Algorithm Theory

Chapter 7
Randomized Algorithms

Part VI:
Implementing Edge Contractions

Fabian Kuhn

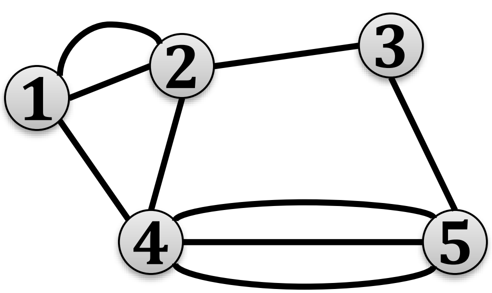
Implementing Edge Contractions

Edge Contraction:

- Given: multigraph with n nodes
  - assume that set of nodes is $\{1, \dots, n\}$
- Goal: contract edge $\{u, v\}$

Data Structure

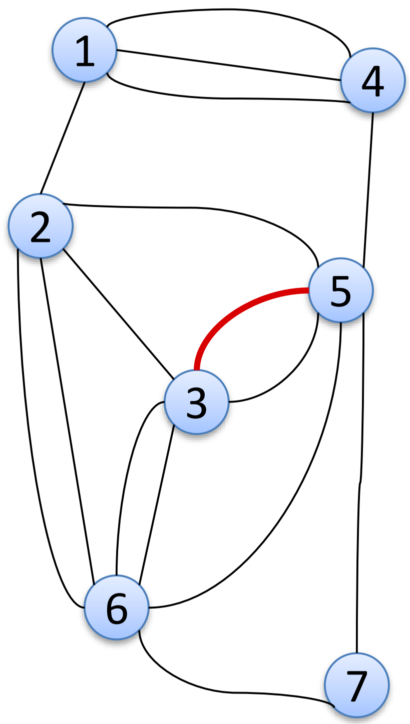
- We can use either adjacency lists or an adjacency matrix
- Entry in row i and column j: #edges between nodes i and j
- Example:



$$A = \begin{pmatrix} 0 & 2 & 0 & 1 & 0 \\ 2 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 3 \\ 0 & 0 & 1 & 3 & 0 \end{pmatrix}$$

Contracting An Edge

Example: Contract one of the edges between 3 and 5

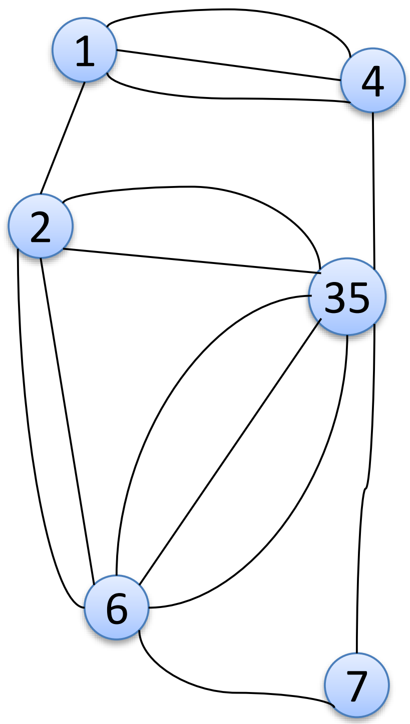


	1	2	3	4	5	6	7
1	0	1	0	3	0	0	0
2	1	0	1	0	1	2	0
3	0	1	0	0	2	2	0
4	3	0	0	0	1	0	0
5	0	1	2	1	0	1	1
6	0	2	2	0	1	0	1
7	0	0	0	0	1	1	0

{3,5}	0	2		1		3	1

Contracting An Edge

Example: Contract one of the edges between 3 and 5



	1	2	35	4		6	7
1	0	1	0	3		0	0
2	1	0	2	0		2	0
35	0	2	0	1		3	1
4	3	0	1	0		0	0
6	0	2	3	0		0	1
7	0	0	1	0		1	0

{3,5}	0	2		1		3	1

Contracting an Edge

Claim: Given the adjacency matrix of an n-node multigraph and an edge $\{u, v\}$, one can contract the edge $\{u, v\}$ in time $O(n)$.

- Row/column of combined node $\{u, v\}$ is sum of rows/columns of u and v
- Row/column of u can be replaced by new row/column of combined node $\{u, v\}$
- Swap row/column of v with last row/column in order to have the new $(n-1)$-node multigraph as a contiguous $(n-1) \times (n-1)$ submatrix

Finding a Random Edge

- We need to contract a uniformly random edge
- How to find a uniformly random edge in a multigraph?
  - Finding a random non-zero entry (with the right probability) in an adjacency matrix costs $O(n^2)$.

Idea for more efficient algorithm:

- First choose a random node u
  - with probability proportional to the degree (#edges) of u
- Pick a random edge of u
  - only need to look at one row → time $O(n)$

Choose a Random Array Entry

Problem: Given an array $A = [a_1, \dots, a_n]$ with $a_i \geq 0$, choose a random index i with probability proportional to a_i. (assume that $S \coloneqq \sum_{i=1}^{n} a_i$)

Choose a random array entry:

```
sum = 0;
for i = 1,...,n:
    with probability a_i/(S − sum):
        pick index i; terminate
    else
        sum += a_i
```

running time $O(n)$

Probability for Picking Index i:

$$\mathbb{P}(\text{index } i) = \left(1 - \frac{a_1}{S}\right) \cdot \left(1 - \frac{a_2}{S - a_1}\right) \cdot \dots \cdot \left(1 - \frac{a_{i-1}}{S - \sum_{j=1}^{i-2} a_j}\right) \cdot \frac{a_i}{S - \sum_{j=1}^{i-1} a_j}$$

$$= \frac{S - a_1}{S} \cdot \frac{S - a_1 - a_2}{S - a_1} \cdot \dots \cdot \frac{S - \sum_{j=1}^{i-1} a_j}{S - \sum_{j=1}^{i-2} a_j} \cdot \frac{a_i}{S - \sum_{j=1}^{i-1} a_j} = \boldsymbol{\frac{a_i}{S}}$$

Choose a Random Node

Edge Sampling:

1. Choose a node $u \in V$ with probability
$$\frac{\deg(u)}{\sum_{v \in V} \deg(v)} = \frac{\deg(u)}{2m}$$
   - Need to keep track of node degrees and number of edges m
   - Can at no extra cost (asymptotically) when doing edge contractions

2. Choose a uniformly random edge of u

Probability for getting edge e between u and v:

$$\mathbb{P}(\text{edge } e) = \frac{\deg(u)}{2m} \cdot \frac{1}{\deg(u)} + \frac{\deg(v)}{2m} \cdot \frac{1}{\deg(v)} = \frac{1}{m}$$

Randomized Min Cut Algorithm

Theorem: If the contraction algorithm is repeated $O(n^2 \log n)$ times, one of the $O(n^2 \log n)$ instances returns a min. cut w.h.p.

Corollary: The contraction algorithm allows to compute a minimum cut in $O(n^4 \log n)$ time w.h.p.

- One instance consists of $n - 2$ edge contractions
- Each edge contraction can be carried out in time $O(n)$
  - Actually: $O(\text{current \#nodes})$
- Time per instance of the contraction algorithm: $O(n^2)$