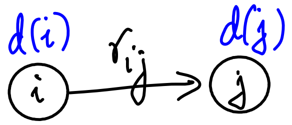
MATH 566: Lecture 29 (12/03/2024)

Today: * On SAP algorithm
* generating random networks for max flow

On Shortest Augmenting Path algorithm

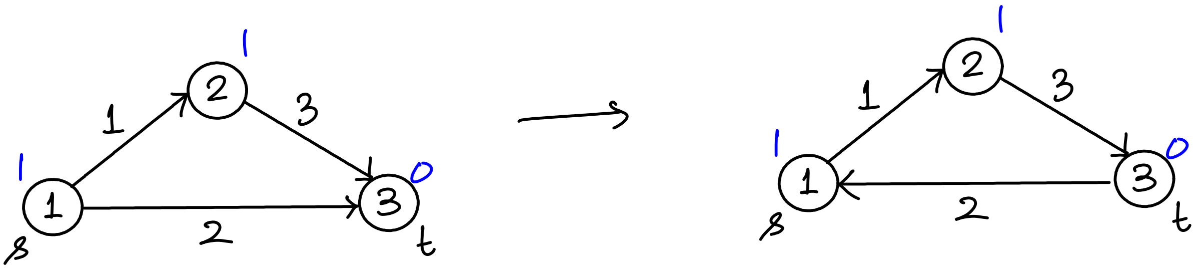
Recall that the main "while" loop runs as long as $d(s) < n$ holds. We could encounter a situation where we can't advance from s, i.e., there are no admissible arcs out of s. But there are also no arcs available to relabel. Yet, since $d(s) < n$ still holds, the algorithm is "stuck". Here is an example.

relabel : set $d(i) = \min_{j:(i,j)\in G(\bar{x})} \{d(j)\} + 1.$

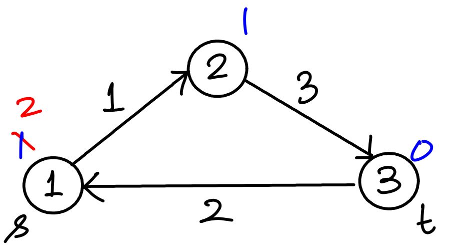


1. Augment along (1,3):

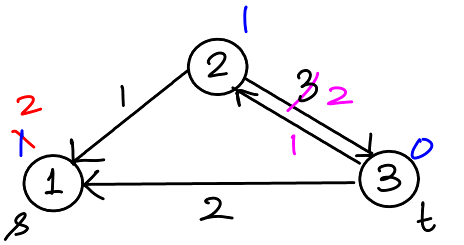
$n = 3$



2. Relabel node $s=1$: $d(1) \leftarrow d(2)+1$ $d(s) = 2 < n = 3$ still.



3. Augment along 1-2-3 ⟶



We can stop here, but $d(s) = 2 < n = 3$, and we cannot relabel $d(s)$ to increase it beyond n.

If you face this situation, just set $d(s) = 2n$ (or $n+1$), for instance.

Test your programs on several small networks, and for different choices of s and t (e.g., $s = t$ should give $v = 0$).

Generating Random Networks for Max Flow

Could justify choosing $s = 1, t = n$ as default.

* For generating "nontrivial" instances, ensure there is an s-t path in the instances.

* Could, alternatively, run the algorithms on two subsets of instances — one with s-t paths, and second without.

So, use an existing random network generator, or generate networks randomly on your own. For each network, test if an s-t directed path exists (using BFS). If yes, put it in Group 1; else in Group 2.

Report statistics (running times) separately for the two groups of networks.

Preflow push **may** take longer time on "infeasible" instances, i.e., from Group 2.

A good approach: "layered pipe" network

- Could have, say, $\frac{n}{3}$ layers of nodes.
- t is at bottom layer (ground, or layer 0).
- s at top layer
- Remaining nodes thrown into the intermediate layers.
- A majority of arcs could go downhill 1 layer,
  - a smaller number go 2 layers downhill,
  - some go 3 layers downhill
- Add some arcs going 1 or 2 layers uphill.

- Check if an s-t path exists → BFS search
  - if yes, put into group 1
  - if no, put into group 2

- Once this structure is built, sample u_{ij}'s from $[1, U]$ for $U \in \mathbb{Z}_{>0}$, and vary U.

- Can describe procedure as pseudocode or in words.
- Suggest plotting graphs of average running times...

You can report the average run times as functions of n, m, U, varying one parameter at a time.

When comparing to the worst-case time bound, take the constant of 'O' as 1. For SAP max flow, for instance, plot n^2m. (recall, it's run time is $O(n^2m)$).