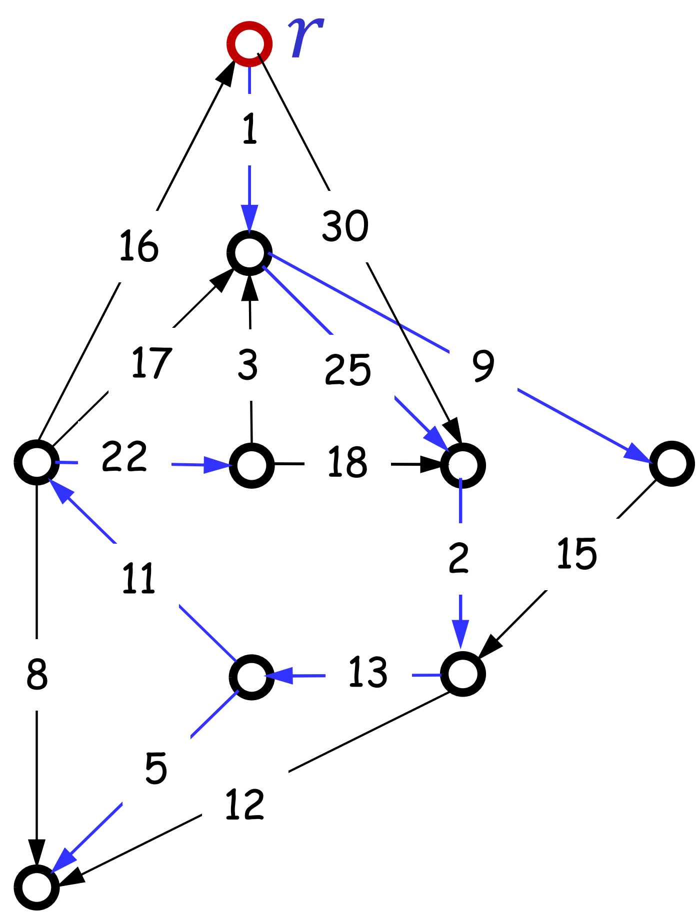
Lec 10: Minimum Spanning Arborescence

Minimum Spanning Arborescence

[Chu-Liu 1965] [Edmonds 1967]

- $G = (V, E; w)$: an edge-weighted directed graph
- r: a root vertex in V
- find a spanning r-arborescence T with minimum weight $w(T)$.



Optimality-Preserving local weight-resetting

Local subtraction at $v \neq r$: subtract the weights of all edges entering v by the same amount.

- Optimality-preserving: exactly one edge entering v in any SA

Basic weight-resetting at $v \neq r$: find the lightest edge entering v, and subtract its weight from all edges entering v.

- at least one 0-edge entering v
- all edges entering v have non-negative weights.

Linear-time preprocessing

In $O(m)$ time,

- remove all edges entering r
- perform basic weight-resetting at every $v \neq r$

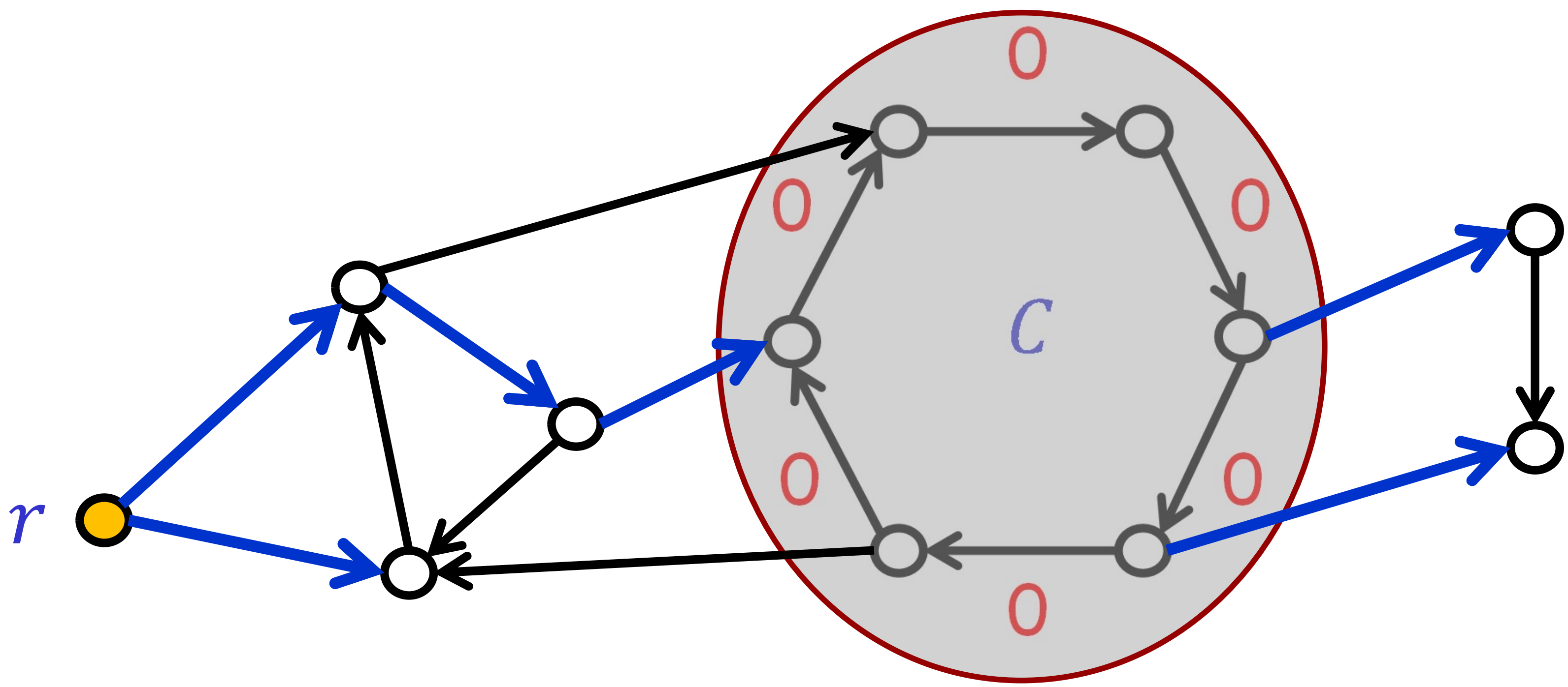
Now, all edges have non-negative weights, and each vertex $v \neq r$ has a 0-edge entering it.

- if there is a 0-SA composed of 0-edges, then it is an MSA;
- otherwise, there must be a 0-circuit. //backtracking

In $O(m)$ time, we can find either a 0-SA T, or a 0-circuit C.

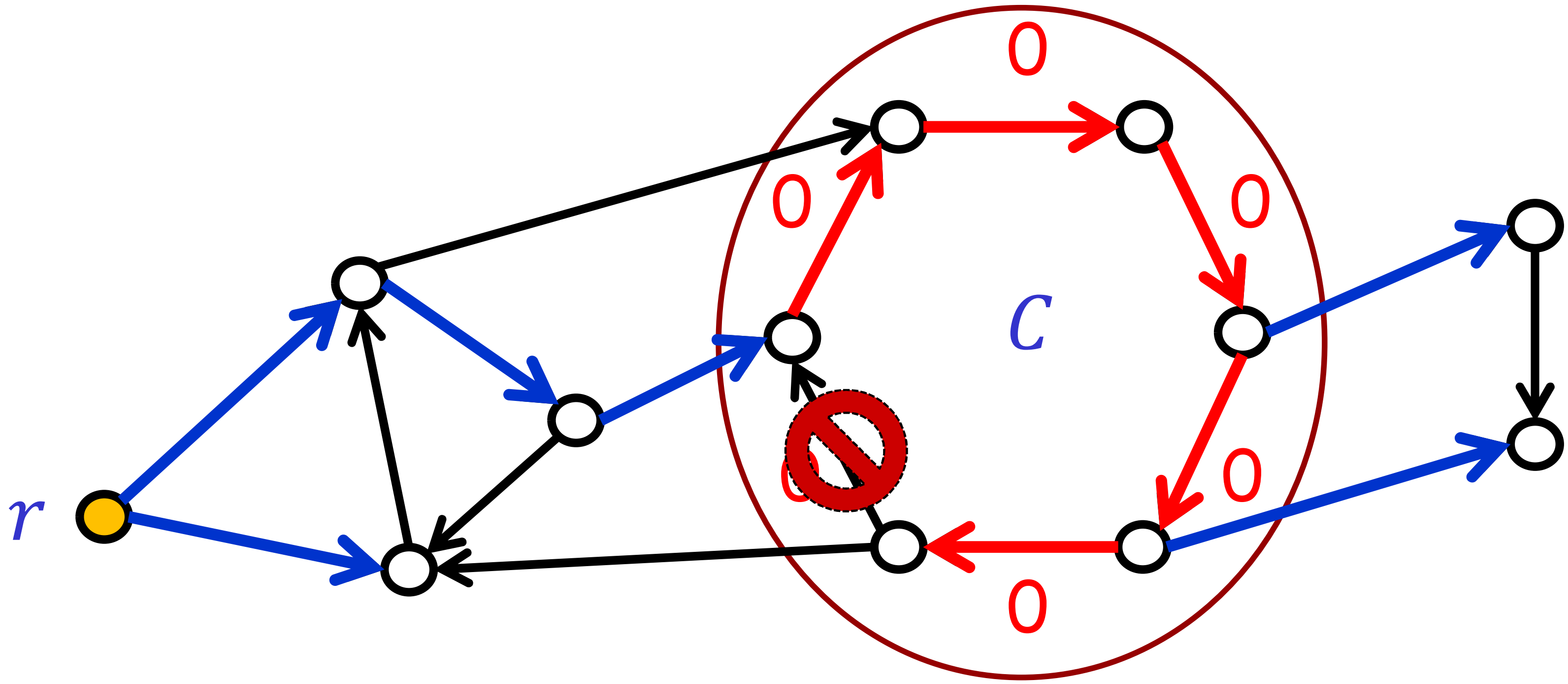
Contract & Conquer

Theorem. Let T' be an MSA of G/C. Then the SA of G obtained by adding to T' all the edges of C except one, is an MSA of G.



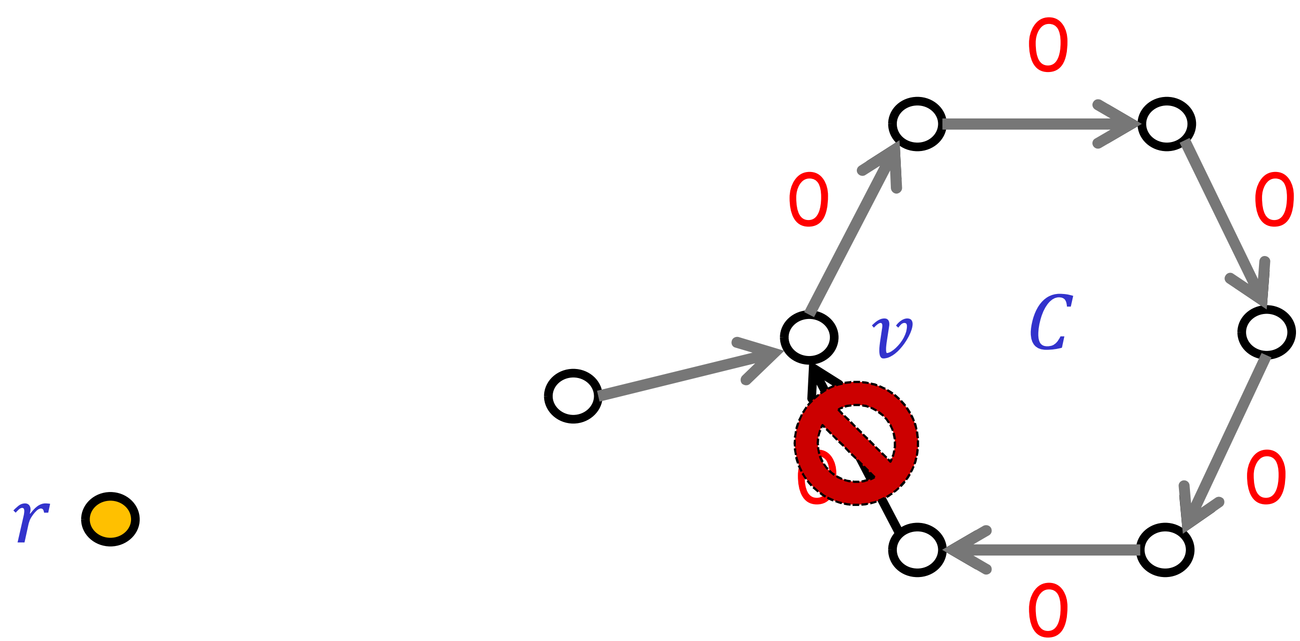
Contract & Conquer

Theorem. Let T' be an MSA of G/C. Then the SA of G obtained by adding to T' all the edges of C except one, is an MSA of G.

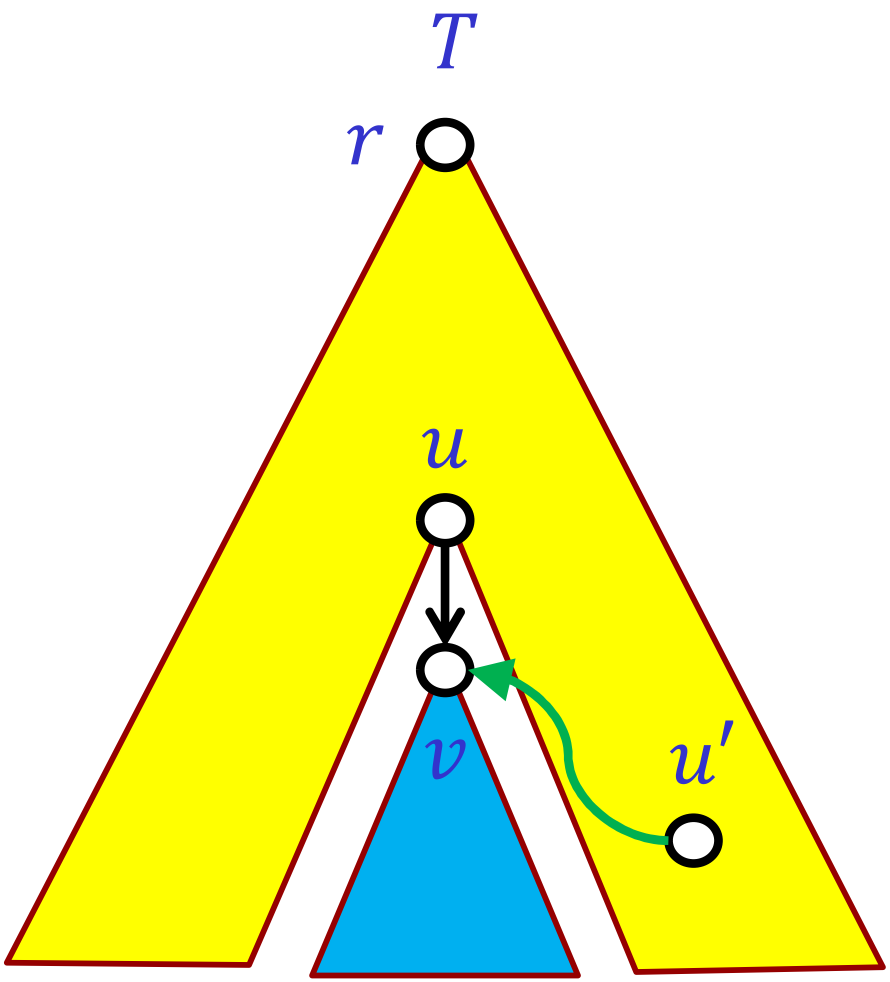


Single-entrance to 0-Circuit

Lemma. There is a MSA that contains only one edge that enters C, at some vertex v, and all the edges of C, except the one that enters v.



Parent replacement



T: a SA rooted at r,

and $(u, v) \in T$.

Let $(u', v) \notin T$, where
v is *not* an ancestor of u'.

Then,
$T' = T \setminus \{(u, v)\} \cup \{(u', v)\}$,
is also a SA.

Suppose T is an MSA enters C with fewest times but still more than once.

A path of T from r first enters C at v_0.

Let $v_0, v_1, \ldots, v_{k-1}, v_k = v_0$ be the vertices of C,

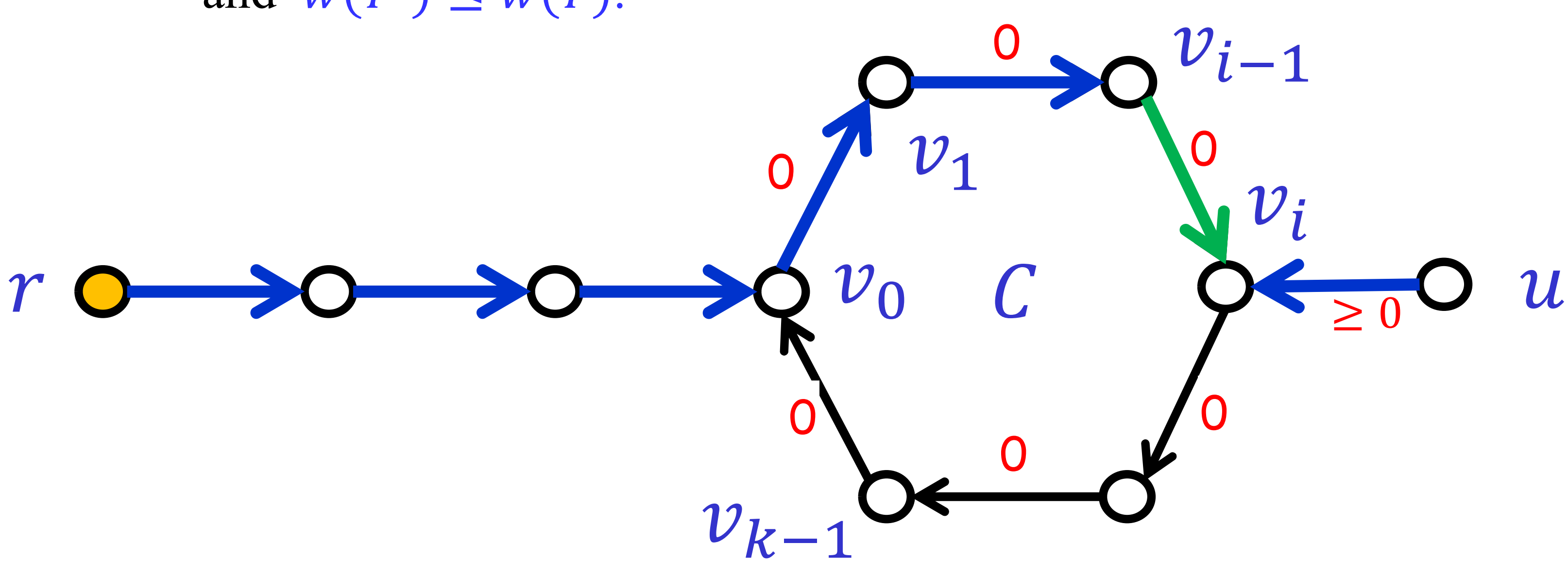
(v_{i-1}, v_i) be the *first* edge on C *not* in T, where $i < k$.

Let (u, v_i) be the edge of T entering v_i.

v_i is *not* an ancestor of v_{i-1} in T.

Hence, $T' = T \setminus \{(u, v_i)\} \cup \{(v_{i-1}, v_i)\}$ is a SA,

and $w(T') \leq w(T)$.



Contract & Conquer Algorithm

[Chin and Liu 1965] [Edmonds 1967] [Bock 1971]

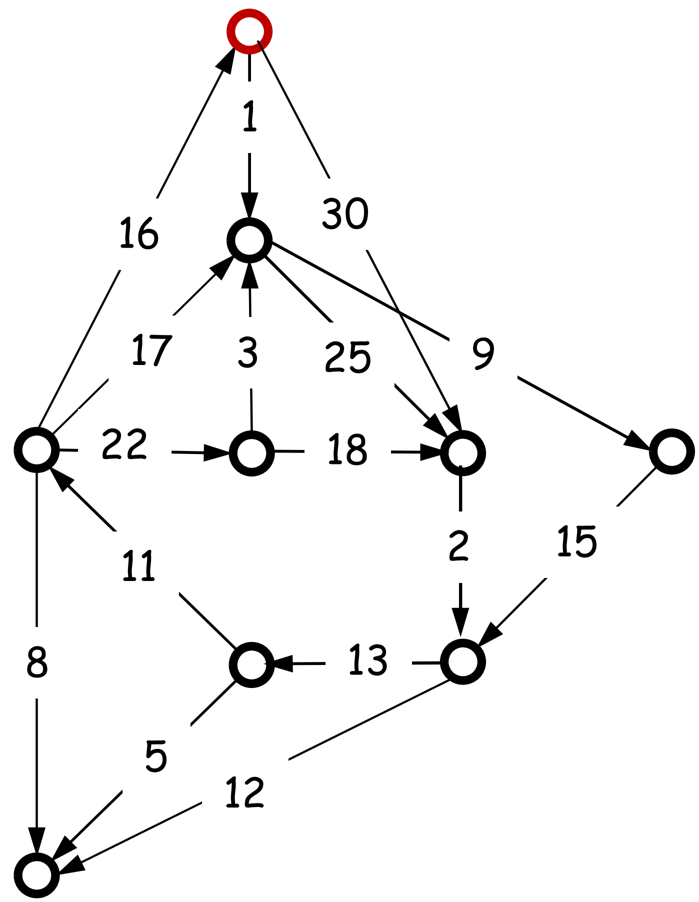
initialize G' by the basic weight resetting;

while G' has no 0-SA
    find a 0-circuit C and *contract* it;
    perform the basic weight resetting at C;

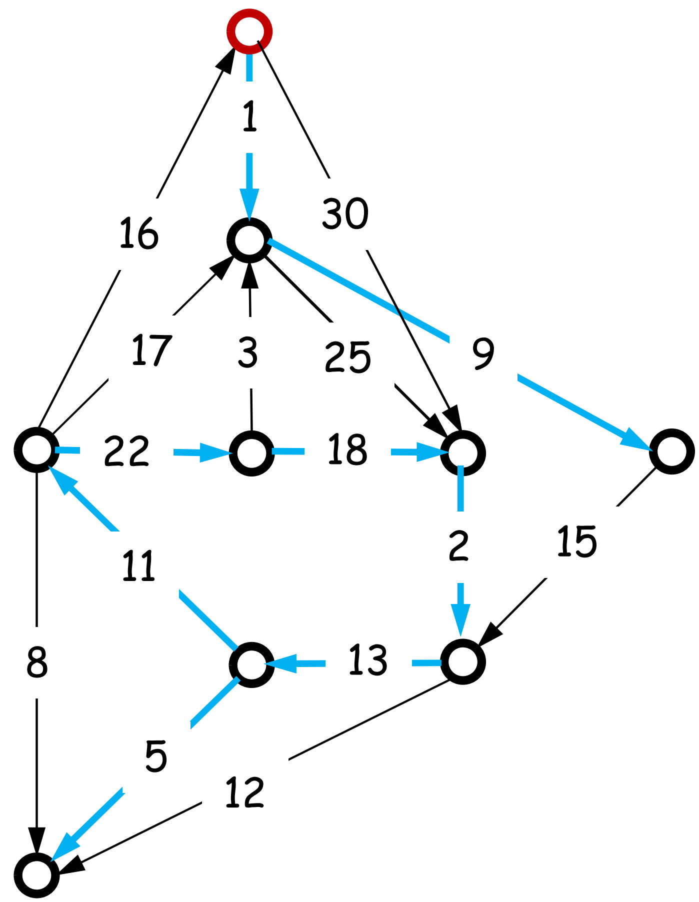
expand the 0-SA into an MSA in G.

Analysis: number of iterations is at most n.
Simple implementation in $O(mn)$ time.

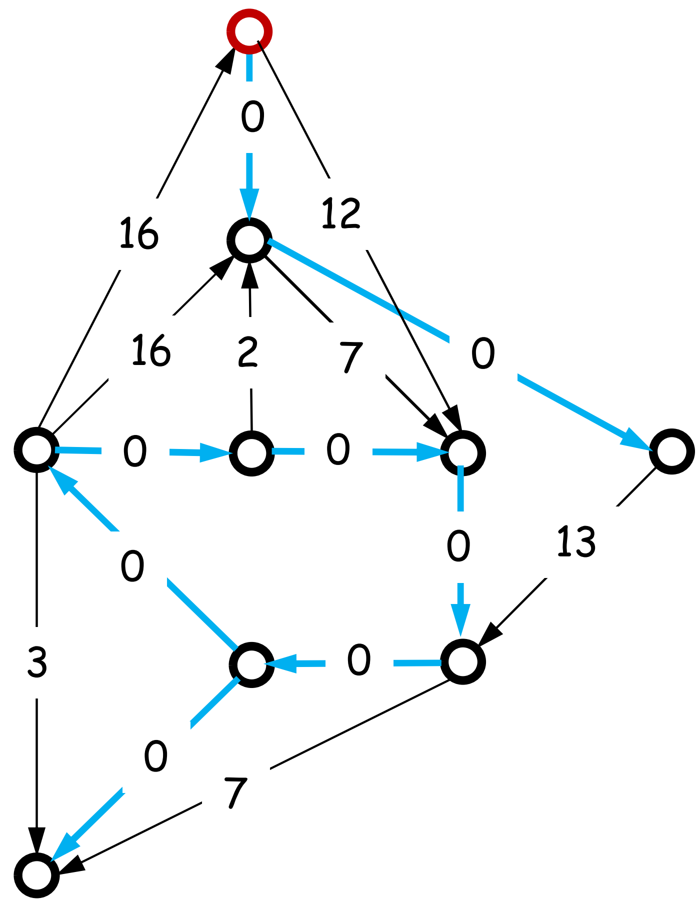
Demo



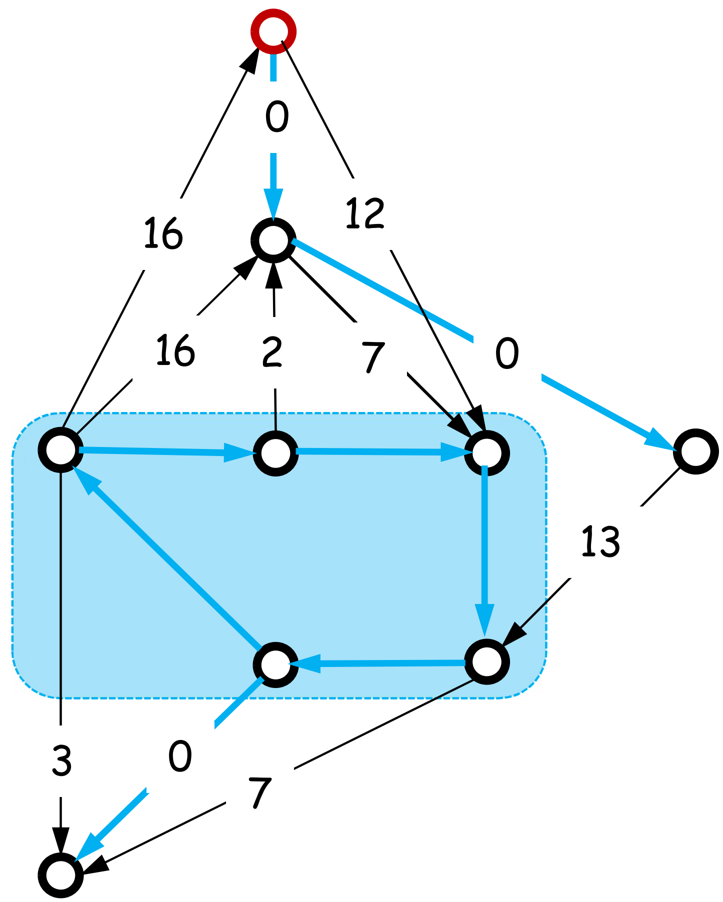
Demo



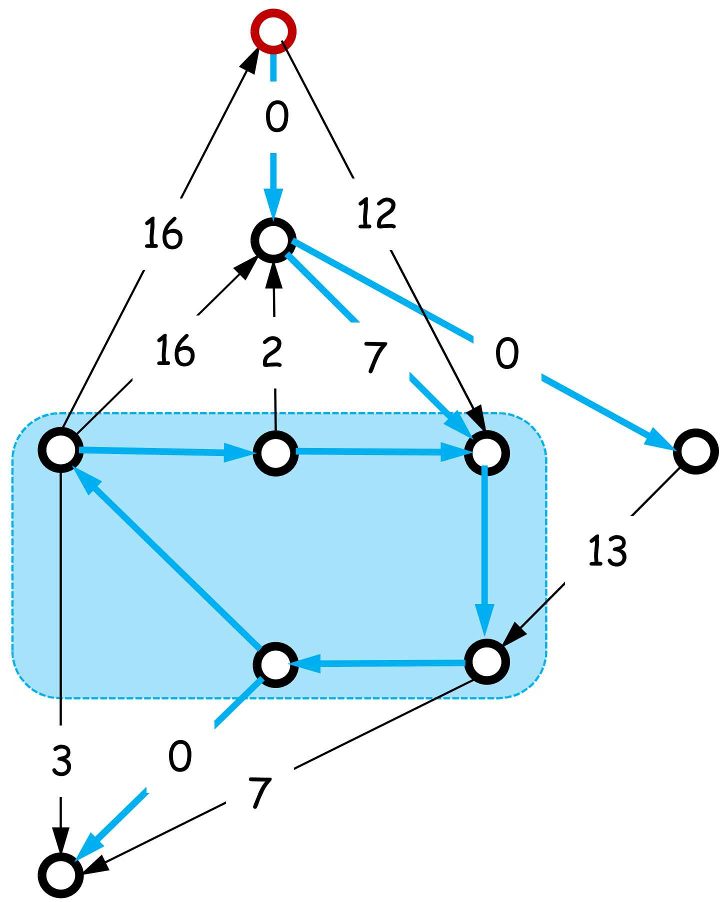
Demo



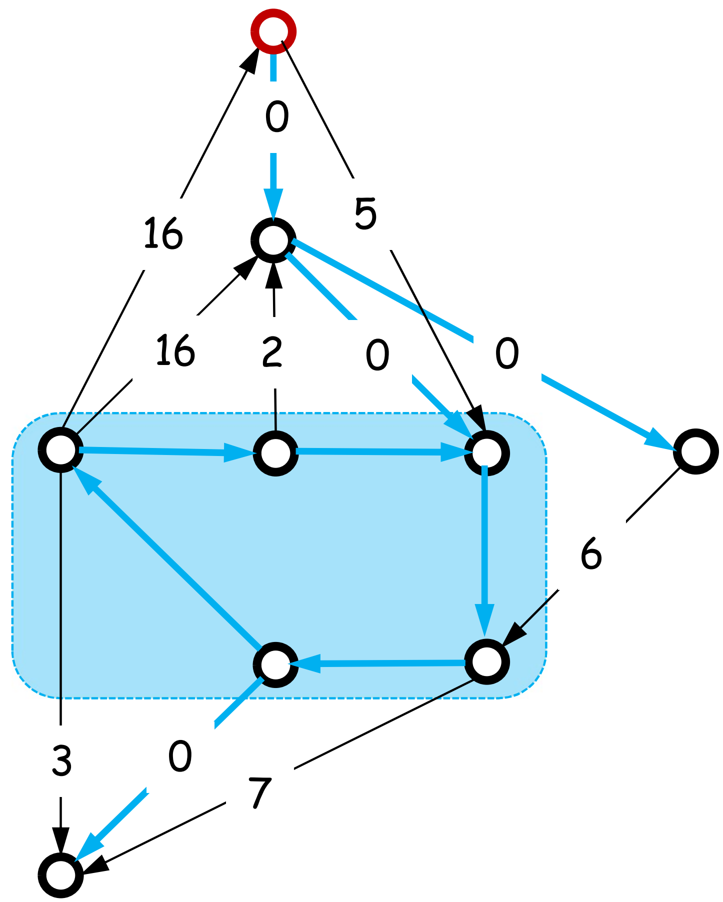
Demo



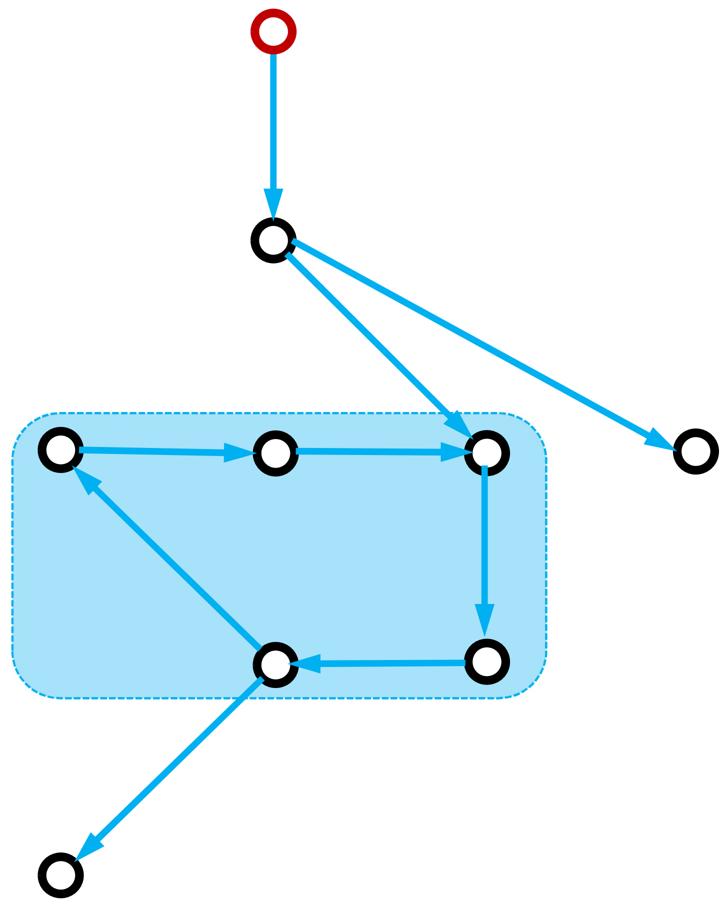
Demo



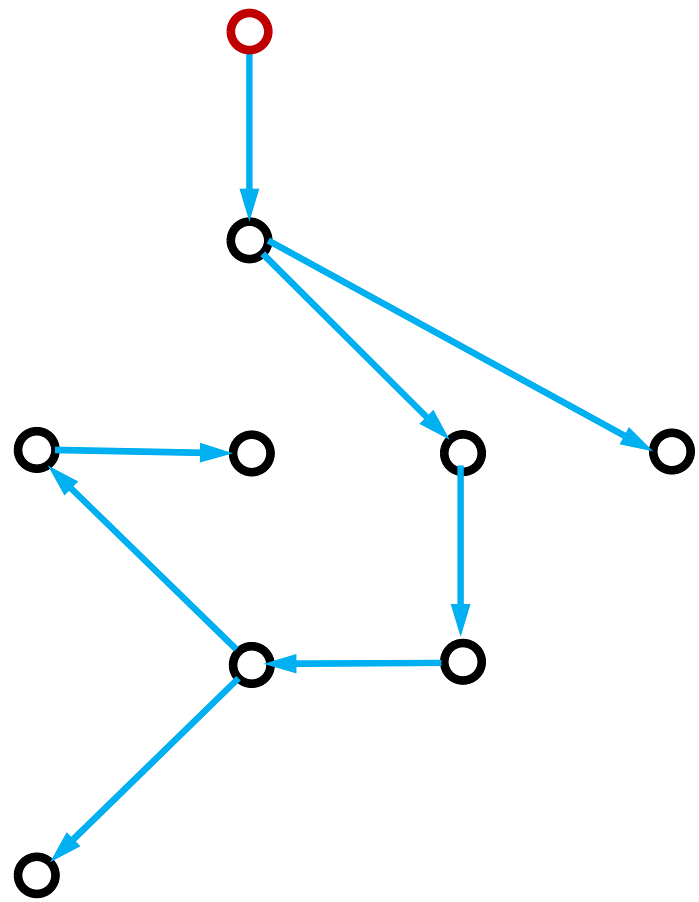
Demo



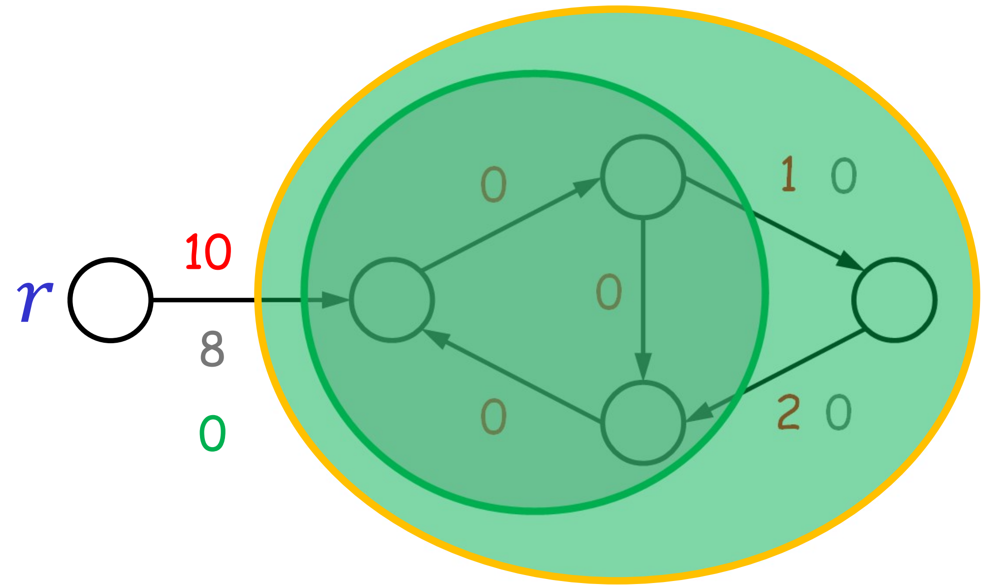
Demo



Demo



Another Small Example

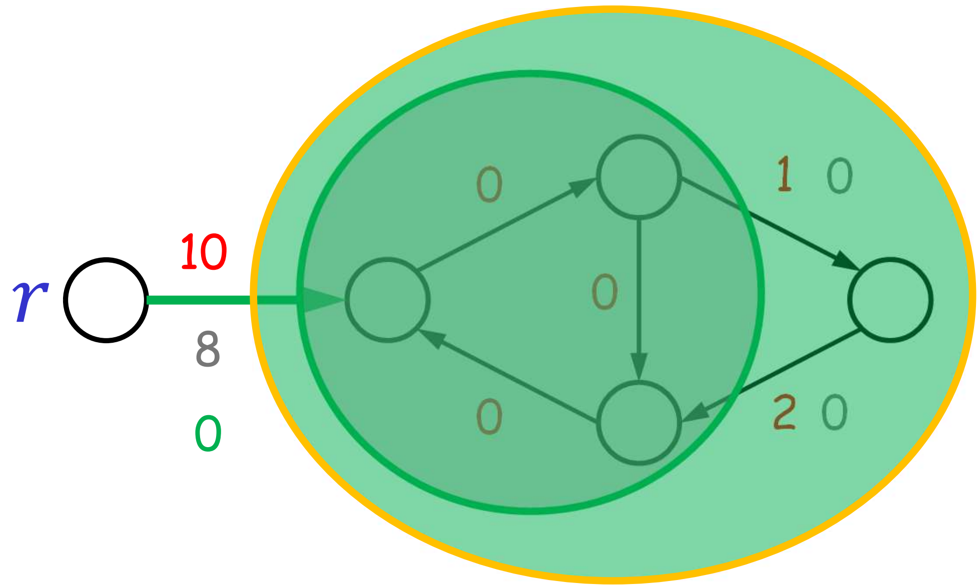


All edges end up with weight 0.

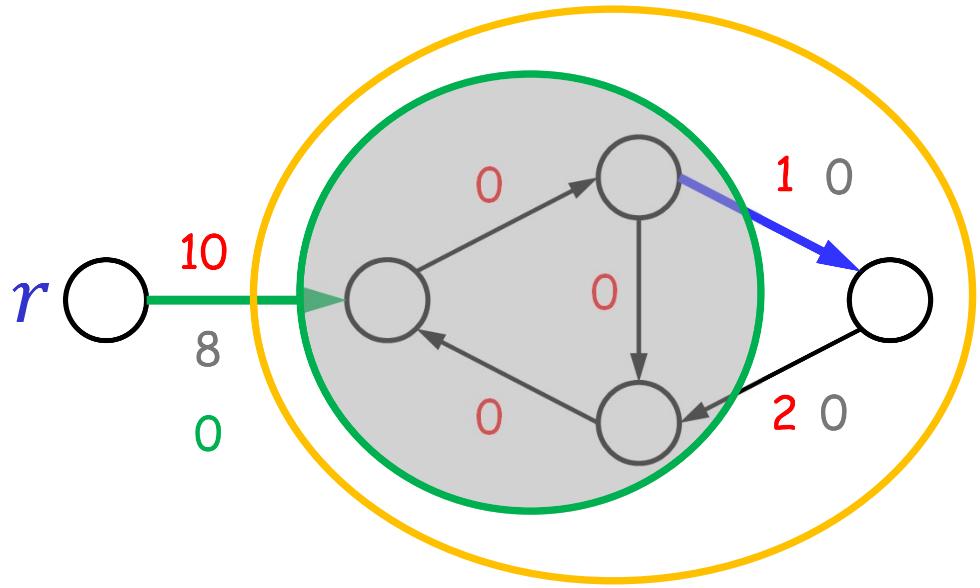
But, *not* every SA is an MSA.

A SA is an MSA iff it enters both circuits *only once*.

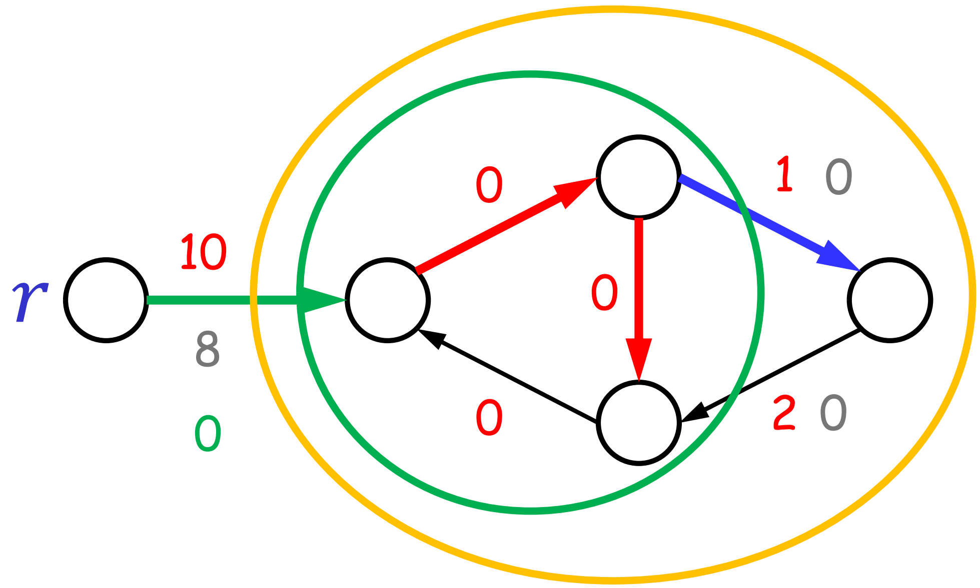
Another Small Example



Another Small Example



Another Small Example



Minimum total weight

Thm. Suppose $C_1, C_2, \ldots, C_k$ is the sequence of circuits contracted by the algorithm, and $w_1, w_2, \ldots, w_k$ are the weights of the *lightest* edges entering them. Then, the weight of the MSA is $\sum_{i=1}^{k} w_i$.

No need for expansion/lift if only the MSA weight is needed!