CS5800: Algorithms — Spring '21 — Virgil Pavlu

Homework 12
Submit via Gradescope

Name: Ketaki Nitin Kolhatkar
Collaborators: Nitheesh Koushik Gattu

Instructions:

- Make sure to put your name on the first page. If you are using the LaTeX template we provided, then you can make sure it appears by filling in the `yourname` command.
- Please review the grading policy outlined in the course information page.
- You must also write down with whom you worked on the assignment. If this changes from problem to problem, then you should write down this information separately with each problem.
- Problem numbers (like Exercise 3.1-1) are corresponding to CLRS 3^{rd} edition. While the 2^{nd} edition has similar problems with similar numbers, the actual exercises and their solutions are different, so make sure you are using the 3^{rd} edition.

1. ***(20 points)*** *Exercise 26.1-3.*

Solution:

When we say that there is no path from s to t through the vertex u, it tells us that either there is no path from s to vertex u or no path from vertex u to t. In the first scenario, let us assume that there is vertex v such that $f(v,u) > 0$. Now v does not have a path from s to v meaning there is a path existing from s to u which contradicts the assumption. Next we create another vertex before v named v′ which has $f(v',v) > 0$. This vertex also should not have any path from vertex s to v′ meaning there would be a path from s to u. This procedure will continue infinitely and new vertices would be added would be added as the predecessor to satisfy the condition saying that there should be no path from s to u. Similarly, this procedure would be followed for case when there is no path from u to t. Here only difference would be the addition of successors instead of predecessors. There needs to be flow through u for the law of conservation to be satisfied, f(u,v) = f(v,u) = 0.

2. ***(20 points)*** *Exercise 26.1-4.*

Solution:

For a flow in a network to be a convex set, it should follow the capacity constraint and the the flow conservation property.

1. f1 and f2 should follow the capacity constraint, hence we get: $0 \leq \alpha f_1(u,v) + (1-\alpha) f_2(u,v) \leq \alpha c(u,v) + (1-\alpha)c(u,v) = c(u,v)$.

2. f1 and f2 also satisfy the flow conservation as:

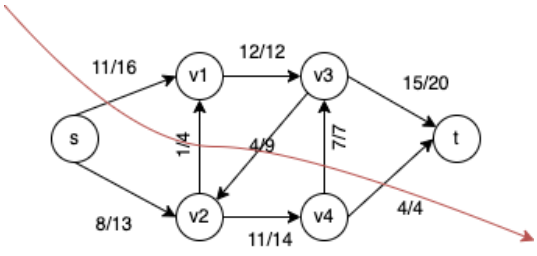
$$0 \leq \alpha f(u,v) + (1-\alpha) f'(u,v) \leq \alpha c(u,v) + (1-\alpha)c(u,v) = c(u,v)$$

$$\sum \alpha f(u,v) + \sum (1-\alpha) f'(u,v) = \alpha \sum f(u,v) + (1-\alpha) \sum f'(u,v)$$

$$\alpha \sum f(u,v) + (1-\alpha) \sum f'(u,v) = \alpha \sum f(u,v) + (1-\alpha) \sum f'(u,v)$$

3. ***(20 points)*** *Exercise 26.2-2.*

Solution:



The cut (s, v2, v4, v1, v2, t) has a flow of 11 + 1 + 4 + 7 - 4 = 19. The cut has a total capacity of 16 + 4 + 7 + 4 = 31.

4. ***(Extra Credit)*** *Exercise 26.2-10.*

Solution:

5. ***(30 points)*** *Implement Push-Relabel for finding maximum flow.*
Extra Credit: use relabel-to-front idea from Chapter 26.5 with the Discharge procedure.

Solution:

```
Initialise height of all the vertices as 0
Initialise Height of source = V

Initialise excess flow of all the vertices as 0
For all neighbouring vertices of source node:
    Flow and excess flow = capacity of the connecting source-neighbour edge

While there is a vertex with excess flow:
    Perform push or relabel

At the end all the vertices should have 0 excess flow
Return maximum flow
```

6. ***(15 points)*** *Explain in a brief paragraph the following sentence from textbook page 737: "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".*

Solution:
In the algorithm, push function sends the flow from a vertex that has excess flow with it. The neighbouring vertices with smaller heights receive flow due to this push function. The flow is these vertices is stored in reservoirs. This means that if there is a condition when a vertex has excess flow and there are no neighbouring vertices where it can send the flow which is a at a height lower than the vertex, we need to move the vertex's height to one higher than the lowest neighbour so we can use the push function again. The source vertex could also be the neighbouring vertices.

7. ***(Extra Credit)*** *Exercise 26.4.4.*

Solution: