1 Determine the saturation of the current flow

1.1 Algorithm

In short, the algorithm firstly generates the residual graph G_f of the current flow f on the network G. Then it checks if there's a $s-t$ path in G_f. If there's no such path, f is the max flow. Otherwise, f is not the max flow. Here are details of the algorithm.

1. Compute the residual network
   The idea is the same as constructing a residual network from the lecture slides.
   1.1. copy the node set of G to G_f
   1.2. for each edge $e := (u, v)$ of G
      - if $f(e) < c(e)$, copy (u, v) to G_f with a residual capacity of $c(e) - f(e)$
      - if $f(e) > 0$, add a new edge (v, u) to G_f with a residual capacity of $f(e)$
2. Attempt to find a simple $s-t$ path in G_f(augmenting path)
   Concretely, the algorithm uses a breadth first search for G_f starting from s. The BFS is terminated if t is visited. As BFS is a common algorithm, the details are omitted here. The BFS is terminated after all the nodes reachable from s have been visited.
   - If t is in the list of visited nodes, there's a simple $s-t$ path in G_f, in other words, f is not the max flow.
   - Otherwise, there's no such path, therefore, f is the max flow.

1.2 Correctness Analysis

- Generate the residual network
  Copying nodes is trivially feasible as the network G is given. Creating forward/backward edges in G_f is also feasible as "*the network you're given is connected, that the capacities are all positive integers, and that the flow you're given is feasible and integtal.*"
- The equivalence of no $s-t$ path in G_f and optimality(max flow)
  The correctness of using a BFS to find a simple path is omitted, again, as BFS is a well known algorithm. In addition, you can refer to week 6 slides about proving "*No Augmenting Path in $G_f \Rightarrow f$ is Max Flow*", in other words, "*If f is an $s-t$ flow such that there is no $s-t$ path in the residual graph G_f, then f has the maximum value of any flow in G*".

1.3 Time Complexity Analysis

1. Construction of the residual network costs $O(m) + O(n) \in O(m+n)$ because

1.1. copying n nodes: $O(n)$

1.2. In the worst cases, each edge in G turns into two edges in G_f. Therefore, $2m \in O(m)$ in total for the worst scenario

2. retrieve the augmenting path: $O(m+n) + O(n) \in O(m+n)$

2.1. BFS whole G_f: $O(m+n)$ because "*the flow's residual graph in adjacency list form*".

2.2. Membership of t in the visited list: $O(n)$ if the visited list is implemented as a linked list.

In total, $2O(m+n) \in \boxed{\text{O(m+n)}}$

2 Improve distributing network

2.1 Existence

2.1.1 *expandable* edge

No.
Statement: There exists a network that has no expandable pipes.

$$\exists G(V,E) : \forall e \in E : e \text{ is not } \textit{expandable}$$

Proof: by giving a concrete instance:
$G(V,E)$ where $V = \{s,m,t\}, E = \{(s,m),(m,t)\}, c((s,m)) = 1, c((m,t)) = 1$.
where c is the capacity mapping $E \to \mathbb{Z}^+$.
It is trivial to show that all e in E of G are not expandable as

$$\text{max flow} = \max(v(f)) = 1$$

No matter if either $c((s,m)) = 2$ or $c((m,t)) = 2$, $\max(v(f))$ is still 1.

2.1.2 *limiting* edge

Yes.
Statement: Every network has at least one limiting pipe.

$$\forall G : \exists e \in E : e \text{ is } \textit{limiting}$$

Proof: By the integrality theorem from the lecture and that "*the capacities are all positive integers*", we know that every such network has a flow max.

$$\forall G : \exists \max(v(f)) := v(f_{max})$$

Further more, by the Ford-Fulkerson theorem, we know that

$$v(f_{max}) = \text{cap}(A,B)$$

where (A, B) is the min-cut partition of V with $s \in A$ and $t \in B$. Explicitly,

$$\text{cap}(A, B) = \sum_{e=(u,v)\in E, u\in A, v\in B} c(e)$$

The theorem also implies that $v(f_{max}) > 0$ since suppose that $v(f_{max}) = 0$ then $\text{cap}(A, B) = \sum_{e=(u,v)\in E,u\in A,v\in B} c(e) = 0$. It either leads to

$$\forall e \in \{(u, v) \in E \mid u \in A, v \in B\} \neq \emptyset : c(e) = 0$$

which contradicts to the assumption "*the capacities are all positive integers*" or even

$$\{(u, v) \in E \mid u \in A, v \in B\} = \emptyset$$

which in consequence makes $t \in A$ violating the assumption that $t \in B$.
Now, we have proved that there exists a min-cut (A, B) where there is at least one pipe going from A to B. We name this pipe as e_i.
Suppose we decrease $c(e_i)$ by 1 to be $c'(e_i)$. As a consequence, $\text{cap}'(A, B)$ is decreased by 1. Again by the min-cut max-flow theorem, we have $v(f'_{max}) = v(f_{max}) - 1$.Thus, e_i is a *limiting* pipe. We have proved the existence of a limiting pipe(an instance e_i) in any network of those given by the assignment.

2.2 all *expandable* pipes

2.2.1 Algorithm

The main idea is to find all cut edges of a min-cut partition firstly and then check whether the partition is the only min-cut partition of the network. If the min-cut (A, B) where $s \in A, t \in B$ is unique, all *expandable* edges are exactly those outgoing edges from A. Otherwise, there's no *expandable* edges at all. Here is the detailed implementation.

1. Compute all cut edges of a min-cut partition
   1.1. Use a BFS in the residual graph G_f staring from s. Put all visited nodes in a set A.
   1.2. Put the unvisited nodes in set B.
   1.3. Iterate through all edges $e := (u, v) \in E$ of $G(V, E)$. If u is in A and v is in B, put e in a set C.
2. Check if the computed partition is unique
   2.1. Choose the first edge e_{cut} from C.
   2.2. Modify the residual graph G_f by adding a forward edge with a residual capacity of 1.($c_f(e_{\text{cut}}) = 1$)
   2.3. Use BFS to find a $s - t$ path in the modified residual graph G'_f
3. List *expandable* edges
   - If there's a $s - t$ path found in G'_f, return the set C
   - Otherwise, return an empty set.

2.2.2 Correctness Analysis

1. Compute all cut edges of a min-cut partition
By the max-flow min-cut theorem, it is guaranteed that there's no $s-t$ path in G_f if f is a max flow. Therefore, there must exsit a min cut partition (A,B) where $s \in A, t \in B$. Further more, all nodes in A are reachable by s while those in B are not. If a node in A is not reachable, it is impossive that the node has been visited during the BFS. The logic holds for nodes in B similarly. The set C is exactly the set of all cut edges(outgoing edges from A) for the partition (A,B) because of the definition of a directed graph. In summary, we have proved that there's such a partition and we've found all cut edges of it.

2. **Claim 1:** The set returned is a set of all *expandable* edges.
To prove the claim, let us prove the following claims firstly.

   - **Claim 2:** If there are more than 1 min-cut partitions, there's no *expandable* edges.
   Proof: In other words, we are give that there exist at least two different min-cut partitions (A,B) and (A',B'). Again by the min-cut max-flow theorem, $\text{CAP}(A,B) = \text{CAP}(A',B') = v(f_{max})$. Suppose there is an *expandable* edge e, by the definition, if $c(e)$ is increased by 1, max flow value is increased by 1 as a result. That is, $v(f'_{max}) = v(f_{max}) + 1$. As an other consequence, both $\text{CAP}(A,B)$ and $\text{CAP}(A',B')$ are increased by 1 otherwise there's an inequivalence of min-cut and max-flow. Therefore, the capacity of an outgoing edge in (A,B) has to increase by 1, so does that in (A',B'). This contradicts the assumption that we have increased the capacity of only one edge in the network.

   - **Claim 3:** If there is exactly 1 min-cut partition, all cut edges of this partition are *expandable* edges while the other edges in the network are not.
   Proof: Given that (A,B) is the unique min-cut partition,

   $$\forall (A',B') \neq (A,B) : \text{CAP}(A',B') > \text{CAP}(A,B) = \text{max flow value}$$

     – the cut edges of the unique partition
     Let's say $e_{\text{to increase}}$ is such a cut edge.

     $$\begin{aligned} & c'(e_{\text{to increase}}) = c(e_{\text{to increase}}) + 1 \\ \Rightarrow & \text{CAP}'(A,B) = \sum_{e \text{ is an outgoing edge of} (A,B)} c'(e) \\ & = c(e_{\text{to increase}}) + 1 + \sum_{\substack{e \text{ is an outgoing edge of}(A,B) \\ \text{other than } e_{\text{to increase}}}} c(e) \\ & = \text{CAP}(A,B) + 1 \leq \text{CAP}'(A',B') = \text{CAP}(A',B') \end{aligned} \tag{1}$$

     Thus, the new max flow value is

     $$\min(\text{CAP}'(A,B), \{\text{CAP}'(A',B')\}) = \text{CAP}'(A,B) = \text{CAP}(A,B) + 1$$

     We have proved that any such a cut edge is *expandable*.

  - the other edges
    The other edges are not *expandable* trivially as $\text{CAP}(A, B)$ remains the same so the max flow value is the same.

  Now, we have covered all situations of the number of the min-cut partitions(0 is impossible as we have showed the existence in the previous section). The final piece to complete the overall proof is that

- **Claim 4:** The existence of the $s - t$ path in algorithm step 2.3 is equivalent to the uniqueness of the min-cut partition.
  Proof: Step 2.1 is viable since there must be at least one outgoing edge for the min cut partition (A, B), where $v(f_{\max}) = \text{CAP}(A, B)$. Otherwise, G is not connected. Before step 2.2, there is no forward edges in G_f from A to B otherwise (A, B) is not a min-cut partition. Let's notate the chosen edge e_{cut} in step 2.1 as (u, v) where $u \in A$ and $v \in B$. After step 2.2, we know that there's a $s - v$ path in G'_f since there's a forward edge. Therefore, (A, B) is no longer a min-cut partition. In step 2.3, if there is not found a $s - t$ path in G'_f, we know that there is at least one min-cut partition (A', B') of which $\text{CAP}(A', B') = \text{CAP}(A, B)$ and this partition is not (A, B). By finding another min-cut partition instance, we have shown that (A, B) is not unique. If there's an augmenting path in G'_f,

  $$\min(\{\text{CAP}(A', B')\}, \text{CAP}'(A, B)) = v(f'_{\max}) > v(f_{\max}) = \text{CAP}(A, B)$$

  Therefore, $\forall (A', B') \neq (A, B) : \text{CAP}(A', B') > \text{CAP}(A, B)$. In other words, (A, B) is the only min-cut partition.

Combining **Claim 2-4**, we have shown that **Claim 1** is valid.

2.2.3 Time Complexity Analysis

$|V| = n, |E| = m$

1. Finding all cutedges of a min-cut partition uses

$$O(n) + O(n + m) + O(m) \in O(n + m)$$

   since

   1.1. set A and B can be implemented as an array M of boolean values indicating being visited or not. The array is of size n and each index is mapped to a node v in G. This costs $O(n)$

   1.2. BFS costs $O(n + m)$.

   1.3. m iterations of checking membership by referring to the value of the target index in array M(a memory reference costs $O(1)$). $O(m)$ for the iterations.

2. 2.1 and 2.2 use constant time. BFS costs $O(n + m)$. Thus checking the uniqueness costs $O(n + m)$ in total.

3. returning *expandable* edges uses a constant time.

The algorithm uses $O(n + m) + O(n + m) + O(1) \in O(n + m)$. Moreover, since the graphs are connected, $m > n - 1$. Therefore, $O(n) \in O(m)$. We can simplify the bound $O(n + m)$ as $\boxed{O(m)}$

2.3 max flow value k feasibility

2.3.1 Algorithm

The algorithm finds the $s-t$ path P with the fewest number of edges in G. Calculate the total capacity of the network. Set the capacities of all edges to be 1. Attempt to redistribute the remaining capacities evenly to edges on P(There could be some spare capacity $c_{\text{remainder}}$). Calculate the max flow value of the redistributed network and compare it with k. Here are the details.

1. Find the $s-t$ path P with the fewest number of edges using a BFS in G.

2. Redistribute capacities

   2.1. sum all edge capacities in G and store it to C.

   2.2. copy the topology of the network and set all edge capacities to be 1.

   2.3. add capacities of edges along P by floor($\frac{C-m}{|P|}$), where floor() is a floor function and $|P|$ is the number of edges along P.

3. Calculate the max flow value of the redistributed network $v(f_{\text{redistributed}})$.
   It is simply the sum of the number of the outgoing edges of the min-cut (A, B) and the number added in step 2.3.

$$v(f_{\text{redistributed}}) = \sum_{e=(u,v)\in E, u\in A, v\in B} 1 + \text{floor}(\frac{C-m}{|P|})$$

4. Compare it with k
   If $v(f_{\text{redistributed}}) \geq k$, return true. Otherwise, return false.

2.3.2 Correctness Analysis

- BFS always returns a path with fewest number of edges because BFS visits all nearest nodes before visting nodes adjacent to the nearest nodes.

- For step 2-4, we use an induction to illustrate.
  We notate the sum of capacities of the edges in a network G as c_{total}.
  Base Case: Suppose that we are given a network G with edge capacities of 1 and there exists at least one $s-t$ path in G. In addition, we are given a max flow f_0 and its residual graph G_f. It is trivial that there's at least a $s-t$ path along which each edge has a flow of value 1. To increase G's max flow value by 1, all edge flow values have to be increased by 1 on one $s-t$ path P so that conservation of flow is not violated. To fulfill the capacity restriction, the capacities of the edges along P have to be 2. It is straightforward that a longer P leads to a larger c_{total}. Therefore, we choose the P of the fewest number of edges. Now the edges along P have capacities and flows of 2. The rest edges remain the same. The new capacity sum $c'_{\text{total}} = \sum_{e \text{ along } P}(1) + c_{\text{total}}$ is optimal.
  Step Case: Now we are given a network G with a max flow f and a path P. $\forall e$ along $P : c(e) = f(e) = i > 1 \in \mathbb{Z}^+$ and $\forall e$ not along $P : c(e) = 1, f(e) = 0$ or 1. In addition, any $s-t$ path p along which flows have the same value has

more or equal edges than P. Again we can choose a p of the fewest possible number of edges. In this case, P satisfies such a requirement. The capacity distribution remains the same except that $\forall e$ along $P : c'(e) = f'(e) = i + 1$. The new total capacity $c'_{\text{total}} = \sum_{e \text{ along } P}(1) + c_{\text{total}}$ is optimal while $v(f') = v(f) + 1$.
It is obvious that the redistributed network in the base case is one of the Gs given in this case; Specially, $\forall e$ along $P : c(e) = f(e) = 2$. Therefore, the network, of which the capacities of edges along P are modified by such a way to increase the max flow value by 1, has an optimally small c_{total}.
Besides, we have induced such a relation

$$\Delta(c_{\text{total}}) = \Delta(v(f)) \times |P|$$

Suppose that $\sum_{e \ inE} c(e) := C$ we then have

$$C - m - c_{\text{remainder}} \geq |P| \times (v(f_{\text{redistributed}}) - v(f_0))$$

Because C is a constant(max total capacity to redistribute), the topology of the network remains the same(m, $c_{\text{remainder}}$ and $|P|$ are constant) and capacities are positive(f_0 is the base case.), $v(f_{\text{redistributed}})$ is the largest value we can have. Now we only need to check the correctness of calculating $v(f_{\text{redistributed}})$.

- The topology is the same so that we can choose the same partition (A, B) from the network of the origninal distribution(before redistribution). During each iteration of increasing 1 max flow value, the capacity mapping remains the same except that the capacity values of the edges along the chosen path with fewest number of edges P are increased by 1. In other words, $CAP(A, B)$ is only affected by the edge e which is along P and is one of the outgoing edges of (A, B). $e = \{e = (u, v) | u \in A, v \in B\} \cap \{e | e \text{ along } P\}$. So

$$v(f_{\text{redistributed}}) = \text{CAP}(A, B)_{\text{redistributed}} = \text{CAP}(A, B)_{\text{all capacities of 1}} + \text{floor}(\frac{C - m}{|P|})$$

where

$$\text{CAP}(A, B)_{\text{all capacities of 1}} = \sum_{e=(u,v) \in E, u \in A, v \in B} 1$$

2.3.3 Time Complexity Analysis

1. BFS uses $O(m + n)$
2. $O(m) + O(m + n) + O(m) + O(m)O(1) \in O(m + n)$
   2.1. $O(m)$ to get capacities and sum them.
   2.2. $O(m + n)$ to duplicate the network topology. $O(m)$ to initialize edge capacities.
   2.3. $O(m)$ if P contains all edges. The calculation is of $O(1)$.
3. compute the outgoing edges of the original min-cut costs $O(n + m)$, which has been shown in section 2.2.3. Arithmetic calculation is of $O(1)$

4. value comparison uses $O(1)$

In total, the algorithm uses $O(n+m)$. Similar to the previous problem, we can simplify the bound to $O(m)$.