MAC0328 GRAPH ALGORITHMS: PROGRAMMING ASSIGNMENT 4

MAX-FLOW MIN-CUT

MARCEL K. DE CARLI SILVA AND THIAGO LIMA OLIVEIRA

DUE DATE: 16/DEC/2021 AT 11:59PM

In this programming assignment, your task is to write an implementation of the Edmonds–Karp Algorithm for the Maximum Integral Flow Problem.

This assignment will be graded out of 100 marks.

Your code **must** be written in `C++14` and use the BGL library. As before, you must use the BGL **only** for the data structures (and corresponding accessor functions) that store a digraph and the attributes for its vertices and arcs. The use of any **BGL algorithm** is forbidden.

1. TEST CASES AND GRADING

Your program should solve each test case in $O((n+m)nm)$ time, where n is the number of vertices and m is the number of arcs of the input digraph $D = (V, A)$.

Each test case has the following format:

- The first line has two integers, n and m, the numbers of vertices and arcs, respectively, such that $1 \leq n \leq 5 \times 10^3$ and $1 \leq m \leq 5 \times 10^3$.
- The second line has two integers, s and t in the range $[1, n]$, the source/start and the target/sink of the flow.
- The next m lines have the description of the arcs. Each arc is described by three integers, u, v, and $c(uv)$, that is, the tail and head of the arc (in the range $[1, n]$, as usual), and the capacity of the arc, respectively. The capacity satisfies $1 \leq c(uv) \leq 10^3$.

The ordering of the arcs in the input will be important for the output, so it may be convenient to keep track of it. You may use the fact that a call to `boost::add_edge` returns a `std::pair<Arc,bool>`, where `Arc` is the type for an edge descriptor of the chosen graph type, containing an edge/arc descriptor and a boolean that indicates whether the edge/arc was successfully added. Hence, one may want to record the value of `a` after running the following snippet:

```
Arc a; std::tie(a, std::ignore) = boost::add_edge(u, v, digraph);
```

It is *mandatory* for the augmented digraph $\widehat{D} = (V, A \times \{\pm 1\}, \varphi)$ to have, for each original arc that has a positive amount of flow, a *new* reverse arc, *even if the digraph already has an original arc in that direction.*

For each iteration of the Edmonds–Karp algorithm, your solution should output a description of the residual capacities of all the arcs in the augmented digraph $\widehat{D} = (V, A \times \{\pm 1\}, \varphi)$ and either the flow update data or a description of a minimum cut, in the following format:

INSTITUTO DE MATEMÁTICA E ESTATÍSTICA, UNIVERSIDADE DE SÃO PAULO, R. DO MATÃO 1010, 05508-090, SÃO PAULO, SP

E-mail address: {mksilva,thilio}@ime.usp.br.

Date: November 25, 2021, git commit `4f41fcb`.

- The first m lines contain a description of the residual capacities. For each arc $a \in A$, *in the same order as the input*, print a line containing two integers, $c_f((a,+1))$ and $c_f((a,-1))$, i.e., the residual capacities of the arcs $(a,+1)$ and $(a,-1)$ of $\widehat{D}$. Note that you should print both residual capacities, even if one of them is zero (so the corresponding arc lies in $\widehat{D}$ but not in the residual digraph D_f).
- If there is no augmenting path, the last line should start with three integers, 1, $\mathrm{val}(f)$, and $|S|$, where $S \subseteq V$ is such that $s \in S$, $t \notin S$, and $\mathrm{val}(f) = c(\delta^{\mathrm{out}}(S))$. After this, the (same) line should have $|S|$ integers, one for each vertex of S, in the range $[1,n]$.
- If there is a *shortest* augmenting path P, the last line should start with three integers, 0, ε, and ℓ, where $\varepsilon > 0$ is the minimum residual capacity of an arc traversed by P and ℓ is the length of P. After this, the (same) line should have ℓ integers, one for each arc traversed by P, in order. To represent an arc (a,α) of D_f traversed by P, with $a \in A$ and $\alpha \in \{\pm 1\}$, print the index of the arc a in the input order (in the range $[1,m]$) if $\alpha = +1$ or minus that index (in the range $[-m,-1]$) if $\alpha = -1$.

Your submission should be a single file, called `NUSP.cpp`, obviously with `NUSP` replaced by your university ID number.

2. Bonus

You may optionally submit a bonus assignment, explained below. In case you submit this bonus, it will be graded out of 150 marks, and the exceeding part will *not* be truncated when contributing to the final course grade. E.g., if the grades for assignments 1, 2, and 3 are 75, 75, and 100, and the bonus gets 150 marks, the final course grade will be 100. (Naturally, the final course grade will be rounded *down* to 100 in case it goes above 100.)

On the other hand, the grade for this bonus part will be *binary*: either the code is correct for *all* test cases and you get 150 marks for this assignment, or the bonus will be dismissed. In the latter case, we will grade the main submission, as described in Section 1, without any penalty, as if the bonus part had not been submitted at all.

This bonus part is a variation of the Ford–Fulkerson algorithm, which we describe next. In place of the flow update rule '$f \leftarrow f + \varepsilon f_P$' where P is an augmenting path for f (i.e., an st-path in the residual digraph D_f), we will use the update rule '$f \leftarrow f + \Delta g$', where $\Delta g \colon A \to \mathbb{R}$ is defined in the following way.

Let us consider the residual digraph D_f of D with respect to the current feasible (s,f)-flow f (and arc capacities c). Let $\mu(D_f)$ denote the set of arcs that are traversed by *some* shortest st-path in D_f, and let D'_f be the digraph obtained from D_f by deleting all the arcs but the ones in $\mu(D_f)$. An (s,t)-flow g in D'_f that is feasible with respect to c_f is called *maximal* if, whenever $\tilde{g}$ is an (s,t)-flow in D'_f such that $g(b) \leq \tilde{g}(b) \leq c_f(b)$ for each arc b of D'_f, one actually has $g = \tilde{g}$.

In the update rule '$f \leftarrow f + \Delta g$', the term g should be a maximal feasible[1] (s,t)-flow in D'_f, and $\Delta g \colon A \to \mathbb{R}$ is defined as

$$(\Delta g)(a) \coloneqq g((a,+1)) - g((a,-1)),$$

for each $a \in A$, where we consider $g(b)$ to be zero if the arc b does not exist in D'_f. In this case, it can be shown that at most n augmentations are performed until one reaches a feasible (s,t)-flow of maximum value.

Should you choose to implement the bonus part, you must implement this algorithm in such a way that each iteration (i.e., the computation of a maxim*al* feasible flow in D'_f) runs in $O(mn)$ time. One way to obtain such a maximal flow is by a modification of DFS.

The output for each iteration is described next. If the current flow f is maximum, print the integer 1 on a line by itself, followed by m lines describing the residual capacities of the augmented

[1] with respect to c_f

digraph $\widehat{D}$, as described in Section 1, i.e. with two residual capacities for each arc $a \in A$. Include a final/last line starting with two integers: $\mathrm{val}(f)$ and $|S|$, where $S \subseteq V$ is such that $s \in S$, $t \notin S$, and $\mathrm{val}(f) = c(\delta^{\mathrm{out}}(S))$. After this, the (same) line should have $|S|$ integers, one for each vertex of S, in the range $[1, n]$.

If the current flow f is *not* maximum, print the integer 0 on a line by itself, followed by a description of D'_f and g as follows. For each arc $a \in A$, *in the same order as the input*, print a line containing the four integers $c_f\big((a,+1)\big)$, $g\big((a,+1)\big)$, $c_f\big((a,-1)\big)$, and $g\big((a,-1)\big)$. As before, if any of the arcs does not exist in D'_f, fill in the corresponding g value to be zero.