

Übung 7 – Algorithmen II

Moritz Laupichler, Nikolai Maas – {moritz.laupichler, nikolai.maas}@kit.edu
https://algo2.iti.kit.edu/AlgorithmenII_WS23.php

Institut für Theoretische Informatik - Algorithm Engineering

```
    result = current_weight;
    return true;
}

for( EdgeID eid = graph.edgeBegin( current ); eid != graph.edgeEnd( current ); ++eid ){
    const Edge & edge = graph.getEdge( eid );
    COUNTING( statistic_data.inc( DijkstraStatisticData::TOUCHED_EDGES ); )
    if( edge.forward ){
        COUNTING( statistic_data.inc( DijkstraStatisticData::RELAXED_EDGES ); )
        weight new_weight = edge.weight + current_weight;
        GUARANTEE( new_weight >= current_weight, std::runtime_error, "Weight overflow detected." );
        if( !priority_queue.isReached( edge.target ) ){
            COUNTING( statistic_data.inc( DijkstraStatisticData::SUCCESSFULLY_RELAXED_EDGES ); )
            COUNTING( statistic_data.inc( DijkstraStatisticData::REACHED_NODES ); )
            priority_queue.push( edge.target, new_weight );
        } else {
            if( priority_queue.getCurrentKey( edge.target ) > new_weight ){
                COUNTING( statistic_data.inc( DijkstraStatisticData::SUCCESSFULLY_RELAXED_NODES ); )
                priority_queue.decreaseKey( edge.target, new_weight );
            }
        }
    }
}
```

Übungsinhalt

- Parametrisierte Algorithmen
- Besprechung Übungsblatt 3

Warum Probleme parametrisieren?

- es gibt “schwierige” Probleme z.B. Minimum Independent Set
 - allgemeine Instanzen haben zu lange Berechnungszeit
- Kann man **Spezialfälle** eventuell **effizient** berechnen?
 - Identifizierung zusätzlicher Parameter k der Problemstellung
 - falls “Komplexität” in k steckt, effiziente Lösung für konstantes k möglich!

- Ein Problem heißt **fixed parameter tractable**, wenn es eine Laufzeit

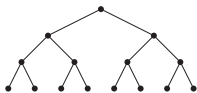
$$T(n, k) = \mathcal{O}(f(k) \cdot p(n))$$

hat, mit $f(\cdot)$ berechenbar, $p(\cdot)$ Polynom.

$f(\cdot)$ darf nicht von n abhängen und $p(\cdot)$ nicht von k

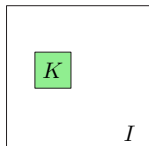
Tiefenbeschränkte Suche

- Aufzählen und Testen aller Möglichkeiten
→ mit geeignetem Suchbaum beschränkter Tiefe
Tiefe ist abhängig von k



Kernbildung

- Probleminstanz auf (schwierigen) **Problemkern** reduzieren
- Problemkern mit anderer Technik lösen



Problemstellung

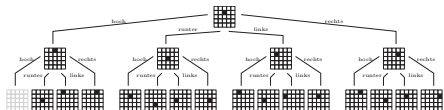
- gegeben $n \times n$ Schiebepuzzle, $k \in \mathbb{N}$
- entscheide, ob das Puzzle in $\leq k$ Zügen gelöst werden kann
 - das Puzzle ist gelöst, wenn die Teile sortiert sind
 - Loch wird pro Zug eine Position horizontal oder vertikal verschoben

Suchbaum-Algorithmus

- es gibt ≤ 4 Möglichkeiten in jedem Zug, k Züge
 - baue Suchbaum (Höhe k , Verzweigungsgrad ≤ 4)
→ Baumgröße $\mathcal{O}(4^k)$
 - teste jeden Knoten auf korrekte Lösung
→ Aufwand $\mathcal{O}(n^2) \in \mathcal{O}(\text{poly}(n))$

⇒ Gesamtaufwand: $\mathcal{O}(4^k n^2) \Rightarrow \text{FPT}$

24	8	13	12	20
11	2		17	21
7	15	14	19	5
6	10	3	9	1
4	23	11	18	22



Ende!



Feierabend!