

that is, the set of vertices adjacent to some member of  $X$ . Prove **Hall's theorem**: there exists a perfect matching in  $G$  if and only if  $|A| \leq |N(A)|$  for every subset  $A \subseteq L$ .

### 26.3-5 ★

We say that a bipartite graph  $G = (V, E)$ , where  $V = L \cup R$ , is  **$d$ -regular** if every vertex  $v \in V$  has degree exactly  $d$ . Every  $d$ -regular bipartite graph has  $|L| = |R|$ . Prove that every  $d$ -regular bipartite graph has a matching of cardinality  $|L|$  by arguing that a minimum cut of the corresponding flow network has capacity  $|L|$ .

## ★ 26.4 Push-relabel algorithms

In this section, we present the “push-relabel” approach to computing maximum flows. To date, many of the asymptotically fastest maximum-flow algorithms are push-relabel algorithms, and the fastest actual implementations of maximum-flow algorithms are based on the push-relabel method. Push-relabel methods also efficiently solve other flow problems, such as the minimum-cost flow problem. This section introduces Goldberg’s “generic” maximum-flow algorithm, which has a simple implementation that runs in  $O(V^2E)$  time, thereby improving upon the  $O(VE^2)$  bound of the Edmonds-Karp algorithm. Section 26.5 refines the generic algorithm to obtain another push-relabel algorithm that runs in  $O(V^3)$  time.

Push-relabel algorithms work in a more localized manner than the Ford-Fulkerson method. Rather than examine the entire residual network to find an augmenting path, push-relabel algorithms work on one vertex at a time, looking only at the vertex’s neighbors in the residual network. Furthermore, unlike the Ford-Fulkerson method, push-relabel algorithms do not maintain the flow-conservation property throughout their execution. They do, however, maintain a **preflow**, which is a function  $f : V \times V \rightarrow \mathbb{R}$  that satisfies the capacity constraint and the following relaxation of flow conservation:

$$\sum_{v \in V} f(v, u) - \sum_{v \in V} f(u, v) \geq 0$$

for all vertices  $u \in V - \{s\}$ . That is, the flow into a vertex may exceed the flow out. We call the quantity

$$e(u) = \sum_{v \in V} f(v, u) - \sum_{v \in V} f(u, v) \quad (26.14)$$

the **excess flow** into vertex  $u$ . The excess at a vertex is the amount by which the flow in exceeds the flow out. We say that a vertex  $u \in V - \{s, t\}$  is **overflowing** if  $e(u) > 0$ .

We shall begin this section by describing the intuition behind the push-relabel method. We shall then investigate the two operations employed by the method: “pushing” preflow and “relabeling” a vertex. Finally, we shall present a generic push-relabel algorithm and analyze its correctness and running time.

### Intuition

You can understand the intuition behind the push-relabel method in terms of fluid flows: we consider a flow network  $G = (V, E)$  to be a system of interconnected pipes of given capacities. Applying this analogy to the Ford-Fulkerson method, we might say that each augmenting path in the network gives rise to an additional stream of fluid, with no branch points, flowing from the source to the sink. The Ford-Fulkerson method iteratively adds more streams of flow until no more can be added.

The generic push-relabel algorithm has a rather different intuition. As before, directed edges correspond to pipes. Vertices, which are pipe junctions, have two interesting properties. First, to accommodate excess flow, each vertex has an out-flow pipe leading to an arbitrarily large reservoir that can accumulate fluid. Second, each vertex, its reservoir, and all its pipe connections sit on a platform whose height increases as the algorithm progresses.

Vertex heights determine how flow is pushed: we push flow only downhill, that is, from a higher vertex to a lower vertex. The flow from a lower vertex to a higher vertex may be positive, but operations that push flow push it only downhill. We fix the height of the source at  $|V|$  and the height of the sink at 0. All other vertex heights start at 0 and increase with time. The algorithm first sends as much flow as possible downhill from the source toward the sink. The amount it sends is exactly enough to fill each outgoing pipe from the source to capacity; that is, it sends the capacity of the cut  $(s, V - \{s\})$ . When flow first enters an intermediate vertex, it collects in the vertex’s reservoir. From there, we eventually push it downhill.

We may eventually find that the only pipes that leave a vertex  $u$  and are not already saturated with flow connect to vertices that are on the same level as  $u$  or are uphill from  $u$ . In this case, to rid an overflowing vertex  $u$  of its excess flow, we must increase its height—an operation called “relabeling” vertex  $u$ . We increase its height to one unit more than the height of the lowest of its neighbors to which it has an unsaturated pipe. After a vertex is relabeled, therefore, it has at least one outgoing pipe through which we can push more flow.

Eventually, all the flow that can possibly get through to the sink has arrived there. No more can arrive, because the pipes obey the capacity constraints; the amount of flow across any cut is still limited by the capacity of the cut. To make the preflow a “legal” flow, the algorithm then sends the excess collected in the reservoirs of overflowing vertices back to the source by continuing to relabel vertices to above

the fixed height  $|V|$  of the source. As we shall see, once we have emptied all the reservoirs, the preflow is not only a “legal” flow, it is also a maximum flow.

### The basic operations

From the preceding discussion, we see that a push-relabel algorithm performs two basic operations: pushing flow excess from a vertex to one of its neighbors and relabeling a vertex. The situations in which these operations apply depend on the heights of vertices, which we now define precisely.

Let  $G = (V, E)$  be a flow network with source  $s$  and sink  $t$ , and let  $f$  be a preflow in  $G$ . A function  $h : V \rightarrow \mathbb{N}$  is a **height function**<sup>3</sup> if  $h(s) = |V|$ ,  $h(t) = 0$ , and

$$h(u) \leq h(v) + 1$$

for every residual edge  $(u, v) \in E_f$ . We immediately obtain the following lemma.

#### Lemma 26.12

Let  $G = (V, E)$  be a flow network, let  $f$  be a preflow in  $G$ , and let  $h$  be a height function on  $V$ . For any two vertices  $u, v \in V$ , if  $h(u) > h(v) + 1$ , then  $(u, v)$  is not an edge in the residual network. ■

### The push operation

The basic operation  $\text{PUSH}(u, v)$  applies if  $u$  is an overflowing vertex,  $c_f(u, v) > 0$ , and  $h(u) = h(v) + 1$ . The pseudocode below updates the preflow  $f$  and the excess flows for  $u$  and  $v$ . It assumes that we can compute residual capacity  $c_f(u, v)$  in constant time given  $c$  and  $f$ . We maintain the excess flow stored at a vertex  $u$  as the attribute  $u.e$  and the height of  $u$  as the attribute  $u.h$ . The expression  $\Delta_f(u, v)$  is a temporary variable that stores the amount of flow that we can push from  $u$  to  $v$ .

---

<sup>3</sup>In the literature, a height function is typically called a “distance function,” and the height of a vertex is called a “distance label.” We use the term “height” because it is more suggestive of the intuition behind the algorithm. We retain the use of the term “relabel” to refer to the operation that increases the height of a vertex. The height of a vertex is related to its distance from the sink  $t$ , as would be found in a breadth-first search of the transpose  $G^T$ .

PUSH( $u, v$ )

```

1  // Applies when:  $u$  is overflowing,  $c_f(u, v) > 0$ , and  $u.h = v.h + 1$ .
2  // Action: Push  $\Delta_f(u, v) = \min(u.e, c_f(u, v))$  units of flow from  $u$  to  $v$ .
3   $\Delta_f(u, v) = \min(u.e, c_f(u, v))$ 
4  if  $(u, v) \in E$ 
5       $(u, v).f = (u, v).f + \Delta_f(u, v)$ 
6  else  $(v, u).f = (v, u).f - \Delta_f(u, v)$ 
7   $u.e = u.e - \Delta_f(u, v)$ 
8   $v.e = v.e + \Delta_f(u, v)$ 

```

The code for PUSH operates as follows. Because vertex  $u$  has a positive excess  $u.e$  and the residual capacity of  $(u, v)$  is positive, we can increase the flow from  $u$  to  $v$  by  $\Delta_f(u, v) = \min(u.e, c_f(u, v))$  without causing  $u.e$  to become negative or the capacity  $c(u, v)$  to be exceeded. Line 3 computes the value  $\Delta_f(u, v)$ , and lines 4–6 update  $f$ . Line 5 increases the flow on edge  $(u, v)$ , because we are pushing flow over a residual edge that is also an original edge. Line 6 decreases the flow on edge  $(v, u)$ , because the residual edge is actually the reverse of an edge in the original network. Finally, lines 7–8 update the excess flows into vertices  $u$  and  $v$ . Thus, if  $f$  is a preflow before PUSH is called, it remains a preflow afterward.

Observe that nothing in the code for PUSH depends on the heights of  $u$  and  $v$ , yet we prohibit it from being invoked unless  $u.h = v.h + 1$ . Thus, we push excess flow downhill only by a height differential of 1. By Lemma 26.12, no residual edges exist between two vertices whose heights differ by more than 1, and thus, as long as the attribute  $h$  is indeed a height function, we would gain nothing by allowing flow to be pushed downhill by a height differential of more than 1.

We call the operation PUSH( $u, v$ ) a **push** from  $u$  to  $v$ . If a push operation applies to some edge  $(u, v)$  leaving a vertex  $u$ , we also say that the push operation applies to  $u$ . It is a **saturating push** if edge  $(u, v)$  in the residual network becomes **saturated** ( $c_f(u, v) = 0$  afterward); otherwise, it is a **nonsaturating push**. If an edge becomes saturated, it disappears from the residual network. A simple lemma characterizes one result of a nonsaturating push.

**Lemma 26.13**

After a nonsaturating push from  $u$  to  $v$ , the vertex  $u$  is no longer overflowing.

**Proof** Since the push was nonsaturating, the amount of flow  $\Delta_f(u, v)$  actually pushed must equal  $u.e$  prior to the push. Since  $u.e$  is reduced by this amount, it becomes 0 after the push. ■

**The relabel operation**

The basic operation  $\text{RELABEL}(u)$  applies if  $u$  is overflowing and if  $u.h \leq v.h$  for all edges  $(u, v) \in E_f$ . In other words, we can relabel an overflowing vertex  $u$  if for every vertex  $v$  for which there is residual capacity from  $u$  to  $v$ , flow cannot be pushed from  $u$  to  $v$  because  $v$  is not downhill from  $u$ . (Recall that by definition, neither the source  $s$  nor the sink  $t$  can be overflowing, and so  $s$  and  $t$  are ineligible for relabeling.)

$\text{RELABEL}(u)$

```

1  // Applies when:  $u$  is overflowing and for all  $v \in V$  such that  $(u, v) \in E_f$ ,
    we have  $u.h \leq v.h$ .
2  // Action: Increase the height of  $u$ .
3   $u.h = 1 + \min \{v.h : (u, v) \in E_f\}$ 

```

When we call the operation  $\text{RELABEL}(u)$ , we say that vertex  $u$  is *relabelled*. Note that when  $u$  is relabelled,  $E_f$  must contain at least one edge that leaves  $u$ , so that the minimization in the code is over a nonempty set. This property follows from the assumption that  $u$  is overflowing, which in turn tells us that

$$u.e = \sum_{v \in V} f(v, u) - \sum_{v \in V} f(u, v) > 0.$$

Since all flows are nonnegative, we must therefore have at least one vertex  $v$  such that  $(v, u).f > 0$ . But then,  $c_f(u, v) > 0$ , which implies that  $(u, v) \in E_f$ . The operation  $\text{RELABEL}(u)$  thus gives  $u$  the greatest height allowed by the constraints on height functions.

**The generic algorithm**

The generic push-relabel algorithm uses the following subroutine to create an initial preflow in the flow network.

$\text{INITIALIZE-PREFLOW}(G, s)$

```

1  for each vertex  $v \in G.V$ 
2       $v.h = 0$ 
3       $v.e = 0$ 
4  for each edge  $(u, v) \in G.E$ 
5       $(u, v).f = 0$ 
6   $s.h = |G.V|$ 
7  for each vertex  $v \in s.Adj$ 
8       $(s, v).f = c(s, v)$ 
9       $v.e = c(s, v)$ 
10      $s.e = s.e - c(s, v)$ 

```

INITIALIZE-PREFLOW creates an initial preflow  $f$  defined by

$$(u, v).f = \begin{cases} c(u, v) & \text{if } u = s, \\ 0 & \text{otherwise.} \end{cases} \quad (26.15)$$

That is, we fill to capacity each edge leaving the source  $s$ , and all other edges carry no flow. For each vertex  $v$  adjacent to the source, we initially have  $v.e = c(s, v)$ , and we initialize  $s.e$  to the negative of the sum of these capacities. The generic algorithm also begins with an initial height function  $h$ , given by

$$u.h = \begin{cases} |V| & \text{if } u = s, \\ 0 & \text{otherwise.} \end{cases} \quad (26.16)$$

Equation (26.16) defines a height function because the only edges  $(u, v)$  for which  $u.h > v.h + 1$  are those for which  $u = s$ , and those edges are saturated, which means that they are not in the residual network.

Initialization, followed by a sequence of push and relabel operations, executed in no particular order, yields the GENERIC-PUSH-RELABEL algorithm:

GENERIC-PUSH-RELABEL( $G$ )

- 1 INITIALIZE-PREFLOW( $G, s$ )
- 2 **while** there exists an applicable push or relabel operation
- 3     select an applicable push or relabel operation and perform it

The following lemma tells us that as long as an overflowing vertex exists, at least one of the two basic operations applies.

**Lemma 26.14** (*An overflowing vertex can be either pushed or relabeled*)

Let  $G = (V, E)$  be a flow network with source  $s$  and sink  $t$ , let  $f$  be a preflow, and let  $h$  be any height function for  $f$ . If  $u$  is any overflowing vertex, then either a push or relabel operation applies to it.

**Proof** For any residual edge  $(u, v)$ , we have  $h(u) \leq h(v) + 1$  because  $h$  is a height function. If a push operation does not apply to an overflowing vertex  $u$ , then for all residual edges  $(u, v)$ , we must have  $h(u) < h(v) + 1$ , which implies  $h(u) \leq h(v)$ . Thus, a relabel operation applies to  $u$ . ■

### Correctness of the push-relabel method

To show that the generic push-relabel algorithm solves the maximum-flow problem, we shall first prove that if it terminates, the preflow  $f$  is a maximum flow. We shall later prove that it terminates. We start with some observations about the height function  $h$ .

**Lemma 26.15 (Vertex heights never decrease)**

During the execution of the GENERIC-PUSH-RELABEL procedure on a flow network  $G = (V, E)$ , for each vertex  $u \in V$ , the height  $u.h$  never decreases. Moreover, whenever a relabel operation is applied to a vertex  $u$ , its height  $u.h$  increases by at least 1.

**Proof** Because vertex heights change only during relabel operations, it suffices to prove the second statement of the lemma. If vertex  $u$  is about to be relabeled, then for all vertices  $v$  such that  $(u, v) \in E_f$ , we have  $u.h \leq v.h$ . Thus,  $u.h < 1 + \min \{v.h : (u, v) \in E_f\}$ , and so the operation must increase  $u.h$ . ■

**Lemma 26.16**

Let  $G = (V, E)$  be a flow network with source  $s$  and sink  $t$ . Then the execution of GENERIC-PUSH-RELABEL on  $G$  maintains the attribute  $h$  as a height function.

**Proof** The proof is by induction on the number of basic operations performed. Initially,  $h$  is a height function, as we have already observed.

We claim that if  $h$  is a height function, then an operation RELABEL( $u$ ) leaves  $h$  a height function. If we look at a residual edge  $(u, v) \in E_f$  that leaves  $u$ , then the operation RELABEL( $u$ ) ensures that  $u.h \leq v.h + 1$  afterward. Now consider a residual edge  $(w, u)$  that enters  $u$ . By Lemma 26.15,  $w.h \leq u.h + 1$  before the operation RELABEL( $u$ ) implies  $w.h < u.h + 1$  afterward. Thus, the operation RELABEL( $u$ ) leaves  $h$  a height function.

Now, consider an operation PUSH( $u, v$ ). This operation may add the edge  $(v, u)$  to  $E_f$ , and it may remove  $(u, v)$  from  $E_f$ . In the former case, we have  $v.h = u.h - 1 < u.h + 1$ , and so  $h$  remains a height function. In the latter case, removing  $(u, v)$  from the residual network removes the corresponding constraint, and  $h$  again remains a height function. ■

The following lemma gives an important property of height functions.

**Lemma 26.17**

Let  $G = (V, E)$  be a flow network with source  $s$  and sink  $t$ , let  $f$  be a preflow in  $G$ , and let  $h$  be a height function on  $V$ . Then there is no path from the source  $s$  to the sink  $t$  in the residual network  $G_f$ .

**Proof** Assume for the sake of contradiction that  $G_f$  contains a path  $p$  from  $s$  to  $t$ , where  $p = \langle v_0, v_1, \dots, v_k \rangle$ ,  $v_0 = s$ , and  $v_k = t$ . Without loss of generality,  $p$  is a simple path, and so  $k < |V|$ . For  $i = 0, 1, \dots, k - 1$ , edge  $(v_i, v_{i+1}) \in E_f$ . Because  $h$  is a height function,  $h(v_i) \leq h(v_{i+1}) + 1$  for  $i = 0, 1, \dots, k - 1$ . Combining these inequalities over path  $p$  yields  $h(s) \leq h(t) + k$ . But because  $h(t) = 0$ ,

we have  $h(s) \leq k < |V|$ , which contradicts the requirement that  $h(s) = |V|$  in a height function. ■

We are now ready to show that if the generic push-relabel algorithm terminates, the preflow it computes is a maximum flow.

**Theorem 26.18 (Correctness of the generic push-relabel algorithm)**

If the algorithm GENERIC-PUSH-RELABEL terminates when run on a flow network  $G = (V, E)$  with source  $s$  and sink  $t$ , then the preflow  $f$  it computes is a maximum flow for  $G$ .

**Proof** We use the following loop invariant:

Each time the **while** loop test in line 2 in GENERIC-PUSH-RELABEL is executed,  $f$  is a preflow.

**Initialization:** INITIALIZE-PREFLOW makes  $f$  a preflow.

**Maintenance:** The only operations within the **while** loop of lines 2–3 are push and relabel. Relabel operations affect only height attributes and not the flow values; hence they do not affect whether  $f$  is a preflow. As argued on page 739, if  $f$  is a preflow prior to a push operation, it remains a preflow afterward.

**Termination:** At termination, each vertex in  $V - \{s, t\}$  must have an excess of 0, because by Lemma 26.14 and the invariant that  $f$  is always a preflow, there are no overflowing vertices. Therefore,  $f$  is a flow. Lemma 26.16 shows that  $h$  is a height function at termination, and thus Lemma 26.17 tells us that there is no path from  $s$  to  $t$  in the residual network  $G_f$ . By the max-flow min-cut theorem (Theorem 26.6), therefore,  $f$  is a maximum flow. ■

### Analysis of the push-relabel method

To show that the generic push-relabel algorithm indeed terminates, we shall bound the number of operations it performs. We bound separately each of the three types of operations: relabels, saturating pushes, and nonsaturating pushes. With knowledge of these bounds, it is a straightforward problem to construct an algorithm that runs in  $O(V^2E)$  time. Before beginning the analysis, however, we prove an important lemma. Recall that we allow edges into the source in the residual network.

**Lemma 26.19**

Let  $G = (V, E)$  be a flow network with source  $s$  and sink  $t$ , and let  $f$  be a preflow in  $G$ . Then, for any overflowing vertex  $x$ , there is a simple path from  $x$  to  $s$  in the residual network  $G_f$ .



**Proof** For an overflowing vertex  $x$ , let  $U = \{v : \text{there exists a simple path from } x \text{ to } v \text{ in } G_f\}$ , and suppose for the sake of contradiction that  $s \notin U$ . Let  $\bar{U} = V - U$ .

We take the definition of excess from equation (26.14), sum over all vertices in  $U$ , and note that  $V = U \cup \bar{U}$ , to obtain

$$\begin{aligned}
 \sum_{u \in U} e(u) &= \sum_{u \in U} \left( \sum_{v \in V} f(v, u) - \sum_{v \in V} f(u, v) \right) \\
 &= \sum_{u \in U} \left( \left( \sum_{v \in U} f(v, u) + \sum_{v \in \bar{U}} f(v, u) \right) - \left( \sum_{v \in U} f(u, v) + \sum_{v \in \bar{U}} f(u, v) \right) \right) \\
 &= \sum_{u \in U} \sum_{v \in U} f(v, u) + \sum_{u \in U} \sum_{v \in \bar{U}} f(v, u) - \sum_{u \in U} \sum_{v \in U} f(u, v) - \sum_{u \in U} \sum_{v \in \bar{U}} f(u, v) \\
 &= \sum_{u \in U} \sum_{v \in \bar{U}} f(v, u) - \sum_{u \in U} \sum_{v \in \bar{U}} f(u, v).
 \end{aligned}$$

We know that the quantity  $\sum_{u \in U} e(u)$  must be positive because  $e(x) > 0$ ,  $x \in U$ , all vertices other than  $s$  have nonnegative excess, and, by assumption,  $s \notin U$ . Thus, we have

$$\sum_{u \in U} \sum_{v \in \bar{U}} f(v, u) - \sum_{u \in U} \sum_{v \in \bar{U}} f(u, v) > 0. \quad (26.17)$$

All edge flows are nonnegative, and so for equation (26.17) to hold, we must have  $\sum_{u \in U} \sum_{v \in \bar{U}} f(v, u) > 0$ . Hence, there must exist at least one pair of vertices  $u' \in U$  and  $v' \in \bar{U}$  with  $f(v', u') > 0$ . But, if  $f(v', u') > 0$ , there must be a residual edge  $(u', v')$ , which means that there is a simple path from  $x$  to  $v'$  (the path  $x \rightsquigarrow u' \rightarrow v'$ ), thus contradicting the definition of  $U$ . ■

The next lemma bounds the heights of vertices, and its corollary bounds the number of relabel operations that are performed in total.

**Lemma 26.20**

Let  $G = (V, E)$  be a flow network with source  $s$  and sink  $t$ . At any time during the execution of GENERIC-PUSH-RELABEL on  $G$ , we have  $u.h \leq 2|V| - 1$  for all vertices  $u \in V$ .

**Proof** The heights of the source  $s$  and the sink  $t$  never change because these vertices are by definition not overflowing. Thus, we always have  $s.h = |V|$  and  $t.h = 0$ , both of which are no greater than  $2|V| - 1$ .

Now consider any vertex  $u \in V - \{s, t\}$ . Initially,  $u.h = 0 \leq 2|V| - 1$ . We shall show that after each relabeling operation, we still have  $u.h \leq 2|V| - 1$ . When  $u$  is

reabeled, it is overflowing, and Lemma 26.19 tells us that there is a simple path  $p$  from  $u$  to  $s$  in  $G_f$ . Let  $p = \langle v_0, v_1, \dots, v_k \rangle$ , where  $v_0 = u$ ,  $v_k = s$ , and  $k \leq |V| - 1$  because  $p$  is simple. For  $i = 0, 1, \dots, k - 1$ , we have  $(v_i, v_{i+1}) \in E_f$ , and therefore, by Lemma 26.16,  $v_i.h \leq v_{i+1}.h + 1$ . Expanding these inequalities over path  $p$  yields  $u.h = v_0.h \leq v_k.h + k \leq s.h + (|V| - 1) = 2|V| - 1$ . ■

**Corollary 26.21 (Bound on relabel operations)**

Let  $G = (V, E)$  be a flow network with source  $s$  and sink  $t$ . Then, during the execution of GENERIC-PUSH-RELABEL on  $G$ , the number of relabel operations is at most  $2|V| - 1$  per vertex and at most  $(2|V| - 1)(|V| - 2) < 2|V|^2$  overall.

**Proof** Only the  $|V| - 2$  vertices in  $V - \{s, t\}$  may be relabeled. Let  $u \in V - \{s, t\}$ . The operation RELABEL( $u$ ) increases  $u.h$ . The value of  $u.h$  is initially 0 and by Lemma 26.20, it grows to at most  $2|V| - 1$ . Thus, each vertex  $u \in V - \{s, t\}$  is relabeled at most  $2|V| - 1$  times, and the total number of relabel operations performed is at most  $(2|V| - 1)(|V| - 2) < 2|V|^2$ . ■

Lemma 26.20 also helps us to bound the number of saturating pushes.

**Lemma 26.22 (Bound on saturating pushes)**

During the execution of GENERIC-PUSH-RELABEL on any flow network  $G = (V, E)$ , the number of saturating pushes is less than  $2|V||E|$ .

**Proof** For any pair of vertices  $u, v \in V$ , we will count the saturating pushes from  $u$  to  $v$  and from  $v$  to  $u$  together, calling them the saturating pushes between  $u$  and  $v$ . If there are any such pushes, at least one of  $(u, v)$  and  $(v, u)$  is actually an edge in  $E$ . Now, suppose that a saturating push from  $u$  to  $v$  has occurred. At that time,  $v.h = u.h - 1$ . In order for another saturating push from  $u$  to  $v$  to occur later, the algorithm must first push flow from  $v$  to  $u$ , which cannot happen until  $v.h = u.h + 1$ . Since  $u.h$  never decreases, in order for  $v.h = u.h + 1$ , the value of  $v.h$  must increase by at least 2. Likewise,  $u.h$  must increase by at least 2 between saturating pushes from  $v$  to  $u$ . Heights start at 0 and, by Lemma 26.20, never exceed  $2|V| - 1$ , which implies that the number of times any vertex can have its height increase by 2 is less than  $|V|$ . Since at least one of  $u.h$  and  $v.h$  must increase by 2 between any two saturating pushes between  $u$  and  $v$ , there are fewer than  $2|V|$  saturating pushes between  $u$  and  $v$ . Multiplying by the number of edges gives a bound of less than  $2|V||E|$  on the total number of saturating pushes. ■

The following lemma bounds the number of nonsaturating pushes in the generic push-relabel algorithm.

**Lemma 26.23 (Bound on nonsaturating pushes)**

During the execution of GENERIC-PUSH-RELABEL on any flow network  $G = (V, E)$ , the number of nonsaturating pushes is less than  $4|V|^2(|V| + |E|)$ .

**Proof** Define a potential function  $\Phi = \sum_{v:e(v)>0} v.h$ . Initially,  $\Phi = 0$ , and the value of  $\Phi$  may change after each relabeling, saturating push, and nonsaturating push. We will bound the amount that saturating pushes and relabelings can contribute to the increase of  $\Phi$ . Then we will show that each nonsaturating push must decrease  $\Phi$  by at least 1, and will use these bounds to derive an upper bound on the number of nonsaturating pushes.

Let us examine the two ways in which  $\Phi$  might increase. First, relabeling a vertex  $u$  increases  $\Phi$  by less than  $2|V|$ , since the set over which the sum is taken is the same and the relabeling cannot increase  $u$ 's height by more than its maximum possible height, which, by Lemma 26.20, is at most  $2|V| - 1$ . Second, a saturating push from a vertex  $u$  to a vertex  $v$  increases  $\Phi$  by less than  $2|V|$ , since no heights change and only vertex  $v$ , whose height is at most  $2|V| - 1$ , can possibly become overflowing.

Now we show that a nonsaturating push from  $u$  to  $v$  decreases  $\Phi$  by at least 1. Why? Before the nonsaturating push,  $u$  was overflowing, and  $v$  may or may not have been overflowing. By Lemma 26.13,  $u$  is no longer overflowing after the push. In addition, unless  $v$  is the source, it may or may not be overflowing after the push. Therefore, the potential function  $\Phi$  has decreased by exactly  $u.h$ , and it has increased by either 0 or  $v.h$ . Since  $u.h - v.h = 1$ , the net effect is that the potential function has decreased by at least 1.

Thus, during the course of the algorithm, the total amount of increase in  $\Phi$  is due to relabelings and saturated pushes, and Corollary 26.21 and Lemma 26.22 constrain the increase to be less than  $(2|V|)(2|V|^2) + (2|V|)(2|V||E|) = 4|V|^2(|V| + |E|)$ . Since  $\Phi \geq 0$ , the total amount of decrease, and therefore the total number of nonsaturating pushes, is less than  $4|V|^2(|V| + |E|)$ . ■

Having bounded the number of relabelings, saturating pushes, and nonsaturating pushes, we have set the stage for the following analysis of the GENERIC-PUSH-RELABEL procedure, and hence of any algorithm based on the push-relabel method.

**Theorem 26.24**

During the execution of GENERIC-PUSH-RELABEL on any flow network  $G = (V, E)$ , the number of basic operations is  $O(V^2E)$ .

**Proof** Immediate from Corollary 26.21 and Lemmas 26.22 and 26.23. ■

Thus, the algorithm terminates after  $O(V^2 E)$  operations. All that remains is to give an efficient method for implementing each operation and for choosing an appropriate operation to execute.

**Corollary 26.25**

There is an implementation of the generic push-relabel algorithm that runs in  $O(V^2 E)$  time on any flow network  $G = (V, E)$ .

**Proof** Exercise 26.4-2 asks you to show how to implement the generic algorithm with an overhead of  $O(V)$  per relabel operation and  $O(1)$  per push. It also asks you to design a data structure that allows you to pick an applicable operation in  $O(1)$  time. The corollary then follows. ■

**Exercises**

**26.4-1**

Prove that, after the procedure `INITIALIZE-PREFLOW`( $G, s$ ) terminates, we have  $s.e \leq -|f^*|$ , where  $f^*$  is a maximum flow for  $G$ .

**26.4-2**

Show how to implement the generic push-relabel algorithm using  $O(V)$  time per relabel operation,  $O(1)$  time per push, and  $O(1)$  time to select an applicable operation, for a total time of  $O(V^2 E)$ .

**26.4-3**

Prove that the generic push-relabel algorithm spends a total of only  $O(VE)$  time in performing all the  $O(V^2)$  relabel operations.

**26.4-4**

Suppose that we have found a maximum flow in a flow network  $G = (V, E)$  using a push-relabel algorithm. Give a fast algorithm to find a minimum cut in  $G$ .

**26.4-5**

Give an efficient push-relabel algorithm to find a maximum matching in a bipartite graph. Analyze your algorithm.

**26.4-6**

Suppose that all edge capacities in a flow network  $G = (V, E)$  are in the set  $\{1, 2, \dots, k\}$ . Analyze the running time of the generic push-relabel algorithm in terms of  $|V|$ ,  $|E|$ , and  $k$ . (*Hint*: How many times can each edge support a nonsaturating push before it becomes saturated?)

**26.4-7**

Show that we could change line 6 of INITIALIZE-PREFLOW to

$$6 \quad s.h = |G.V| - 2$$

without affecting the correctness or asymptotic performance of the generic push-relabel algorithm.

**26.4-8**

Let  $\delta_f(u, v)$  be the distance (number of edges) from  $u$  to  $v$  in the residual network  $G_f$ . Show that the GENERIC-PUSH-RELABEL procedure maintains the properties that  $u.h < |V|$  implies  $u.h \leq \delta_f(u, t)$  and that  $u.h \geq |V|$  implies  $u.h - |V| \leq \delta_f(u, s)$ .

**26.4-9 ★**

As in the previous exercise, let  $\delta_f(u, v)$  be the distance from  $u$  to  $v$  in the residual network  $G_f$ . Show how to modify the generic push-relabel algorithm to maintain the property that  $u.h < |V|$  implies  $u.h = \delta_f(u, t)$  and that  $u.h \geq |V|$  implies  $u.h - |V| = \delta_f(u, s)$ . The total time that your implementation dedicates to maintaining this property should be  $O(VE)$ .

**26.4-10**

Show that the number of nonsaturating pushes executed by the GENERIC-PUSH-RELABEL procedure on a flow network  $G = (V, E)$  is at most  $4|V|^2|E|$  for  $|V| \geq 4$ .

## ★ 26.5 The relabel-to-front algorithm

The push-relabel method allows us to apply the basic operations in any order at all. By choosing the order carefully and managing the network data structure efficiently, however, we can solve the maximum-flow problem faster than the  $O(V^2E)$  bound given by Corollary 26.25. We shall now examine the relabel-to-front algorithm, a push-relabel algorithm whose running time is  $O(V^3)$ , which is asymptotically at least as good as  $O(V^2E)$ , and even better for dense networks.

The relabel-to-front algorithm maintains a list of the vertices in the network. Beginning at the front, the algorithm scans the list, repeatedly selecting an overflowing vertex  $u$  and then “discharging” it, that is, performing push and relabel operations until  $u$  no longer has a positive excess. Whenever we relabel a vertex, we move it to the front of the list (hence the name “relabel-to-front”) and the algorithm begins its scan anew.