AADS - Assignment 1

Aditya Fadhillah (hjg708), Mads Nørregaard (gnz359), Nikolaj Schaltz (bxz911)

January 8, 2025

Contents

1 Exercises from Section 26.1

1.1 CLRS 26.1-1

Show that splitting an edge in a flow network yields an equivalent network. More formally, suppose that flow network G contains edge (u, v), and we create a new flow network G' by creating a new vertex x and replacing (u, v) with new edges (u, x) and (x, v), such that $c(u, x) = c(x, v) = c(u, v)$. Show that a maximum flow in G' has the same value as a maximum flow in G.

Answer:

We will prove this by contradiction, while showing that flows in G and G' can be mapped to each other while preserving flow validity and total flow value.

Step 1: Define Maximum Flows f and f'

Let f be a maximum flow in G, with flow value $|f|$, and let f' be a maximum flow in G', with flow value $|f'|$.

We aim to show that $|f| = |f'|$. Suppose, for the sake of contradiction, that $|f| \neq |f'|$.

Step 2: Map a Flow f in G to f' in G'

First, suppose $|f'| > |f|$. Construct a flow f' in G' corresponding to f in G:

- Since the capacity of both of the two edges that are introduced by the splitting of that edge have the same capacity, we define:
$$f'(u, x) = f'(x, v) = f(u, v).$$
We must have that $f'(u, x) = f'(x, v)$ because the only edge into x is (u, x) and the only edge out of x is (x, v), and the net flow into and out of each vertex must be zero.
- For all other edges, we define:
$$f'(a, b) = f(a, b), \quad \forall (a, b) \neq (u, v).$$

Verify f' is a Valid Flow in G'

Capacity Constraints:

$$f'(u, x) = f(u, v) \leq c(u, v) = c(u, x), \quad f'(x, v) = f(u, v) \leq c(u, v) = c(x, v).$$

For all other edges, $f'(a, b) = f(a, b)$, so the capacity constraints are satisfied.

Flow Conservation:

For the new vertex x:

$$\sum_{v \in V} f'(v, x) = f'(u, x) = f(u, v), \quad \sum_{v \in V} f'(x, v) = f'(x, v) = f(u, v).$$

Thus, the total flow entering x equals the total flow leaving x. For all other vertices, flow conservation is preserved because $f'(a, b) = f(a, b)$ for $(a, b) \neq (u, v)$.

The total flow value in G' is:
$$|f'| = \sum_{v \in V} f'(s, v) - \sum_{v \in V} f'(v, s) = |f|.$$

Hence, f' in G' corresponds to f in G, and $|f'| \leq |f|$. This contradicts the assumption that $|f'| > |f|$.

Step 3: Map a Flow f' in G' to f in G

Now, suppose $|f| > |f'|$. Construct a flow f in G corresponding to f' in G':

- For the split edge (u, x), (x, v) in G', define:

$$f(u, v) = \min(f'(u, x), f'(x, v)).$$

- For all other edges, define:

$$f(a, b) = f'(a, b), \quad \forall (a, b) \neq (u, v).$$

Verify f is a Valid Flow in G

1. Capacity Constraints: By construction:

$$f(u, v) = \min(f'(u, x), f'(x, v)) \leq c(u, x) = c(u, v), \quad \text{and } f(a, b) = f'(a, b) \leq c(a, b).$$

2. Flow Conservation: For all vertices $v \neq x$, flow conservation in G' implies flow conservation in G. For vertex x, since:

$$f'(u, x) = f'(x, v),$$

the contribution from x collapses into $f(u, v)$, preserving flow conservation at u and v.

The total flow value in G is:

$$|f| = \sum_{v \in V} f(s, v) - \sum_{v \in V} f(v, s) = |f'|.$$

Hence, f in G corresponds to f' in G', and $|f| \leq |f'|$. This contradicts the assumption that $|f| > |f'|$.

Step 4: Conclude

Since we have shown that $|f| \leq |f'|$ and $|f'| \leq |f|$, we conclude that:

$$|f| = |f'|.$$

We have shown that any flow in G can be mapped to an equivalent flow in G' with the same total flow value, and vice versa. Furthermore, we demonstrated that if $|f| \neq |f'|$, it would lead to a contradiction, as the transformation preserves both flow conservation and capacity constraints.

Thus, the maximum flow value in G' is equal to the maximum flow value in G. Splitting an edge in a flow network does not alter the maximum flow, as the transformation is equivalent with respect to the flow properties of the network.

1.2 CLRS 26.1-4

Let f be a flow in a network, and let α be a real number. The *scalar flow product*, denoted αf, is a function from $V \times V$ to $\mathbb{R}$ defined by

$$(\alpha f)(u, v) = \alpha \cdot f(u, v).$$

Answer:
A flow f satisfies the capacity constraint:

$$f(u, v) \leq c(u, v), \quad \forall (u, v) \in E.$$

For αf, we have:

$$(\alpha f)(u, v) = \alpha \cdot f(u, v).$$

- If $\alpha > 0$, then $0 \leq (\alpha f)(u, v) \leq \alpha \cdot c(u, v)$, so the capacity constraint holds.
- If $\alpha = 0$, $(\alpha f)(u, v) = 0$ trivially satisfies the capacity constraint.
- If $\alpha < 0$, $(\alpha f)(u, v) < 0$, violating the non-negativity requirement for flows.

Thus, αf satisfies the capacity constraint only if $\alpha \geq 0$.

A flow f satisfies flow conservation:

$$\sum_{w \in V} f(w,u) = \sum_{w \in V} f(u,w), \quad \forall u \in V \setminus \{s,t\}.$$

For αf:

$$\sum_{w \in V} (\alpha f)(w,u) = \alpha \cdot \sum_{w \in V} f(w,u),$$

and

$$\sum_{w \in V} (\alpha f)(u,w) = \alpha \cdot \sum_{w \in V} f(u,w).$$

Since $\sum_{w \in V} f(w,u) = \sum_{w \in V} f(u,w)$, flow conservation is preserved for αf regardless of α.

The value of flow f is:

$$|f| = \sum_{(s,v) \in E} f(s,v).$$

For αf, the flow value is:

$$|\alpha f| = \sum_{(s,v) \in E} (\alpha f)(s,v) = \alpha \cdot \sum_{(s,v) \in E} f(s,v).$$

Thus:

$$|\alpha f| = \alpha \cdot |f|.$$

- If $\alpha > 0$: The scalar flow product αf satisfies all properties of a flow, and the flow value scales as $|\alpha f| = \alpha \cdot |f|$.
- If $\alpha = 0$: $(\alpha f)(u,v) = 0$ results in a trivial flow with $|\alpha f| = 0$.
- If $\alpha < 0$: The scalar flow product violates the non-negativity constraint and is not a valid flow.

1.3 CLRS 26.1-7

Suppose that, in addition to edge capacities, a flow network has *vertex capacities*. That is, each vertex v has a limit $l(v)$ on how much flow can pass through v. Show how to transform a flow network $G = (V, E)$ with vertex capacities into an equivalent flow network $G' = (V', E')$ without vertex capacities, such that a maximum flow in G' has the same value as a maximum flow in G. How many vertices and edges does G' have?

Answer:
How is the transformation done:
The transformation is done by creating a duplicate of each vertex, the duplicates will have an added prime $'$ to its name. An edge will be made between an existing vertex x and its duplicate x' where the edge will have the capacity that the vertex x had in G. All the edges going into the original x will continue to go into x while all the outgoing will now be going out of x'. Therefore the edge (x, y) would now be (x', y).

Proof that the transformed network has the same maximum flow value:

We will prove this statement by contradiction. Assume f is the maximum flow in G, and let f' be the maximum flow in G'. We aim to show that the maximum flow in G' f' cannot exceed the maximum flow in G f.

Define f in G and construct f' in G' as:

Let f be a maximum flow in G, and construct f' in G'. Let $l(x)$ be the capacity of vertex x in G and let $c(x', y)$ be the capacity of the edge from vertex x' to y in G'. Let $c(x, x')$ be the capacity of the edge from vertex x to vertex x'. By our transformation we have that:

$$l(x) = c(x, x'), \quad f(x, y) = f'(x', y), \quad l(x) = c(x, x')$$

In G', f' satisfies: The total flow that can enter x' equals the capacity of $l(x)$ from G thus no new bottlenecks has been made resulting in flow conservation and capacity constraints continuing to hold.

Contradiction: Assume that G' has a larger max flow than G

We have that f_{max} in G has the same max flow value as f' in G' suppose now that f' is not the max flow in G' it would give a new maxflow f'_{max} such that:

$$f'_{max} > f'$$

We now create a new transformation from G' to G where we create a new flow value f which has the same value as f'_{max}. We now have a f_{max} and a f in G where $f_{max} < f$ because of our assumption that G' has a larger max flow value than G. This is impossible as a max flow was already found in G and can therefore not be increased, confirming the answer that the max flow value in G and G' are equal.

How many vertices and edges does G' have?
G has n vertices and m edges.
G' has $2n$ vertices and $n + m$ edges.

The reason for these numbers are that, as explained in how the transformation is done, all vertices are duplicated making the amount of vertices 2n. G' will have $n + m$ edges as theres a new edge between each old vertice and its old, adding n edges, as the existing edges in G are not changed it will be $n + m$

2 Exercises from Section 26.2

2.1 CLRS 26.2-2

In Figure 26.1(b), what is the flow across the cut $(\{s, v_2, v_4\}, \{v_1, v_3, t\})$? What is the capacity of this cut?

Answer:
To find the flow, we look at all edges that s, v_2, v_4 lead to that do not lead to each other and look at the flows, then add the flows together:

$f(s, v_1) = 11$
$f(v_2, v_1) = 1$
$f(v_4, v_3) = 7$
$f(v_4, t) = 4$
$f(v_3, v_2) = 4$

We then add all flows together to obtain the flow across the cut. Because the vertex v_3 flows back to v_2, we need to subtract its flow from the equation.

$$\begin{aligned} |f| &= f(s, v_1) + f(v_2, v_1) + f(v_4, v_3) + f(v_4, t) - f(v_3, v_2) \\ &= 11 + 1 + 7 + 4 - 4 \\ &= 19 \end{aligned}$$

Thus, the flow across the cut is 19.

To calculate the capacity, we look at the capacities of the same edges and add them together (we do not look at the vertices that flow back, so no subtraction is done):

$c(s, v_1) = 16$
$c(v_2, v_1) = 4$
$c(v_4, v_3) = 7$
$c(v_4, t) = 4$

$$|c| = c(s, v_1) + c(v_2, v_1) + c(v_4, v_3) + c(v_4, t)$$

$$= 16 + 4 + 7 + 4$$
$$= 31$$

Thus, the capacity across the cut is 31.

2.2 CLRS 26.2-3

Show the execution of the Edmonds-Karp algorithm on the flow network of Figure 26.1(a).

Answer:
First, since we are not given any flow values and only capacities, we initialize all flow values to 0 before running the algorithm.

We then use breadth-first-search to look at the vertices in order from source s to sink t. The algorithm will use BFS to look at the first augmenting path $\{s \to v_1 \to v_3 \to t\}$. For this path, we have the capacities:
$c(s, v_1) = 16$
$c(v_1, v_3) = 12$
$c(v_3, t) = 20$
$min(16, 12, 20) = 12$

The bottleneck/min. capacity of the first path is 12, so we send 12 flow over the edges and subtract 12 from the capacities for each edge. Thus, we now have:
$c(s, v_1) = (16 - 12) = 4$
$f(v_1, v_3) = (12 - 12) = 0$
$f(v_3, t) = (20 - 12) = 8$

The algorithm then finds the second augmenting path, $\{s \to v_2 \to v_4 \to t\}$. For this path we have the capacities:
$c(s, v_2) = 13$
$c(v_2, v_4) = 14$
$c(v_4, t) = 4$
$min(13, 14, 4) = 4$

The bottleneck/min. capacity of the second path is 4, so we send 4 flow over the edges and update the capacities again. Now we have 16 flow sent so far and the edges:
$f(s, v_2) = (13 - 4) = 9$
$f(v_2, v_4) = (14 - 4) = 10$
$f(v_4, t) = (4 - 4) = 0$

We then look at the third augmenting path, $\{s \to v_2 \to v_4 \to v_3 \to t\}$. For this path we have the capacities:
$c(s, v_2) = 9$
$c(v_2, v_4) = 10$
$c(v_4, v_3) = 7$
$c(v_3, t) = 8$
$min(9, 10, 7, 8) = 7$

The bottleneck/min. capacity of the third path is 7, so we send 7 flow over the edges and update the flow values again. Thus, 23 flow has been sent so far. We now have:
$c(s, v_2) = (9 - 7) = 2$
$c(v_2, v_4) = (10 - 7) = 3$
$c(v_4, v_3) = (7 - 7) = 0$
$c(v_3, t) = (8 - 7) = 1$

With that, there are no more augmenting paths and no more flow can be sent from source s to sink t without overloading, so the algorithm terminates and we have found a max flow of 23.

2.3 CLRS 26.2-4

In the example of Figure 26.6, what is the minimum cut corresponding to the maximum flow shown? Of the augmenting paths appearing in the example, which one cancels flow?

Answer:
Looking at the figure, we see that flow is only canceled between the edge that connects v_3 and v_2. Therefore, our cut should not include both v_2 and v_3, in order to keep this edge active and act as an augmenting path.

With this knowledge, the optimal minimum cut must be $S = \{s, v_1, v_2, v_4\}$ and $T = \{v_3, t\}$, as it satisfies the condition while also keeping a max flow, since there are no other edges that cancel flow. There are 3 different augmenting paths provided in Figure 26.6, from observing the flow values across the edges, it can be seen that path (c), $\{s \to v_2 \to v_1 \to v_3 \to t\}$ cancels out the flow between v_3 and v_2, since the edge has 4 flow after path (a) and (b) and 0 after path (c). Thus, augmenting path (c) cancels flow.

2.4 CLRS 26.2-7

Lemma 26.2
Let $G = (V, E)$ be a flow network, let f be a flow in G, and let p be an augmenting path in G_f. Define a function $f_p : V \times V \to \mathbb{R}$ by

$$f_p(u, v) = \begin{cases} c_f(p) & \text{if } (u, v) \text{ is on } p, \\ 0 & \text{otherwise.} \end{cases} \tag{26.8}$$

Then, f_p is a flow in G_f with value $|f_p| = c_f(p) > 0$.

Prove Lemma 26.2.

Answer:
If f_p is a flow, it must satisfy both the capacity constraint and the flow constraint. For the capacity constraint, $c_f(p)$ is defined as the smallest residual capacity of any edges along augmenting path p. Given the definition of a residual capacity, $c_f(p) \leq c_f(u, v)$, we have that:

$$f_p(u, v) \leq c_f(u, v)$$

All edges not on p can be ignored, as they are equal to 0, maintaining capacity constraint.

Thus f_p satisfies the capacity constraint, given the properties and definition of a residual capacity.

For the flow constraint, at the source s of augmenting path p, f_p sends $c_f(p)$ flow to the next vertex, and at the sink t, flow corresponding to $c_f(p)$ amount are received. In between vertices before s and t, f_p keeps balance since the same $c_f(p)$ flow is passed between vertices in p. As with before, all vertices not on p can be ignored as they are all equal to 0, maintaining flow constraint.

Thus f_p satisfies the flow constraint as well, given the amount of flow in $c_f(p)$.

Finally, to see if $|f_p| = c_f(p) > 0$, which is the total sum of the forward flow in f_p, $f_p(u, v)$, minus the backward flow in f_p, $f_p(v, u)$, written as:

$$|f_p| = \sum_u f_p(s, u) - \sum_v f_p(s, u).$$

Because $f_p(u, v) = c_f(p)$ for a forward edge to the sink t as t will always be on the augmenting path p, since it connects the source s and sink t through any vertices with positive residual capacities, and $f_p(v, u) = 0$ for all backward edges from t. We look at backward edges from the sink rather than forward edges through the source since we know that flow conservation is true, so we can do it either way, but using backward edges instead is easier, since due to the properties of an augmenting path, flow does not leak from s to t through p.

Through this, we then have that $|f_p| = c_f(p)$, and by definition of p, $c_f(p) > 0$ in p, which thus means that the condition holds for $|f_p|$.

2.5 CLRS 26.2-9

Suppose that both f and f' are flows in a network G and we compute flow $f \uparrow f'$. Does the augmented flow satisfy the flow conservation property? Does it satisfy the capacity constraint?

Answer:
We look at both properties individually, beginning with flow conversation, which simply states that the total incoming flow must be equal to the total outgoing flow:

$$\sum_u f(v,u) = \sum_u f(u,v)$$

We can assume that if both f and f' are valid flows in G, they independently satisfy the flow conversation. The augmented flow $f \uparrow f'$ is the same as the sum of the two flows f and f':

$$(f \uparrow f')(u,v) = f(u,v) + f'(u,v)$$

This satisfies flow conversation, as the sums between incoming and outgoing flows in both f and f' remain balanced. Thus, $f \uparrow f'$ satisfies flow conversation.

For capacity constraint, it states that flow on a given edge can never exceed the capacity of said edge:

$$0 \leq f(u,v) \leq c(u,v) \text{ and } 0 \leq f'(u,v) \leq c(u,v)$$

While this constraint can be satisfied independently, it cannot always be satisfied for the augmented flow $f \uparrow f'$ if the sum of the flow from f and f' is greater than the capacity:

$$(f \uparrow f')(u,v) = (f(u,v) + f'(u,v)) \geq c(u,v)$$

This can be seen via. an example; let's say if an edge has a capacity $c(u,v) = 8$, and two different flows, $f(u,v) = 4$ and $f'(u,v) = 5$ are passed through, we can see that they individually satisfy the capacity constraint, but $f + f'$ does not, as $9 > 8$, which violates the constraint.

Therefore, the capacity constraint is not guaranteed to be satisfied for all flow and capacities.

3 Exercises from Section 26.3

3.1 CLRS 26.3-2

Theorem 26.10
If the capacity function c takes on only integral values, then the maximum flow f produced by the Ford-Fulkerson method has the property that $|f|$ is an integer. Moreover, for all vertices u and v the value of $f(u,v)$ is an integer.

Prove Theorem 26.10.

Answer:
We prove the theorem by induction on the amount of iterations done in Ford-Fulkerson.

First, our base case, i.e. at the initialization of the algorithm ($n = 0$ iterations), the flow f is initialized to 0 for all edges. If c can only take on integral values, then it must be true that both $f(u,v)$ and $|f|$ are integers, since they are 0 during initialization.

For the induction step, we look at the $(n+1)$th iteration of the algorithm. After $n+1$ iterations, a residual graph is constructed, where each edge has a capacity of:

$$c_f(u,v) = c(u,v) - f(u,v)$$

which is an integer, since both c and f only take integer values, and this also holds true for flow of backward edges, $f(v, u)$. If an augmenting path, $c_f(p)$ exists, then it has a residual capacity of:

$$min\{c_f(u, v) : (u, v) \text{ is in } p\}$$

and since $c_f(u, v)$ are integers for all edges, then $c_f(p)$ is also an integer.

For each iteration, the flow is then updated based on the property of the edges; the residual capacity $c_f(p)$ is added to forward edges and $c_f(p)$ is subtracted to backwards edges, as seen in the last lines (6-8) of the algorithm. This can also be illustrated as:

$$f(u, v) = \begin{cases} f(u, v) + c_p & \text{if } (u, v) \text{ is a forward edge on } p \\ f(u, v) - c_p & \text{if } (u, v) \text{ is a backward edge on } p \end{cases}$$

Since we know that both the flow $f(u, v)$ and the residual capacity $c_f(p)$ are both integers, then the updated flow values stay as integers. Since the total flow $|f|$ is a sum of flow values:

$$|f| = \sum_v f(u, v)$$

which have all been proven to be individual integers, then $|f|$ is also an integer.

Thus it is proven that for any given flow, $f(u, v)$ is an integer, and the total flow $|f|$ is also an integer, for any given iteration of the Ford-Fulkerson algorithm.