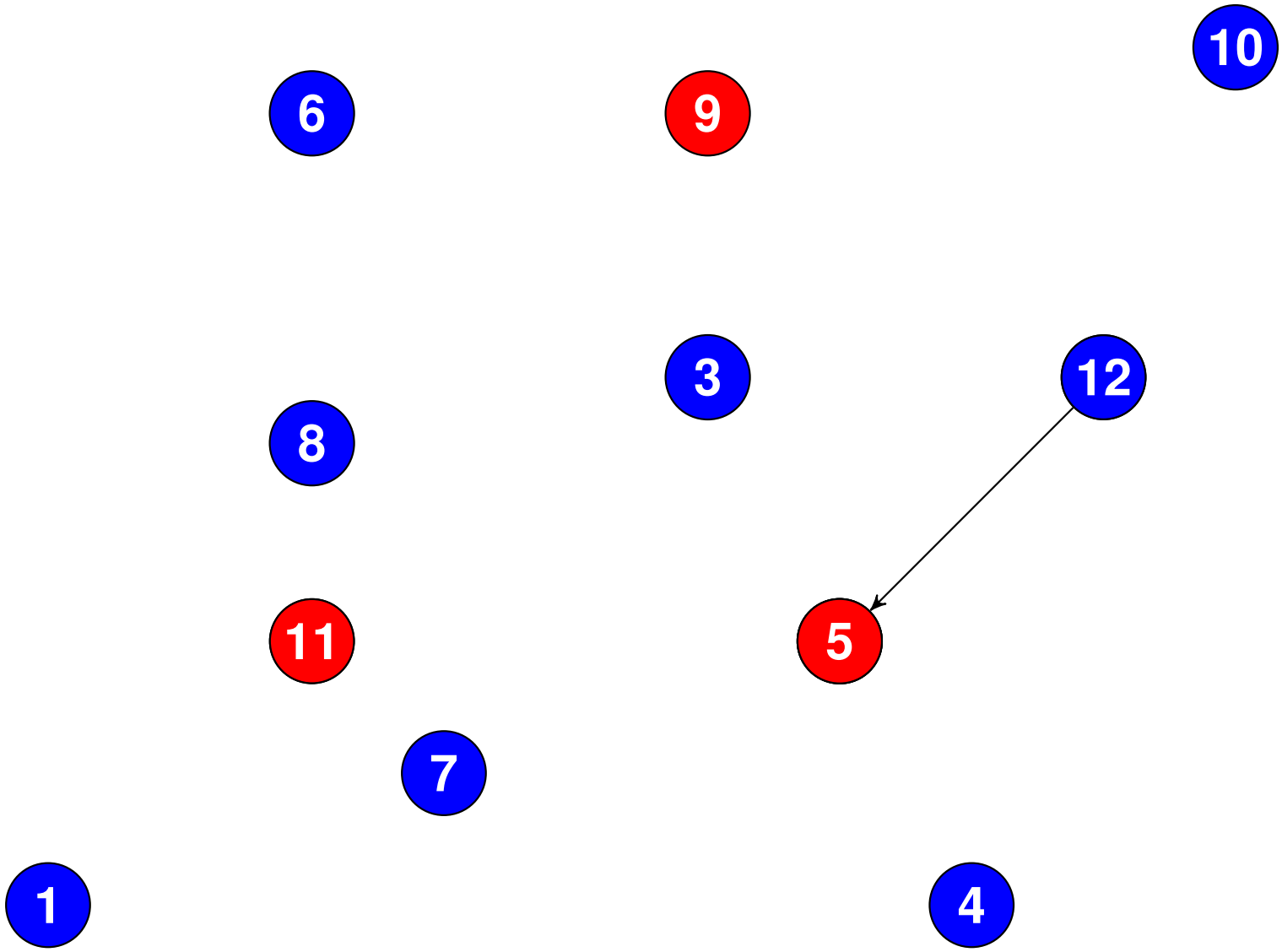
Ein Beispiel



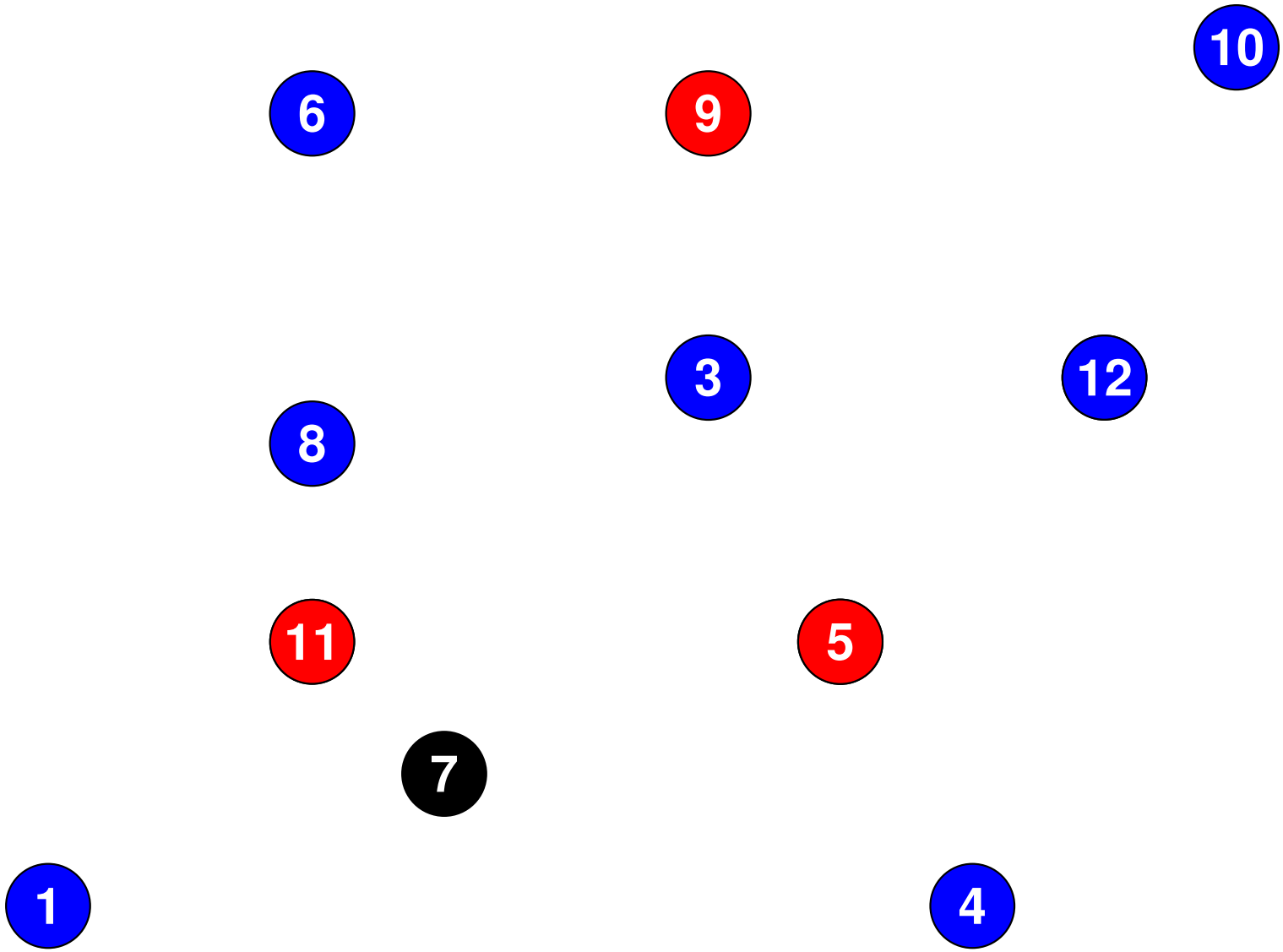
Ein Beispiel



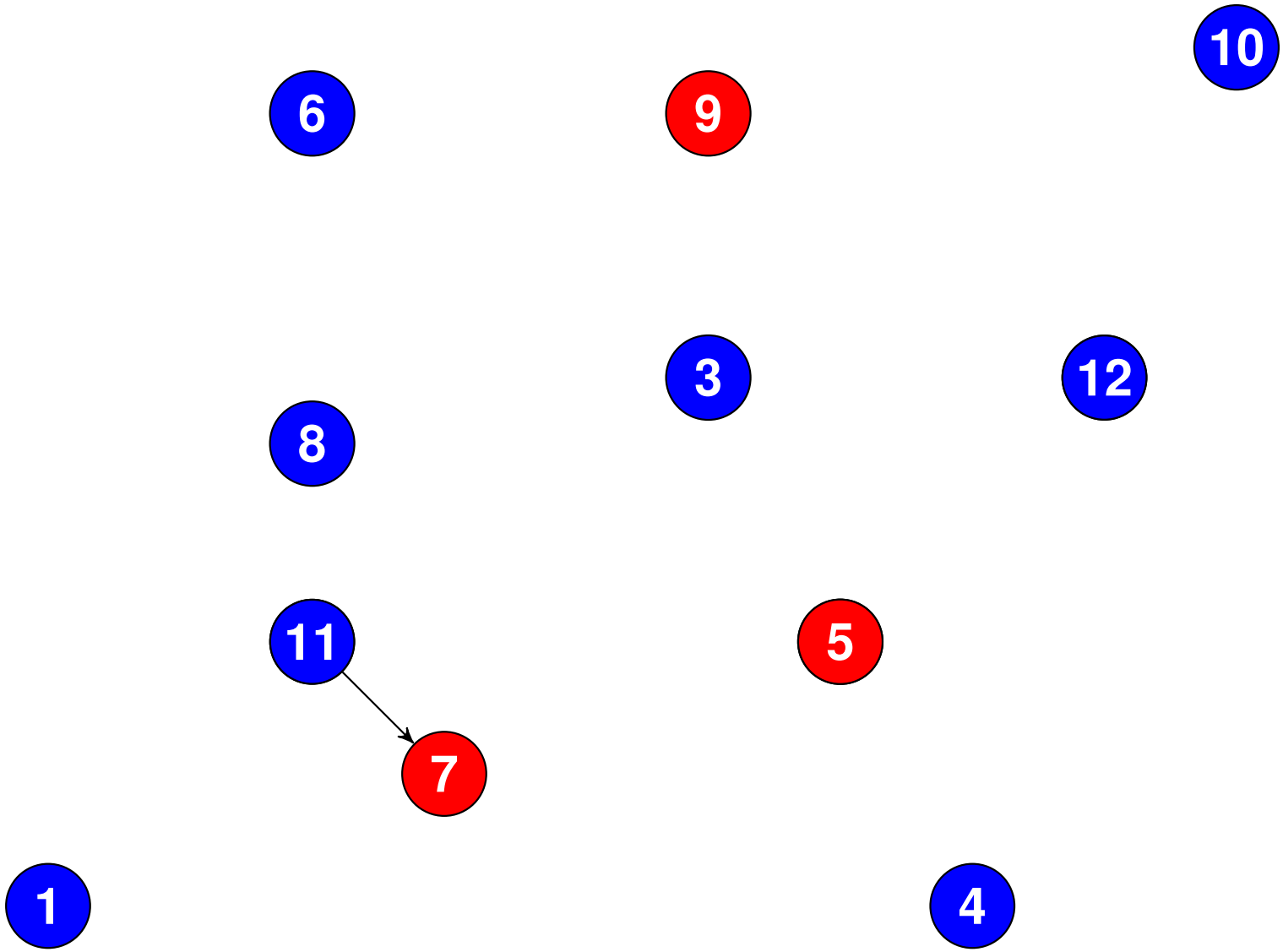
Ein Beispiel



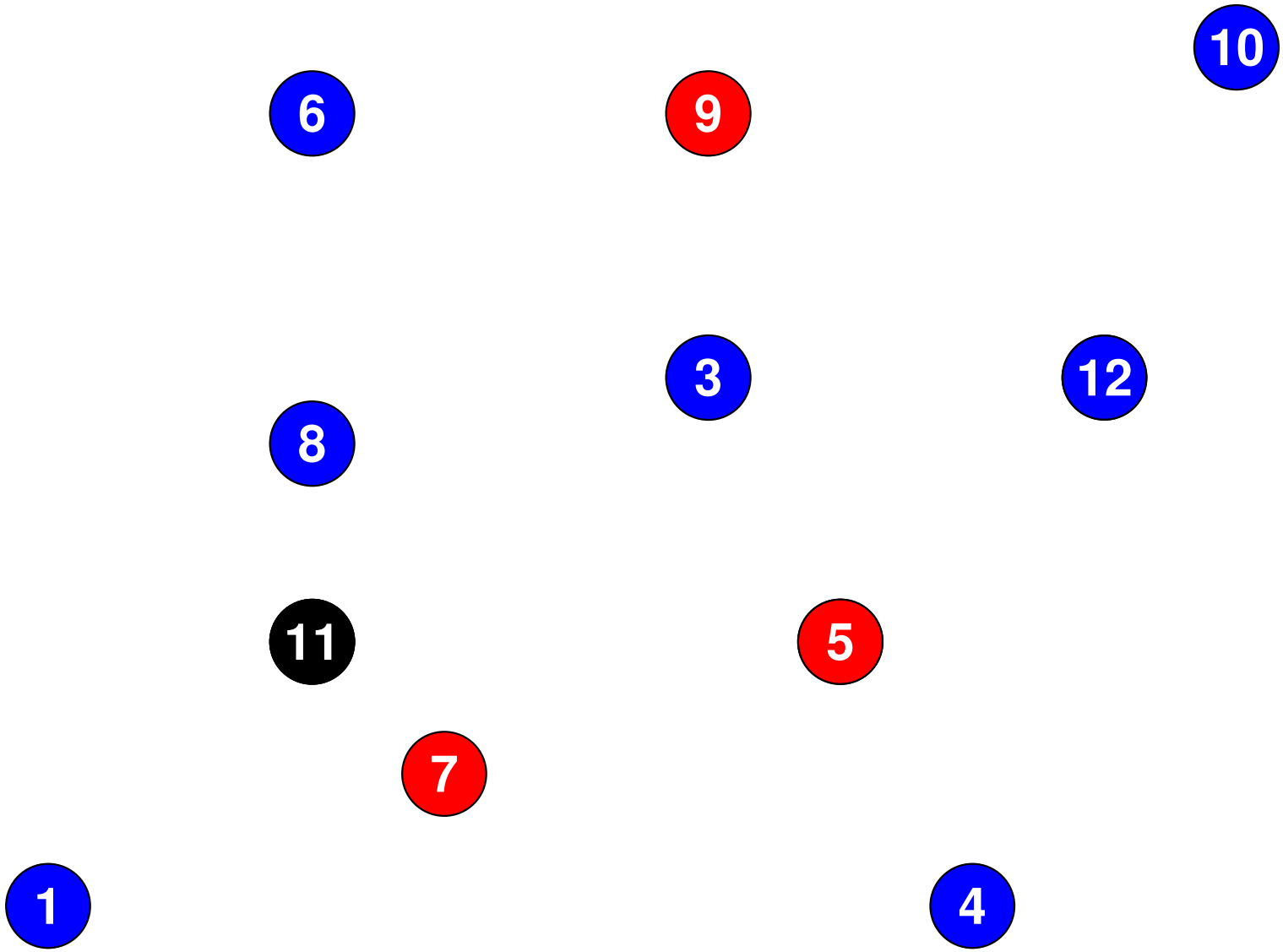
Ein Beispiel



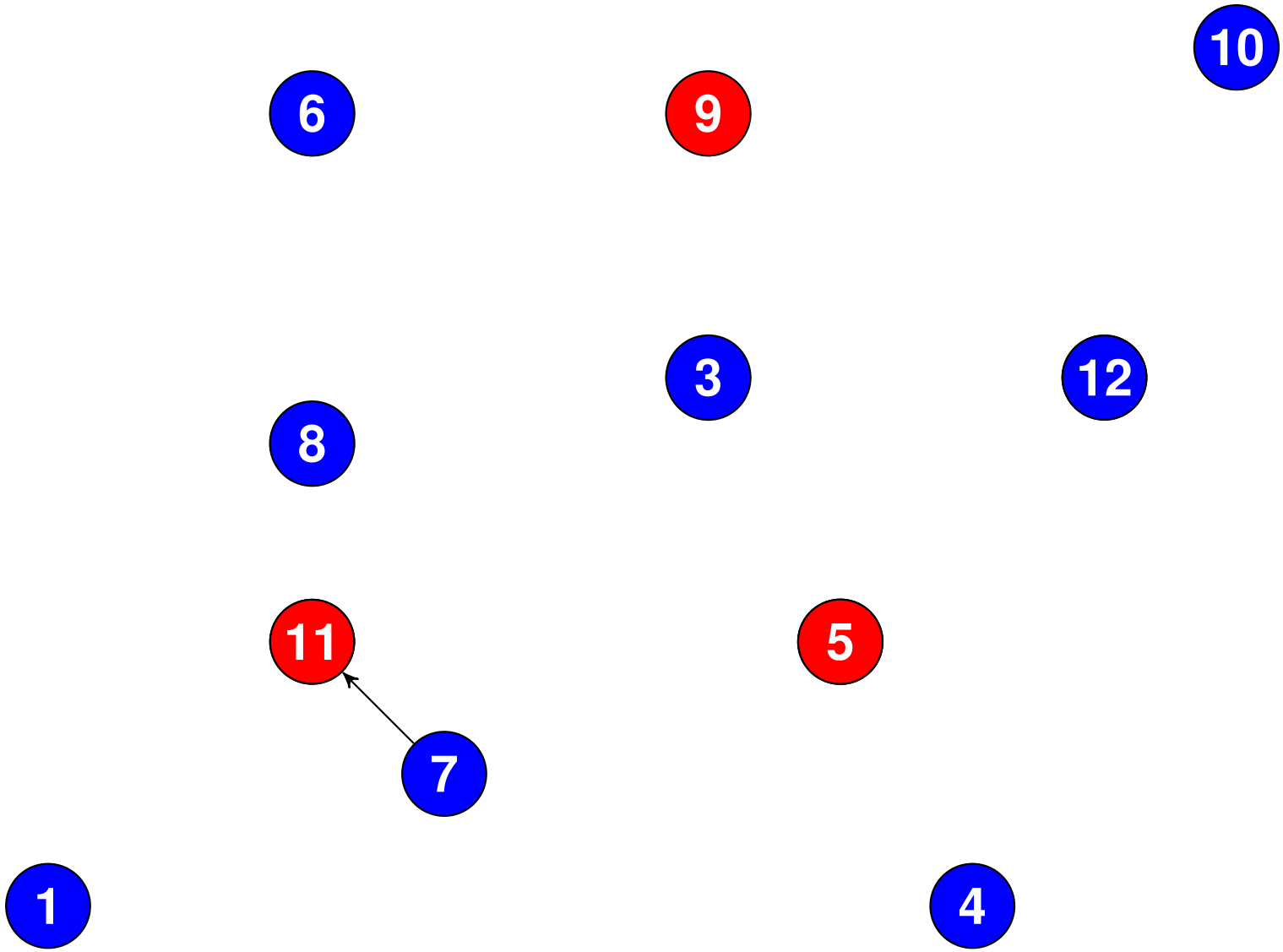
Ein Beispiel



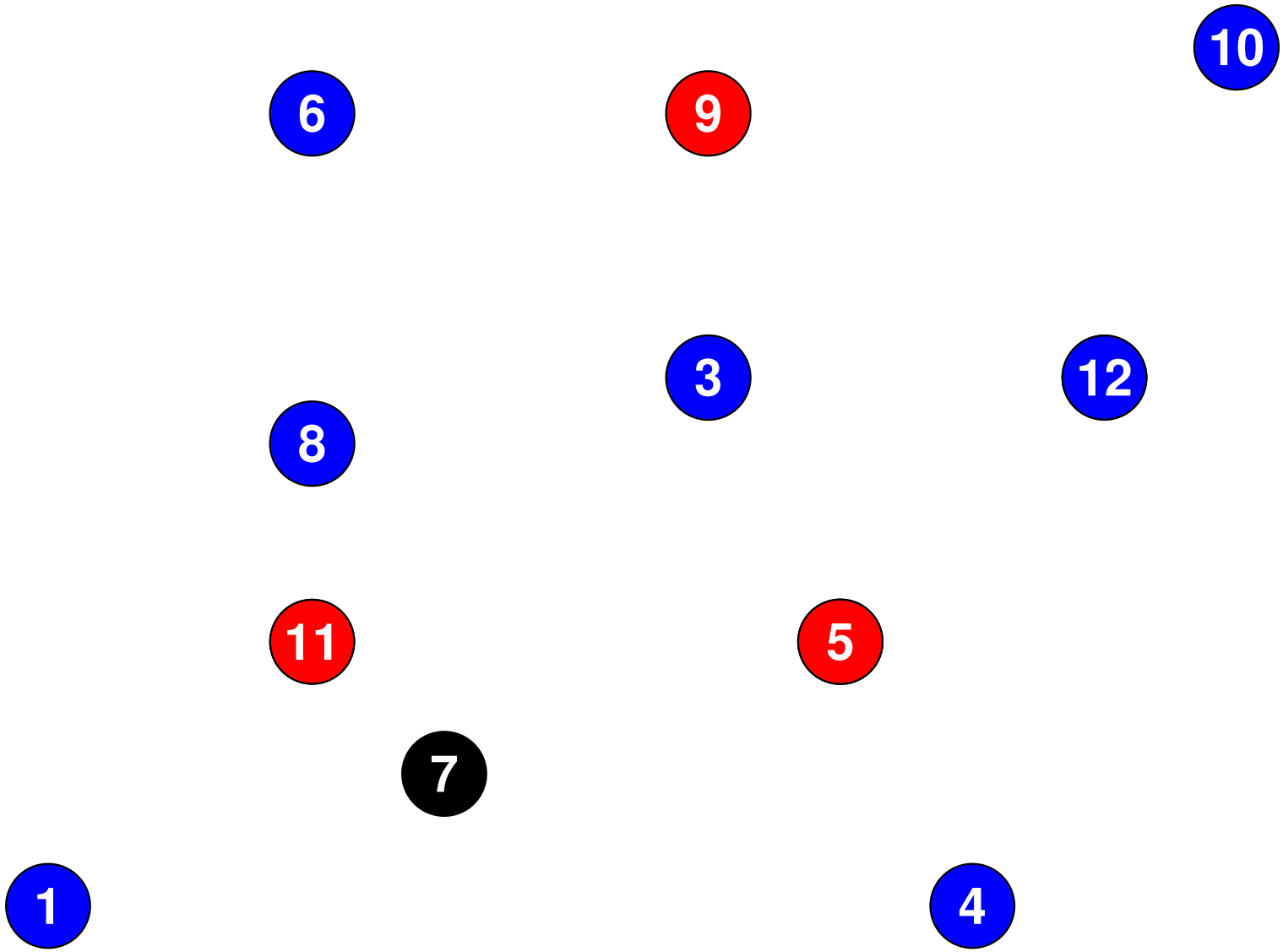
Ein Beispiel



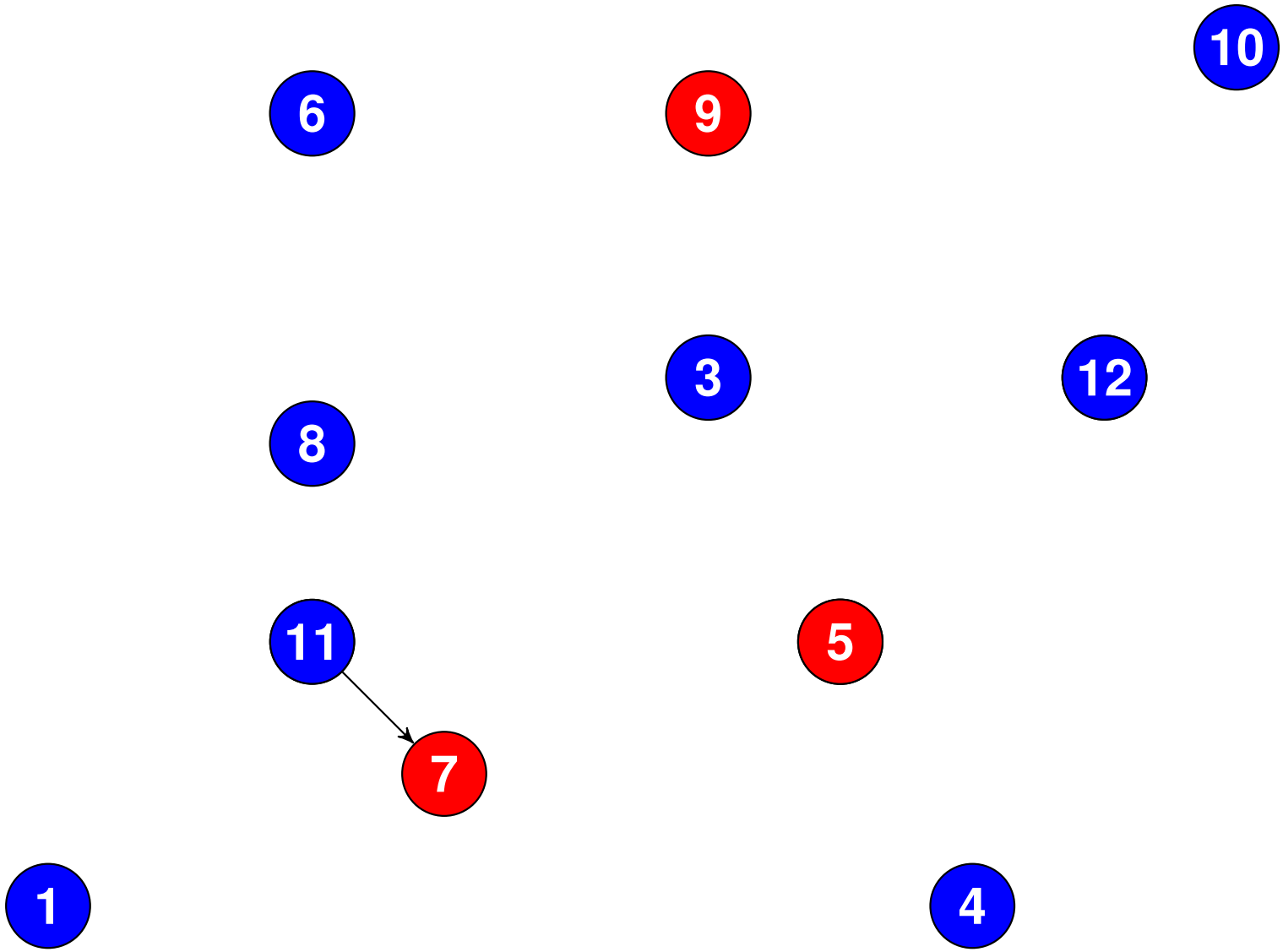
Ein Beispiel



Ein Beispiel



Ein Beispiel



Offline Algorithmen

Nehmen wir kurz an, wir kennen die Zukunft. In diesem Fall: die Sequenz der brennenden Standorte. Wir sprechen dann von einem offline Problem.

Offline Algorithmen

Nehmen wir kurz an, wir kennen die Zukunft. In diesem Fall: die Sequenz der brennenden Standorte. Wir sprechen dann von einem offline Problem.

Wie würden Sie vorgehen?

Offline Algorithmen

Nehmen wir kurz an, wir kennen die Zukunft. In diesem Fall: die Sequenz der brennenden Standorte. Wir sprechen dann von einem offline Problem.

Wie würden Sie vorgehen?

Algorithmus LFD (Longest Forward Distance):

Immer dasjenige Auto schicken, welches an dem Standort steht, an dem es am spätesten wieder brennt.

Offline Algorithmen

Nehmen wir kurz an, wir kennen die Zukunft. In diesem Fall: die Sequenz der brennenden Standorte. Wir sprechen dann von einem offline Problem.

Wie würden Sie vorgehen?

Algorithmus LFD (Longest Forward Distance):

Immer dasjenige Auto schicken, welches an dem Standort steht, an dem es am spätesten wieder brennt.

Theorem

LFD ist optimal, also kann es keine Eingabe geben, auf der ein anderer Algorithmus weniger Umstationierungen als LFD hat.

Schade ist...

Wir können nicht LFD verwenden, da wir in der Regel die Sequenz der Feuer nicht kennen.

Schade ist...

Wir können nicht LFD verwenden, da wir in der Regel die Sequenz der Feuer nicht kennen.

Können wir aber ggf. einen Algorithmus finden, der nicht viel öfter umstationiert als LFD?

Kompetitiver Faktor

Wir messen die “Güte” eines Online-Algorithmus A mit dem Kompetitiven Faktor:

$$\max_{I} \left(\frac{\text{Kosten von A auf } I}{\text{Kosten vom besten offline Algorithmus auf } I} \right).$$

Kompetitiver Faktor

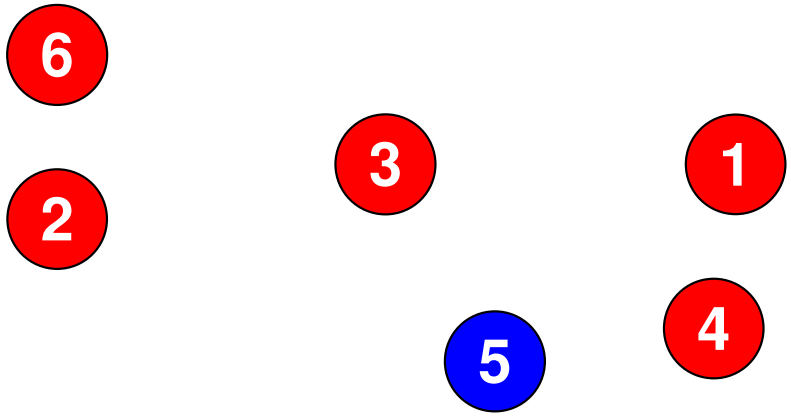
Wir messen die “Güte” eines Online-Algorithmus A mit dem Kompetitiven Faktor:

$$\max_{I} \left(\frac{\text{Kosten von A auf } I}{\text{Kosten vom besten offline Algorithmus auf } I} \right).$$

Siehe auch: Blatt 9, Aufgabe 2.

Theorem

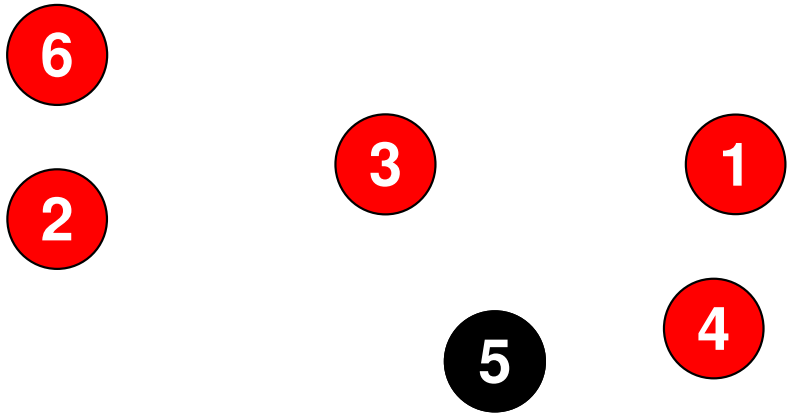
Kein (deterministischer) Algorithmus für das Feuerwehr-Problem kann einen Kompetitiven-Faktor besser als k haben.



- Betrachten wir beliebigen Algorithmus A.
- Sequenz: $k + 1$ Standorte, mit Feuer immer dort, wo A kein Auto hat.
- A: Anzahl Umstationierungen $=$ Länge der Sequenz
- LFD: Nicht mehr als (Länge der Sequenz)$/k$.

Theorem

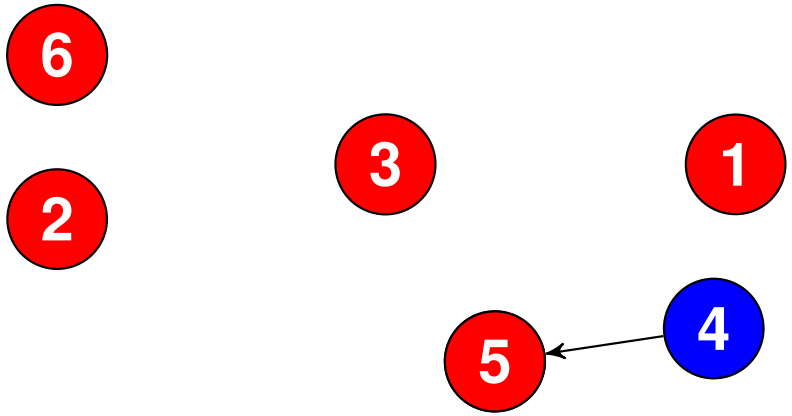
Kein (deterministischer) Algorithmus für das Feuerwehr-Problem kann einen Kompetitiven-Faktor besser als k haben.



- Betrachten wir beliebigen Algorithmus A.
- Sequenz: $k + 1$ Standorte, mit Feuer immer dort, wo A kein Auto hat.
- A: Anzahl Umstationierungen $=$ Länge der Sequenz
- LFD: Nicht mehr als (Länge der Sequenz)$/k$.

Theorem

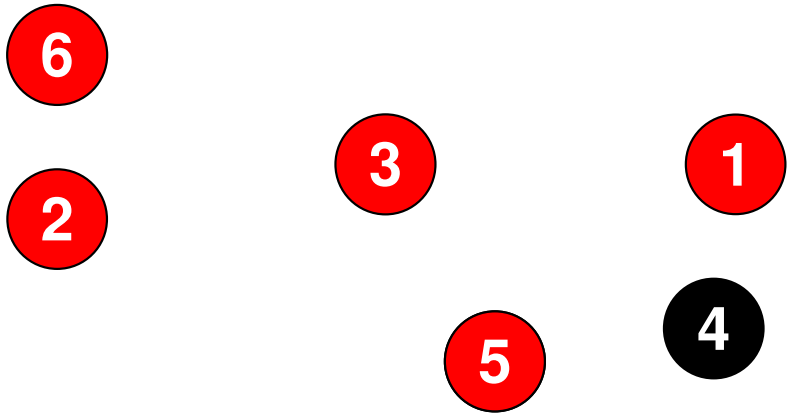
Kein (deterministischer) Algorithmus für das Feuerwehr-Problem kann einen Kompetitiven-Faktor besser als k haben.



- Betrachten wir beliebigen Algorithmus A.
- Sequenz: $k + 1$ Standorte, mit Feuer immer dort, wo A kein Auto hat.
- A: Anzahl Umstationierungen $=$ Länge der Sequenz
- LFD: Nicht mehr als (Länge der Sequenz)$/k$.

Theorem

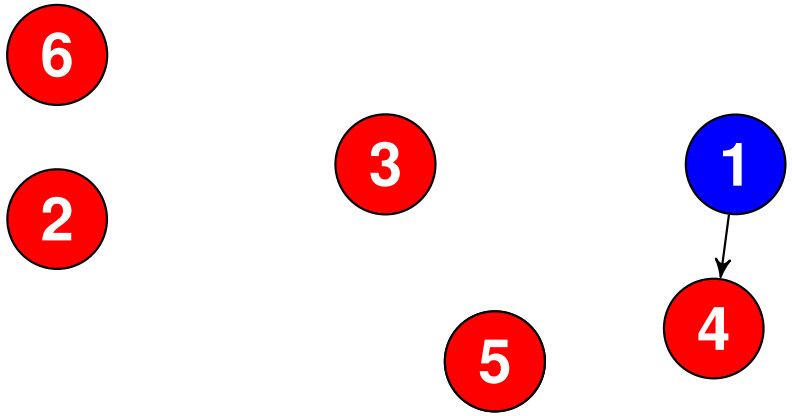
Kein (deterministischer) Algorithmus für das Feuerwehr-Problem kann einen Kompetitiven-Faktor besser als k haben.



- Betrachten wir beliebigen Algorithmus A.
- Sequenz: $k + 1$ Standorte, mit Feuer immer dort, wo A kein Auto hat.
- A: Anzahl Umstationierungen $=$ Länge der Sequenz
- LFD: Nicht mehr als (Länge der Sequenz)$/k$.

Theorem

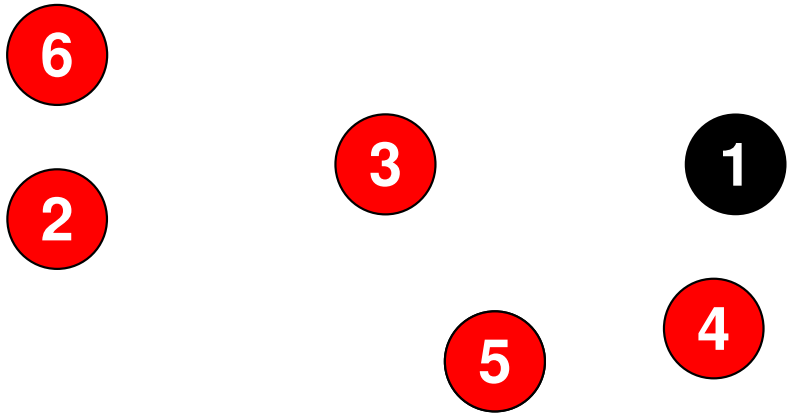
Kein (deterministischer) Algorithmus für das Feuerwehr-Problem kann einen Kompetitiven-Faktor besser als k haben.



- Betrachten wir beliebigen Algorithmus A.
- Sequenz: $k + 1$ Standorte, mit Feuer immer dort, wo A kein Auto hat.
- A: Anzahl Umstationierungen $=$ Länge der Sequenz
- LFD: Nicht mehr als (Länge der Sequenz)$/k$.

Theorem

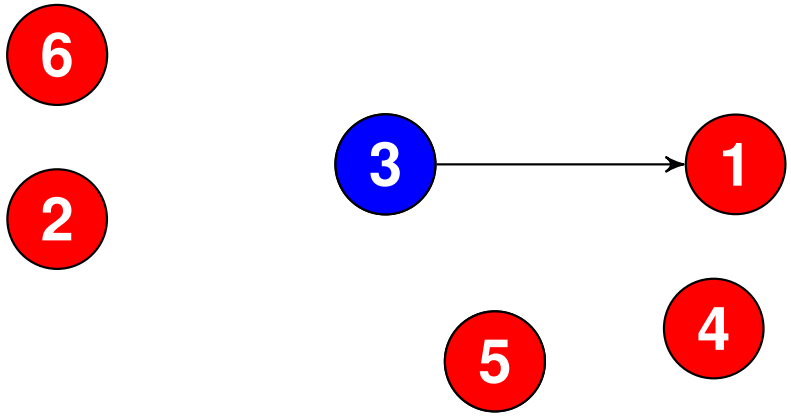
Kein (deterministischer) Algorithmus für das Feuerwehr-Problem kann einen Kompetitiven-Faktor besser als k haben.



- Betrachten wir beliebigen Algorithmus A.
- Sequenz: $k + 1$ Standorte, mit Feuer immer dort, wo A kein Auto hat.
- A: Anzahl Umstationierungen $=$ Länge der Sequenz
- LFD: Nicht mehr als (Länge der Sequenz)$/k$.

Theorem

Kein (deterministischer) Algorithmus für das Feuerwehr-Problem kann einen Kompetitiven-Faktor besser als k haben.



- Betrachten wir beliebigen Algorithmus A.
- Sequenz: $k + 1$ Standorte, mit Feuer immer dort, wo A kein Auto hat.
- A: Anzahl Umstationierungen $=$ Länge der Sequenz
- LFD: Nicht mehr als (Länge der Sequenz)$/k$.

Markierungsalgorithmen

Klasse von Algorithmen, die:

- jeden neu brennenden Standort markieren.
- Immer wenn ein Auto umstationiert werden muss, wird eins von einem nicht markierten Standort ausgewählt.
- Wenn k Standorte markiert sind, werden alle Markierungen aufgehoben (Ende dieser Phase, beginn einer neuen Phase).

Theorem

Jeder Markierungsalgorithmus ist k-kompetitiv.

- Umstationierungen eines Markierungsalgorithmus pro Phase: maximal k
- Umstationierungen von LFD pro Phase: 1 (formal gesehen wird hier auch das erste Feuer der nächsten Phase mit berücksichtigt).
- Insgesamt:

$$\text{Kompetitiver Faktor} \leq \frac{\sum_{\text{Phasen}} k}{\sum_{\text{Phasen}} 1} = k$$

Diskussion

- Recht restriktives Modell.
- In der Praxis sind oft einige Informationen bekannt:
  - Finanzwelt: Annahmen über Kursverläufe
  - Entwicklung des Ölpreises
  - Logistik: Verteilung der Aufträge
- Solches Wissen kann bei der Modellierung berücksichtigt werden.

Experten

- Angenommen Sie wollen einen Monat lang das Wetter vorhersagen (oder Aktien kaufen/verkaufen).
- Es gibt *n* Experten, die Vorhersagen liefern, z.B. ob Aktie *X* heute steigt oder fällt.
- Am Ende jedes Tages weiss man, welche Experten recht hatten und welche nicht.
- Welcher der insgesamt beste Experte ist, wissen Sie nur am Ende des Monats.
- Erstaunlich ist, dass es möglich ist, eine Leistung, ähnlich der des besten Experten zu erzielen!

Algorithmus

- Wir geben jedem Experten i ein Gewicht w_i^t in jedem Schritt t. Am Anfang $w_i^1 = 1$ für alle, also ist das Gesamtgewicht $W^1 = n$.
- Wenn er falsch liegt, wird sein Gewicht nach dem Schritt mit $(1 - \epsilon)$ multipliziert (also wir vertrauen ihm weniger).
- An jedem Tag entscheiden wir uns für das, was die gewichtete Mehrheit sagt.

Jedes Mal wenn wir falsch liegen:

- $W^{t+1} \leq (1 - \epsilon/2)W^t \Rightarrow W^M \leq (1 - \epsilon/2)^M \cdot n$ wenn M die Anzahl Tage ist an der wir falsch lagen.
- Für jeden Experten i, $(1 - \epsilon)^{m_i} \leq W^M$, wenn m_i die Anzahl seiner falschen Vorhersagen ist.
- Also, $(1 - \epsilon)^{m_i} \leq (1 - \epsilon/2)^M \cdot n$, und mit Auflösen nach M,

$$M \leq 2m_i(1 + \epsilon) + \frac{2}{\epsilon} \cdot \log n.$$

- Also sind wir fast halb so gut wie der beste Experte. Mit Randomisierung können wir uns noch mehr der Leistung des besten Experten annähern.

Zusammenfassung

- In der Praxis/Im Leben herrscht oft Ungewissheit – Man kennt die Zukunft nicht.

- Man kann oft trotzdem nachweisbar sehr gute Entscheidungen treffen.