## Simple Network Simplex Algorithm for Max-Flow

Network simplex is an augmenting path algorithm. Each step of the algorithm (pivot) may augment flow on a path and may rearrange the basis (defined below). A pivot will do one, the other, or both. Pivots that do not augment are called degenerate.

The algorithm maintains a flow (either explicitly or using residual capacities) and two trees, an out-tree S rooted at the source s and an in-tree T rooted at the sink t. Every vertex is either in S or in T. The basis B is the union of S and T. The version of the algorithm we consider maintains the invariant that every arc in S is residual (S arcs are directed away from s).

A pivot arc is a residual arc from S to T. If there is no pivot arc, (S,T) is a cut and the current flow is maximum.

: Choose any pivot arc . Let P be the path obtained by concatenating the s-v path in S, (v,w), and the w-t path in T, ( is the unique path from to of ). If P is a residual path, we augment flow, saturating at least one arc on P. Otherwise (degenerate pivot) P already contains a saturated arc. Let be the first saturated arc on P. The arc is an entering arc and is a leaving arc. We update B by replacing by . Note that if and are the same, B does not change; in this case, got saturated, so we did augment flow. There are three cases to consider: is in , is in or . From now on assume is in . Other cases are similar.  
  
Now we describe the operation for the case in S (If is in the operation is symmetric, and if , there is nothing to do): The sub-tree of rooted at needs to changes its root to so it becomes a sub-tree of . This operation updates the fields in order to maintain a DFS ordering and reverses the pointers on the to sub-path of and sets ’s parent to .  
To describe this in more detail we need to define two basic operations , and .  
  
: To delete from a tree Q a sub-tree rooted at u: start at u, follow next pointers for the SIZE steps to find the last vertex, remove the sublist corresponding to the sub-tree (hence the need of a doubly linked list). Then go from u up to the root of Q updating sub-tree sizes.   
: To add to a tree R a sub-tree as a child of u: insert the sub-tree list in the tree list immediately after u, go up updating sub-tree sizes.  
  
In the case in S, call the sub-tree of rooted at . And the same sub-tree rooted at . We can now describe how to transform into .   
   
Delete from S the sub-tree rooted at y and call it Q (Disconnect Q from S)  
Delete from Q the sub-tree rooted at v and call it R.  
Set z = v  
while z!=y   
set p=(old) parent of z  
Delete from Q the sub-tree rooted at p and add it to R as a child of z  
make z the parent of p  
set z=p  
repeat  
Make w the parent of v.  
Finally R has the same elements as Q but is rooted at v, we can connect R to T:  
Add R to T as a child of w.   
  
Note that augmentation can saturate several tree arcs, so we have a choice of leaving arcs. A choice rule we use (which is known to prevent cycling) is to choose the arc closest to s. Note that this maintains the invariant that S tree arcs directed away from s are residual (if this invariant holds for the initial tree).

One way to initialize the tree is to build a BFS tree S out of s in the residual graph with t deleted. We can delete vertices other than t which are not in S (they can be reached only via t) and make T = {t}.

## New GGT based Simplex Algorithm for Max-Flow

Some terminology:

An arc is a pivot arc if and is residual  
An arc is a tree arc if which means u’s parent pointer is v or v’s parent pointer is u. An arc is pseudo-residual if it is a residual arc or a tree arc.  
Let if uv is pseudo-residual, or otherwise. It is the trivial length on the pseudo-residual graph.  
An arc is admissible if it is pseudo-residual and (it is going 1 step uphill).  
An arc is reverse admissible if its reverse arc is admissible.  
The labeling is valid if for every pseudo-residual arc , (a pseudo-residual arc cannot go more than 1 step uphill). Implies tree arcs can only go 1 step up/down or stay flat.  
 is correct/exact/true if for every vertex the SP from to has (pseudo-residual) length   
Each vertex has a current arc pointer: .  
 is current if its current arc is the first (lowest index) reverse admissible arc on its arcs list.  
So if the labeling is valid, for a given vertex say , the vertex is current if and only if all incoming pseudo-residual arcs , with a of smaller index than ’s current arc satisfy and   
We maintain 2 priority queues of vertices indexed by all possible vertex labels (from 0 to n).  
Qr will be used to store non-current vertices that need to be relabeled.  
Qt will be used to store all current vertices of T.  
Call Mr and Mt the minimum labels of Qr and Qt respectively with the convention if

Operations

: Select a pivot arc with minimal . To do this, extract with min d(w) of Qt and vw the reverse arc of cur(w) should be a pivot arc with minimal d(v). Let P be the path obtained by concatenating the s-v path in S, (v,w), and the w-t path in T. Augment flow on until an arc is saturated, let be the first saturated arc on P so that is closest to . The arc is an entering arc and is a leaving arc. Three cases: is in , is in or .  
If is in S, move the sub-tree rooted at y of S into the sub-tree rooted at v of T using change\_root. Add to Qt all vertices of the sub-tree that are current. (and reinsert w to Qt if deleted by extract-min of Qt.)  
If is in T, move the sub-tree rooted at x of T into the sub-tree rooted at w of S using change\_root. Delete from Qt all elements of this sub-tree. Delete y from Qt (since we will call make\_cur on it). (and delete w from Qt if not done by extract-min of Qt.)  
If , do nothing yet.  
Afterwards call on y, and then start relabeling elements of Qr using extract min repeatedly.  
In basic variant 0, stop when that is .  
In lazy variant 1, stop when .  
In lazy variant 2, stop when and .

advances until it is reverse admissible, that is until is pseudo-residual and or until cur(v) is invalid (cur(v)= v+1’s first arc)  
If a reverse admissible arc is found: If v is in T, adds v to Qt and returns true, otherwise:  
cur(vu) is invalid, v is added it to Qr, If v is in T, deletes v from Qt, and returns false.  
(If a reverse admissible arc is found, is current. Otherwise cannot be made current.)

operation finds the minimum of for all pseudo-residual arcs into , sets to the minimum, and sets to the first reverse admissible arc on its list .  
(After relabeling, becomes current. Increasing maintains a valid labeling. Relabeling may make non-current some of its neighbors so we need to fix them.) For each outgoing arc . If then call .

The algorithm stops when Qt is empty. (no pivots left)

**make\_cur properties**

d properties are invariants of since it does not affect d

If d is valid, is current if and only if all incoming pseudo-residual arcs , with a of smaller index than satisfy and .

If d is valid, if after calling make\_cur, a reverse admissible arc is found, is current. Otherwise if make\_cur fails, cannot be made current.

make\_cur(v) takes finite time since v has finite number of arcs

**Relabel properties**

is only called on vertices extracted from Qr, after a failed

If d is valid, increases d(v) strictly.  
Proof: a relabel(v) can only occur after a failed make\_cur(v) so it means for all since d is valid. So when d(v) is set to the min of d(u)+1’s it increases.

d is valid is an invariant of   
Proof: For incoming pseudo-residual arcs of v, after relabeling of v, since d(v) is the min of the d(u)+1.  
For outgoing pseudo-residual arcs of v, before relabel so after relabel it is maintained since d(v) increases.

relabel(v) takes finite time

**Pivot invariants:** (True for basic variant, lazy variant 1, lazy variant 2)

Before pivot, assume the properties:  
1. S arcs are residual  
2. d is valid  
3.   
4. the reverse current arc of w the min of Qt is a pivot arc  
5. , equivalently If v is not in Qt nor in Qr, v is in S  
6. Label<=Mt-1 vertices are in S, are current, and have correct distances.  
7. Qt is the set of all current vertices of T  
8. Qr is the set of all non-current vertices.  
9. Vertices whose label reach during relabeling can be deleted.  
10. Pivot takes finite time.  
11. There remain pivot arcs (the algorithm is not finished) if and only if   
  
After pivot they are maintained.  
Proof:

1. S arcs are residual:  
During a pivot the leaving saturated arc is chosen so that is closest to .

2. d is valid:  
The only operation affecting d is which maintains validity.

3.  
In basic variant after pivot we stop relabeling when and in lazy variant we stop when

4. the reverse current arc of w the min of Qt is a pivot arc:  
w is in Qt so vw is admissible, so v is not in Qt nor in Qr, so v is in S, so vw is in SxT and necessarily residual since not a tree arc.

5. , equivalently If v is not in Qt nor in Qr, v is in S  
Every non-current vertex (in T or S) is in Qr and every current T vertex is in Qt.

6. Label<=Mt-1 vertices are in S, are current, have correct distances.  
If , v is not in Qt nor in Qr, so v is in S. v is not in Qr so v is current.  
So this is true for all label<=Mt-1 vertices. Now we can show v has correct distance:  
Going back from v to s following pointers defines a path of length which is shortest because the valid labeling condition implies that any (pseudo-)residual path cannot climb more than 1 step up per arc but it must reach from 0 so it needs at least steps.

7. Qt is the set of all current vertices of T  
8. Qr is the set of all non-current vertices  
9. Vertices whose label reach during relabeling can be deleted.  
10. Pivot takes finite time  
Lemma: During a pivot, the only vertex that was potentially made non-current is .  
 is no longer a pseudo-residual arc, but could be ’s admissible arc so can be non-current now.   
 also needs to be considered carefully because it is possible that was not pseudo-residual before the pivot. However cannot be reverse admissible arc of because is admissible. All other vertices in the moving sub-tree remain current since their current arcs remain tree arcs if they were or residual arcs. All other vertices remain current.  
After a pivot we call . What make\_cur does ensures Qt remains the set of all current vertices of T.  
At this point all new non-current vertices are in . We extract an element of Qr and call until the variant criterion is met.  
For vertices whose label reach , we know that there will never be a finite length residual path from to . (it is on side of a minimum cut.), since relabeling only increase labels, so we can delete them.  
This relabeling must end because strictly increases labels, and vertices with label reaching n are deleted.

11. There remain pivot arcs (the algorithm is not finished) if and only if   
Proof: If the reverse current arc of w the min of Qt is a pivot arc. If , then so (all its elements were deleted because their label reached n).

In the basic variant, we have the following more specific invariants:  
3. , (  
5.   
6. Label<=Mt-1 are in S. All vertices are current and have correct distances.

In the lazy variant 2, we have the following more specific invariants:  
5. is partitioned in Qt and Qr, equivalently exactly 1 of holds  
6. Label<=Mt-1 are in S. All S vertices are current and have correct distances.

Observation:

In the lazy variant 1 when a subtree moves from S to T, we need to:  
add to Qt all vertices of the sub-tree that are current, so we need to check if they are current.  
In the basic variant and lazy variant 2, all S vertices are current, so we do not need to do this check so we can add to Qt all vertices of the sub-tree. However it can be made cheap to check if a vertex is current (checking it is not in Qr).

Algorithm initialization:   
One way to initialize the tree is to build a BFS tree S out of s in the residual graph with t deleted. We can delete vertices other than t which are not in S (they can be reached only via t) and make T = {t}.  
Then we can do a global relabel to initialize distances correctly.  
Setting all labels to 0 except sink to 1 is not enough because we need the correct distance assumption.  
A global relabel does a BFS from source to assign true distance labels and make all vertices current.

Global relabel heuristic

Gap relabel heuristic