Datenstrukturen und Algorithmen

Exercise 11

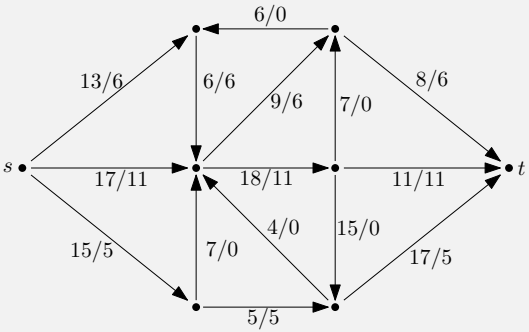
FS 2018

Program of today

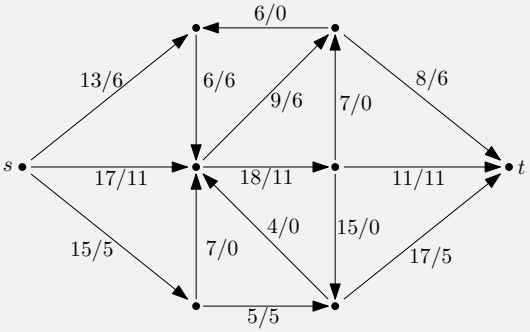
1. Feedback of last exercise
2. Self Assessment

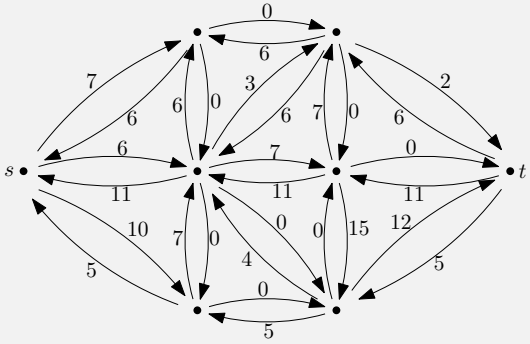
1. Feedback of last exercise

Exercise 10.1: Manual Max-Flow



Exercise 10.1: Manual Max-Flow



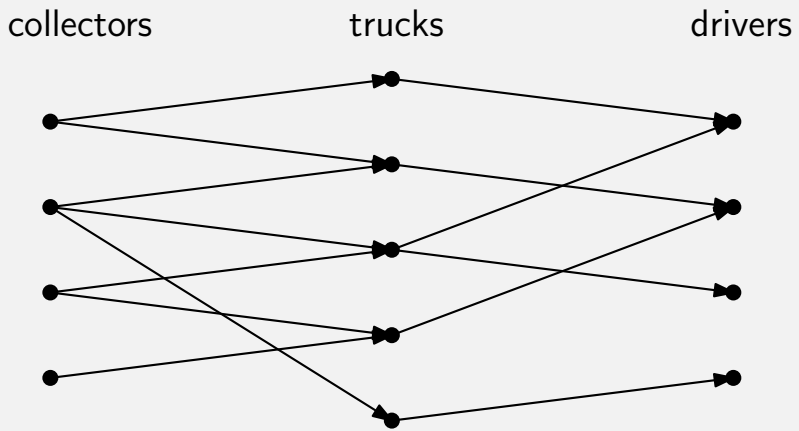


Exercise 10.2: Applying Maximum Flow

- Vertex capacity: replace vertex with an in-vertex and and out-vertex.
- Connect these vertices by an edge with this capacity.

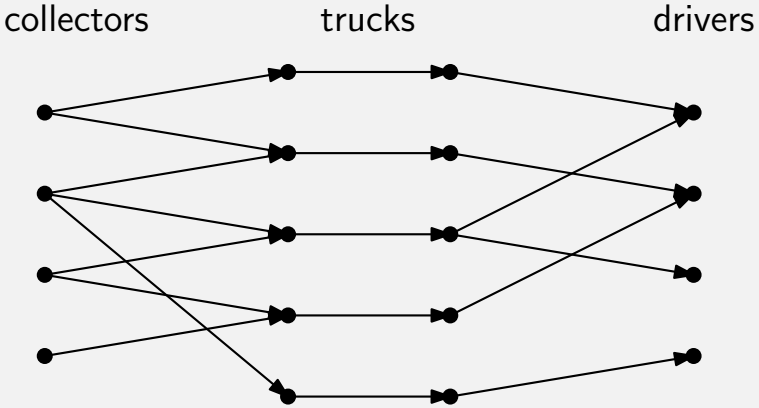
Exercise 10.2: Applying Maximum Flow

- We have collectors, drivers, and trucks



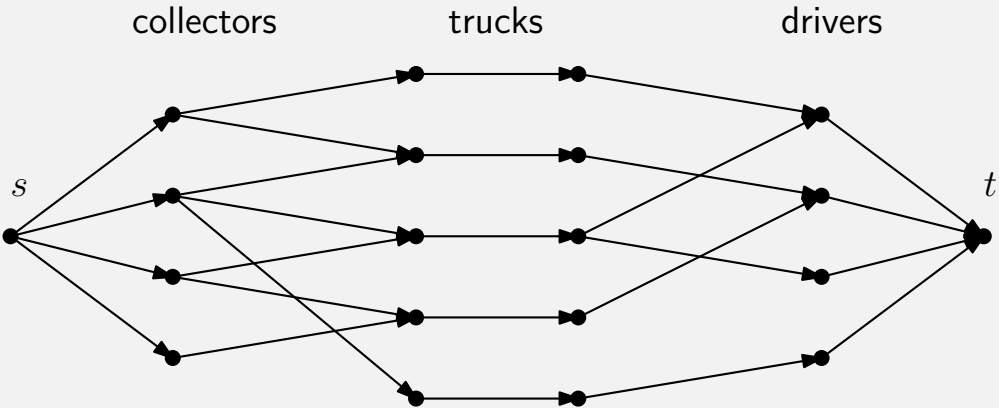
Exercise 10.2: Applying Maximum Flow

- We have collectors, drivers, and trucks



Exercise 10.2: Applying Maximum Flow

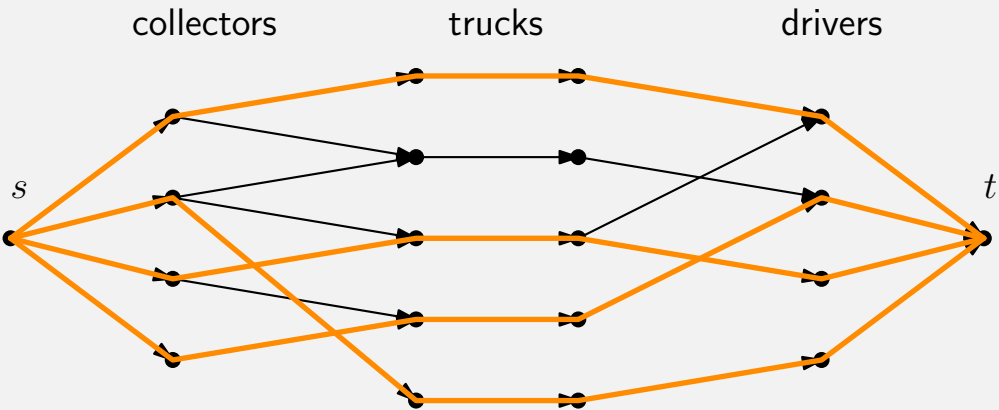
- We have collectors, drivers, and trucks



all edges: capacity 1

Exercise 10.2: Applying Maximum Flow

- We have collectors, drivers, and trucks



all edges: capacity 1

Exercise 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;
    }
};
```

Exercise 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_;
    }
};
```

Exercise 10.4: Kruskal

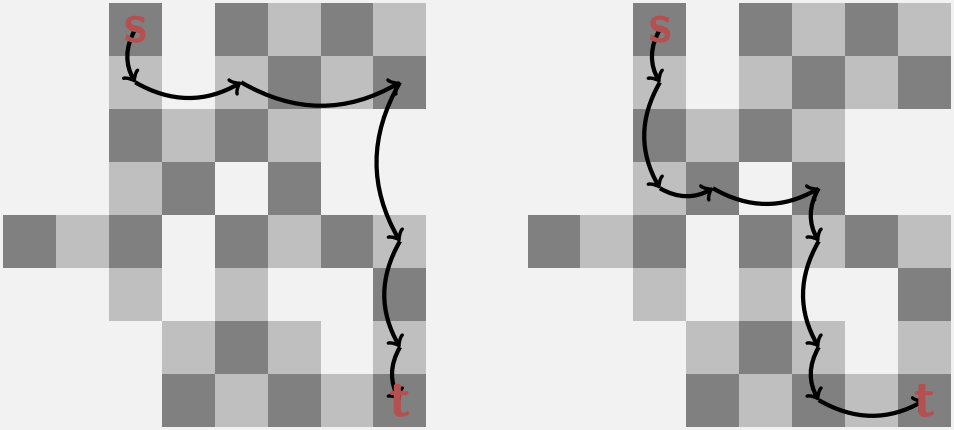
```
std::vector<Edge> edges;

...

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

Shortest Path Question



Most important question: What is the corresponding state space?

Max Flow Question

Most important question: How to map this to a max-flow (matching) setup?

Questions?