Datenstrukturen und Algorithmen

Übung 11

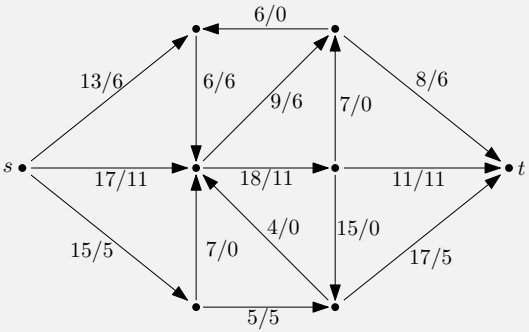
FS 2018

Programm von heute

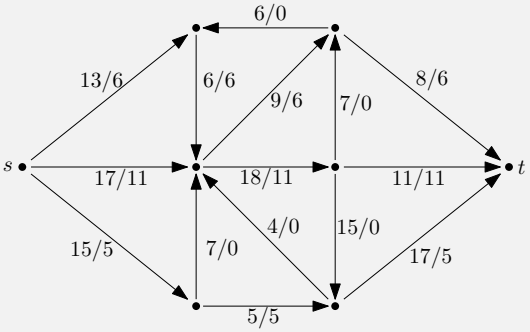
1. Feedback letzte Übung
2. Self Assessment

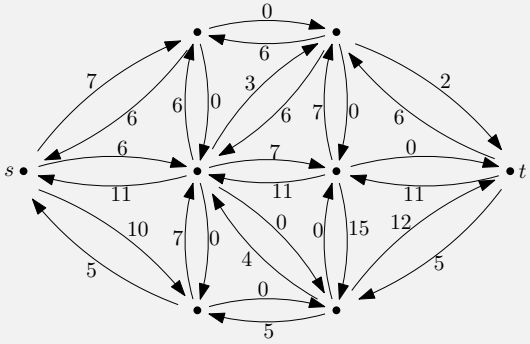
1. Feedback letzte Übung

Aufgabe 10.1: Manual Max-Flow



Aufgabe 10.1: Manual Max-Flow



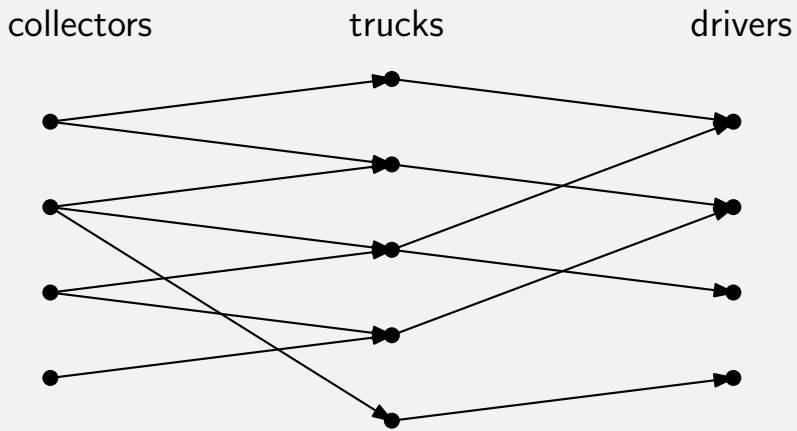


Aufgabe 10.2: Applying Maximum Flow

- Knotenkapazität: Knoten durch Eingangs- und Ausgangsknoten ersetzen.
- Verbinde Knoten durch Kante mit dieser Kapazität.

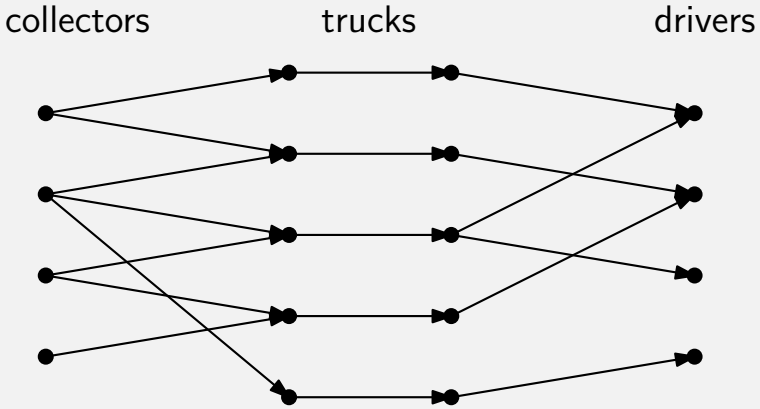
Aufgabe 10.2: Applying Maximum Flow

- Wir haben Auflader, Fahrer und Fahrzeuge.



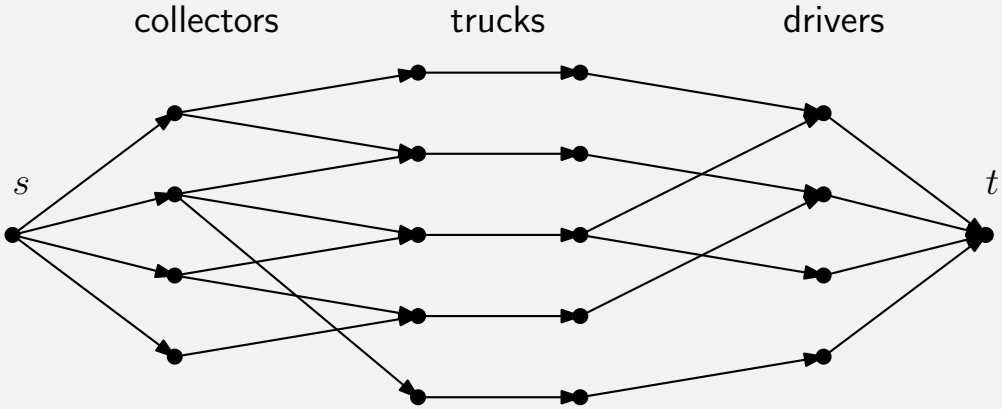
Aufgabe 10.2: Applying Maximum Flow

- Wir haben Auflader, Fahrer und Fahrzeuge.



Aufgabe 10.2: Applying Maximum Flow

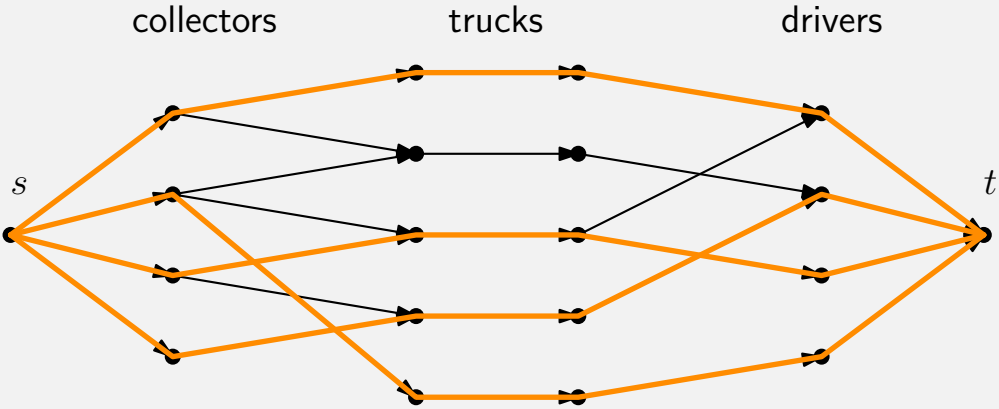
- Wir haben Auflader, Fahrer und Fahrzeuge.



all edges: capacity 1

Aufgabe 10.2: Applying Maximum Flow

- Wir haben Auflader, Fahrer und Fahrzeuge.



all edges: capacity 1

Aufgabe 10.3: Union-Find

```
class UnionFind{
    std::vector<size_t> parents_;
public:
    UnionFind(size_t size) : parents_(size, size) { };

    size_t find(size_t index){
        while(parents_[index] != parents_.size())
            index = parents_[index];
        return index;
    }

    void unite(size_t a, size_t b){
        parents_[find(a)] = b;
    }
};
```

Aufgabe 10.4: Kruskal

```
class Edge{
public:
    size_t u_, v_;
    int c_;
    Edge(size_t u, int v, int c) : u_(u), v_(v), c_(c) {}

    bool operator<(const Edge& other) const {
        return c_ < other.c_;
    }
};
```

Aufgabe 10.4: Kruskal

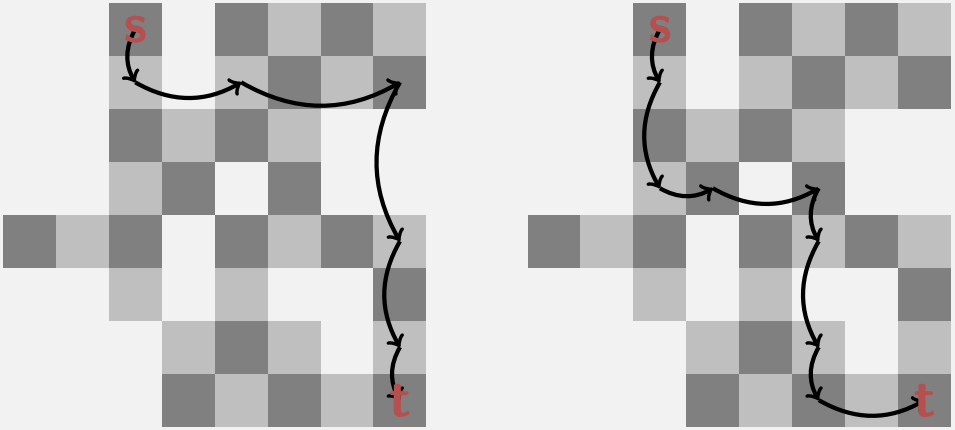
```
std::vector<Edge> edges;

...

UnionFind uf(n_ + 1);
sort(edges.begin(), edges.end());
for(auto e : edges){
        size_t i=uf.find(e.u_);
        size_t j=uf.find(e.v_);
        if(i != j){
                out.addEdge(e);
                uf.unite(i, j);
        }
}
```

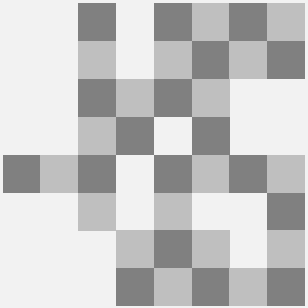
2. Self Assessment

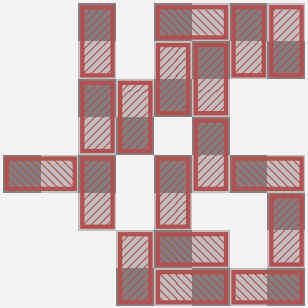
Kürzeste Wege Frage



Wichtigste Frage: Was ist der dazugehörige Zustandsraum?

Max Flow Question





Wichtigste Frage: Wie bildet man das auf ein Max-Flow (Matching) Problem ab?

Fragen?