CSE 6331 HOMEWORK 9

BRENDAN WHITAKER

1. *Consider a flow network in which vertices, as well as edges, have capacities. In addition to the original edge capacity constraint, there is now a new vertex capacity constraint: the total positive flow entering any vertex u cannot exceed its capacity $c(u)$. Show that determining the maximum flow in a network with edge and vertex capacities can be reduced to an ordinary maximum flow problem.*

   Proof. We simply replace each vertex u with a pair of vertices v_1, v_2 and an edge $(v_1, v_2) \in E$ between them such that $c(u) = c(v_1, v_2)$. All incoming edges to u now enter v_1, and all outgoing edges from u now exit v_2. $\square$

2. *Suppose that during an execution of Relabel-to-Front, Discharge(u) is called* ***twice*** *for some particular node u. Prove that if an edge (u, v) is inadmissible at the end/exit of the first Discharge(u), then it is still inadmissible at the beginning/entry of the second Discharge(u).*

   Proof. Let (u, v) be inadmissible at the exit of the first Discharge(u). Then either $c_f(u, v) = 0$ or $h(u) \leq h(v)$.

   Case 1: Suppose $c_f(u, v) = 0$. Then after we exit the first Discharge(u), we could execute a push from v to u before we come back to the second Discharge(u), or we could not. Suppose we push from v to u before starting the second Discharge(u). Then we know that in order to push from v to u, we had to have $h(v) = h(u) + 1$. So since the height of u will not change since we are not discharging u, we know that when we enter Discharge(u) for the second time $h(v) \geq h(u) + 1$, since $h(v)$ never decreases. But then we know that at the start of the second Discharge(u), we have $h(u) \leq h(v) - 1 < h(v) + 1$, hence we must have that (u, v) is inadmissible.

   Case 2: Suppose $c_f(u, v) > 0$. Then we must have that $h(u) \leq h(v)$ at the exit of the first Discharge(u) since we said (u, v) is inadmissible. This is because since (u, v) is a residual edge, we know $h(u) \leq h(v) + 1$, and we can't have $h(u) = h(v) + 1$ since this would violate the assumption that (u, v) is inadmissible. Now since $h(u)$ does not change if we are not in Discharge(u), and $h(v)$ can only increase, we know that when we come back to the start of the second Discharge(u), $h(u) \leq h(v)$ still, so (u, v) is still inadmissible. $\square$

3. *Let $G = (V, E)$ be a flow network with source s, sink t, and integer capacities. Suppose we are given a maximum flow f in G, and suppose the capacity of a single edge $(u^*, v^*) \in E$ is* ***increased*** *by 1. Give an $O(V + E)$-time algorithm to update the maximum flow.*

Date: SP18.

Algorithm 1: Update-Flow(G)

Data: The set of vertices V and edges E of the given flow network G. A maximum flow f in G.

Result: The updated maximum flow f.

```
begin
    /* We compute the residual network E_f of G.                          */
    for (u, v) ∈ E do
        if c(u, v) − f(u, v) > 0 then
            add (u, v) to E_f;
        end
    end
    /* We check if (u*, v*) ∈ E_f.                                         */
    if (u*, v*) ∈ E_f then
        return f;
    end
    /* Else, we do DFS on s in residual network to find if there is a
       path to t (augmenting path). We return a linked list of the
       augmenting path if it exists, and we return NULL otherwise.        */
    augPath ⟵ DFS(V, E_f, s, t);
    if augPath = NULL then
        return f;
    end
    child ⟵ augPath.root();
    /* We increase the flow by 1 for each edge in the linked list
       (augmenting path).                                                  */
    while augPath.next(child) ≠ NULL do
        u ⟵ child;
        v ⟵ augPath.next(child);
        f(u, v) ⟵ f(u, v) + 1;
    end
    return f;
end
```

4. *Let $G = (V, E)$ be a flow network with source s, sink t, and integer capacities. Suppose we are given a maximum flow f in G, and suppose the capacity of a single edge $(u, v) \in E$ is decreased by 1. Give an $O(V + E)$-time algorithm to update the maximum flow.*

Algorithm 2: Update-Flow2(G)

Data: The set of vertices V and edges E of the given flow network G. A maximum flow f in G.

Result: The updated maximum flow f.

begin
- `/* We compute the residual network` E_f `of` G`. */`
- **for** $(u, v) \in E$ **do**
  - **if** $c(u, v) - f(u, v) > 0$ **then**
    - **add** (u, v) **to** E_f;
  - **end**
- **end**
- `/* We check if` $(u^*, v^*) \in E_f$`. */`
- **if** $(u^*, v^*) \in E_f$ **then**
  - **return** f;
- **end**
- `/* Else, we do DFS on` s `in` G `with the max flow` f `to find if there is a path to` u^*`. We return a linked list of the path if it exists, and we return NULL otherwise. We do the same for DFS from` v^* `to` t`. */`
- $pathU \longleftarrow DFS(G, f, s, u^*)$;
- $pathV \longleftarrow DFS(G, f, v^*, t)$;
- **if** $pathU = NULL$ ***or*** $pathV = NULL$ **then**
  - **return** f;
- **end**
- $child \longleftarrow pathU.root()$;
- `/* Else, we decrease the flow by 1 for each edge in the linked lists. */`
- **while** $pathU.next(child) \neq v^*$ **do**
  - $u \longleftarrow child$;
  - $v \longleftarrow pathU.next(child)$;
  - $f(u, v) \longleftarrow f(u, v) - 1$;
- **end**
- **while** $pathV.next(child) \neq NULL$ **do**
  - $u \longleftarrow child$;
  - $v \longleftarrow pathV.next(child)$;
  - $f(u, v) \longleftarrow f(u, v) - 1$;
- **end**
- **return** f;

end