## 13.3 Highest Label

**Algorithm 1** highest-label (G, s, t)

1: initialize preflow

2: **foreach**  $u \in V \setminus \{s, t\}$  **do** 

*u.current-neighbour* ← *u.neighbour-list-head* 

4: **while**  $\exists$  active node u **do** 

select active node u with highest label

discharge(u)6:



126/131

# 13.3 Highest Label

Since a discharge-operation is terminated by a deactivating push this gives an upper bound of  $\mathcal{O}(n^3)$  on the number of discharge-operations.

The cost for relabels and saturating pushes can be estimated in exactly the same way as in the case of the generic push-relabel algorithm.

### **Ouestion:**

How do we find the next node for a discharge operation?

### 13.3 Highest Label

#### Lemma 6

When using highest label the number of deactivating pushes is only  $\mathcal{O}(n^3)$ .

A push from a node on level  $\ell$  can only "activate" nodes on levels strictly less than  $\ell$ .

This means, after a deactivating push from u a relabel is required to make u active again.

Hence, after n deactivating pushes without an intermediate relabel there are no active nodes left.

Therefore, the number of deactivating pushes is at most  $n(\#relabels + 1) = \mathcal{O}(n^3).$ 

# 13.3 Highest Label

Maintain lists  $L_i$ ,  $i \in \{0, ..., 2n\}$ , where list  $L_i$  contains active nodes with label i (maintaining these lists induces only constant additional cost for every push-operation and for every relabel-operation).

After a discharge operation terminated for a node u with label k, traverse the lists  $L_k, L_{k-1}, \dots, L_0$ , (in that order) until you find a non-empty list.

Unless the last (deactivating) push was to s or t the list k-1 must be non-empty (i.e., the search takes constant time).

13.3 Highest Label

128/131

### 13.3 Highest Label

Hence, the total time required for searching for active nodes is at most

 $O(n^3) + n(\# deactivating-pushes-to-s-or-t)$ 

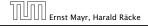
#### Lemma 7

The number of deactivating pushes to s or t is at most  $O(n^2)$ .

With this lemma we get

#### **Theorem 8**

The push-relabel algorithm with the rule highest-label takes time  $\mathcal{O}(n^3)$ .



13.3 Highest Label

22. Feb. 2020

130/131

## 13.3 Highest Label

#### Proof of the Lemma.

- We only show that the number of pushes to the source is at most  $\mathcal{O}(n^2)$ . A similar argument holds for the target.
- After a node v (which must have  $\ell(v) = n+1$ ) made a deactivating push to the source there needs to be another node whose label is increased from  $\leq n+1$  to n+2 before v can become active again.
- This happens for every push that v makes to the source. Since, every node can pass the threshold n+2 at most once, v can make at most n pushes to the source.
- As this holds for every node the total number of pushes to the source is at most  $\mathcal{O}(n^2)$ .

Ernst Mayr, Harald Räcke	13.3 Highest Label	22. Feb. 2020
∐ └─ │ │ Ernst Mayr, Harald Räcke		131/131