

# Introduction to Pseudocode and Math

Anna Ritz  
Biology Department, Reed College  
Portland, Oregon  
aritz@reed.edu

## ABSTRACT

This ACM-style template describes how to typeset pseudocode as well as write common mathematical symbols. **Copy this project and start by modifying the title, author, etc.** There are also very useful URLs on Moodle for more information.

## Keywords

pseudocode, algorithms, math, LaTeX

## 1. PSEUDOCODE

The goal is to find the flow-betweenness centrality of all of the nodes in the graph. To do this I use algorithm 1 to get the normalized amount of flow calculated through each pair of points. This algorithm uses the Ford-Fulkerson method to calculate individual flows through nodes. I separate the flows into four different sections which use different subsets depending on which social group the node in question and to the source and target node are from. First,  $c_{total}$  includes all of the possible pairing of source and sink nodes. Second  $c_{btwn}$  consists of source and target are not from the same social group. The next two are if the source and the target are in the same social group:  $c_{inter}$  has the node in question in that same social group,  $c_{out}$  has the node in question in a different social group. Algorithm 1 iterates between all unique combinations of sources and targets with  $(u, v) = (v, u)$  and at each combination finds the normalized flow for all the nodes using 2. It then adds those flows to the centrality group that they are a part of.

The Ford-Fulkerson method (Algorithm 2) works by simulating putting flow through the graph. It does this by keeping track of a residual network ( $G_f$ ) which is a representation of how much more flow can go through each of the edges. The algorithm goes until there is not path from the source to the sink where all of the edges  $> 0$  in the residual graph. In other words this is until there can be no more flow to the target from the source. Each iteration the algorithm find a path using depth first search on  $G_f$ , then it

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

Bio331 Fall 2016, Reed College, Portland, OR

© 2016 Copyright held by the owner/author(s).

ACM ISBN X-XXXX-XX-XXXX-XX.

DOI: XX.XXX/XXX\_X

---

## Algorithm 1 Flow-Betweenness Centrality

---

**Inputs:**  $G(E, V)$  and edge weights  $w$  social groups  $s$

**Outputs:** flow-betweenness centrality for nodes

$c_{inter}(n) \leftarrow 0$  for  $n \in E$

$c_{btwn}(n) \leftarrow 0$  for  $n \in E$

$c_{out}(n) \leftarrow 0$  for  $n \in E$

**for**  $k \in V$  **do**

**for**  $j \in V : (j, k) \in E$  and  $j < k$  **do**

$f(u, v) = \text{Ford-Fulkerson}(G, w, j, k)$

**for**  $n \in V : n \neq j$  and  $n \neq k$  **do**

$c(n) = \sum_{o \in N(n)} \frac{|f(n, o)|}{2(\sum_{o \in N(k)} |f(k, o)|)}$  where  $N(n)$  are  $n$ 's neighbors

**if**  $s(j) = s(k)$  **and**  $s(j) = s(n)$  **then**

$c_{inter} = c_{inter} + c$

**else if**  $s(j) = s(k)$  **then**

$c_{out} = c_{out} + c$

**else**

$c_{btwn} = c_{btwn} + c$

**end if**

$c_{total} = c_{total} + c$

**end for**

**end for**

**end for**

**return**  $c_{total}, c_{inter}, c_{btwn}, c_{out}$

---

adds the minimum  $edge_{cf}$  value to each of the flows and the recalculates  $G_f$ . This is like sending the most possible flow through that path which is restricted by edge with the lowest flow. Once there are no more paths in  $G_f$  from the source to the target it will return the flows for each of the edges in  $G$ .

---

## Algorithm 2 Ford-Fulkerson method

---

**Inputs:** A network  $G$  =

$(V, E)$  with flow capacity  $c$ , source  $s$ , and target  $t$

**Outputs:** Flows  $f(u, v)$  for all  $(u, v) \in E$  between  $s$  and  $t$  ∈

$f(u, v) \leftarrow 0$  for all edges  $(u, v)$

**while** there exists a path  $p_{st}$  in  $G_f$  :  $c_f(u, v) >$

0 for all edges  $(u, v) \in p$  **do**

  find  $c_f(p) = \min(c_f : (u, v) \in p)$

**for** each edge  $(u, v) \in p$  **do**

$f(u, v) \leftarrow f(u, v) + c_f(p)$

$f(v, u) \leftarrow f(v, u) - c_f(p)$

**end for**

**end while**

**return**  $f(u, v)$

---