Stoer-Wagner Algorithm

A Minimum Cut Algorithm for Undirected Graphs

BigNews

CS214: Algorithms and Complexity
Shanghai Jiao Tong University

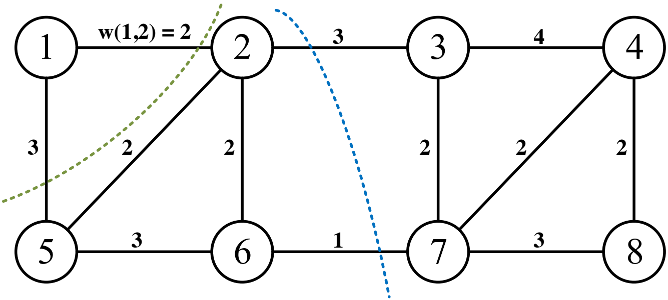
2016.12.06

Outline

Global Minimum Cut

Definition

Given an undirected graph $G(V, E)$, a global min-cut is a partition of V into two subsets (S, T) such that the sum of weights of edges between S and T is minimized.



Outline

The Algorithm
Global Minimum Cut
Stoer-Wagner Algorithm

Correctness
Lemma & Theorem
Proof

Running Time
Proof

Stoer-Wagner Algorithm (1)

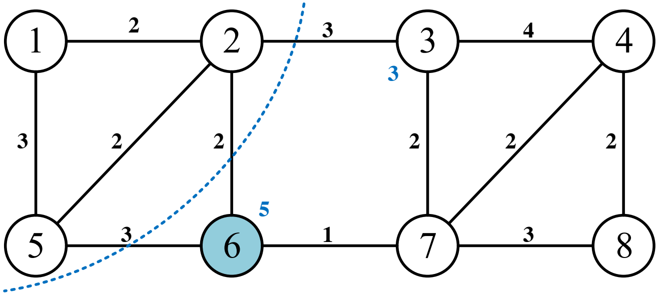
MinimumCutPhase(G, w, a)

$A \leftarrow \{a\}$
while $(A \neq V)$ **do**
    add to A the most tightly connected vertex
end
store the *cut-of-the-phase*
shrink G by merging the two vertices added last

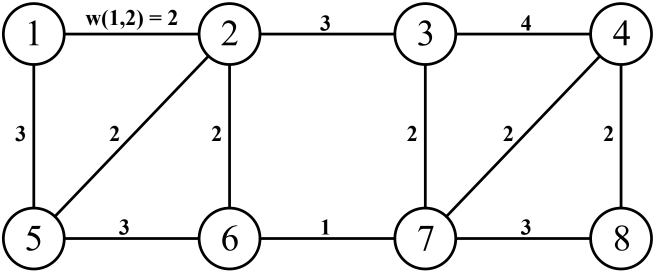
Most Tightly Connected

Definition

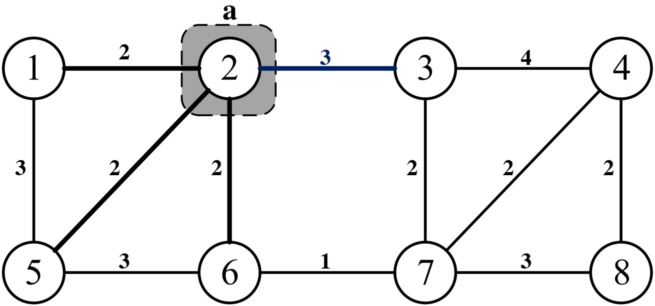
A vertex t are called **most ptightly connected** to vertex set A if sum of the weights of the edges connected between t and A is the highest among other vertices not belonging to A.



A graph $G = (V, E)$ with edge-weights

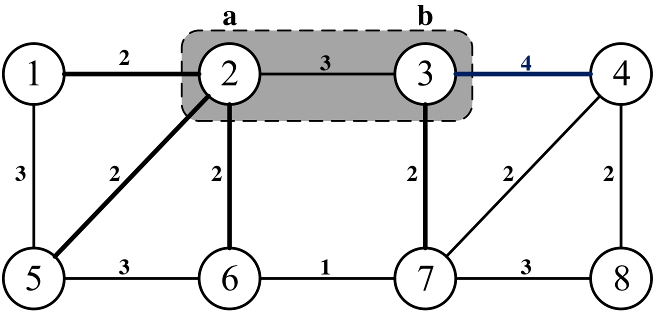


Start with vertex a = 2



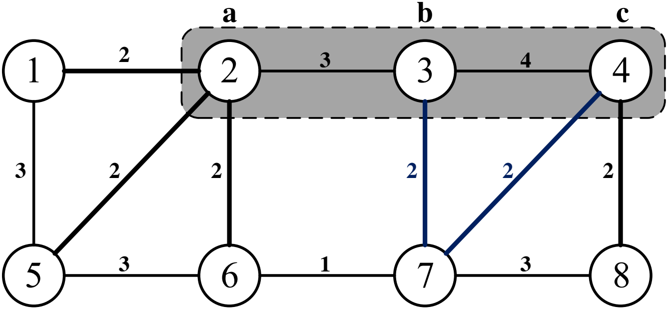
$A = \{2\}$

Add vertex b = 3



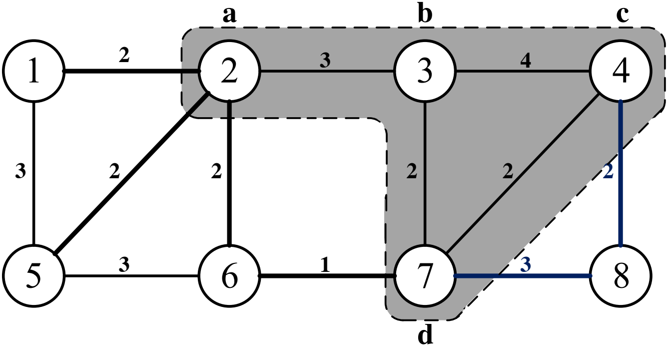
$A = \{2, 3\}$

Add vertex c = 4



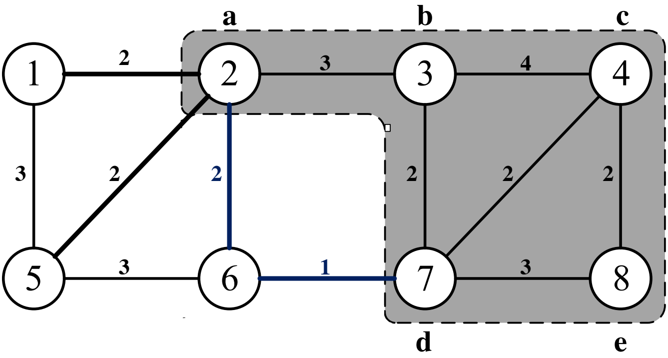
$A = \{2, 3, 4\}$

Add vertex d = 7



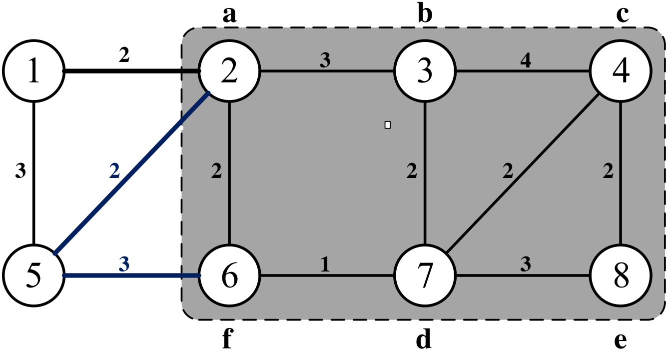
$A = \{2, 3, 4, 7\}$

Add vertex e = 8



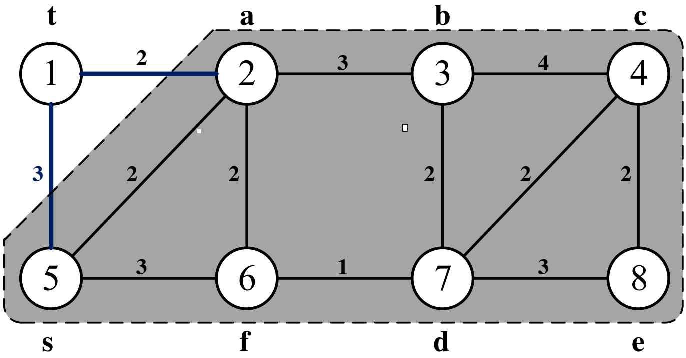
$A = \{2, 3, 4, 7, 8\}$

Add vertex f = 6



$A = \{2, 3, 4, 7, 8, 6\}$

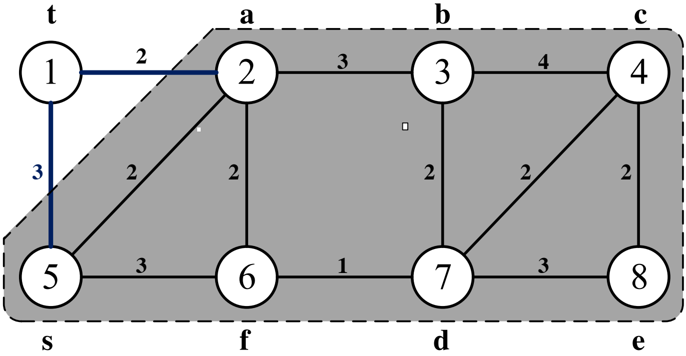
Add vertex s = 5 and vertex t = 1



$A = \{2, 3, 4, 7, 8, 6, 5\}$
$A = \{2, 3, 4, 7, 8, 6, 5, 1\}$
s and t are the last two vertices (in order) added to A, and we get a cut $C(A\text{-}t, t)$, which is so-called *cut-of-the-phase*.

Lemma

Each ***cut-of-the-phase*** *is a* ***minimum*** **s-t** ***cut*** *in the current graph, where* **s** *and* **t** *are the two vertices added last in the phase.*



Implications

What if the global min cut of G separates s and t?

Implications

What if the global min cut of G separates s and t?

Then min s-t cut is also a global min cut of G.

Implications

What if the global min cut of G separates s and t?

Then min s-t cut is also a global min cut of G.

What if min cut of G does not separate s and t?

Implications

What if the global min cut of G separates s and t?

Then min s-t cut is also a global min cut of G.

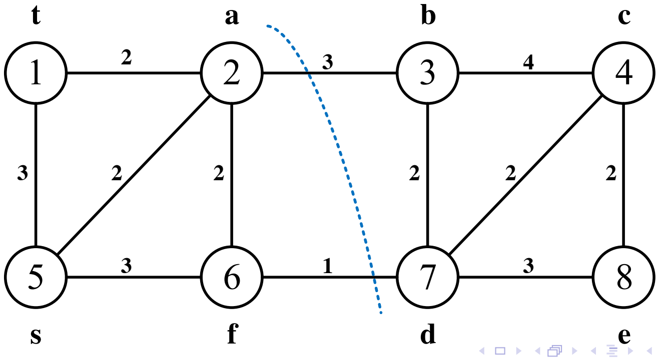
What if min cut of G does not separate s and t?

Then s and t are in the same partition of the global min cut, and we can **merge** them without changing the global min cut.

Merge

Definition

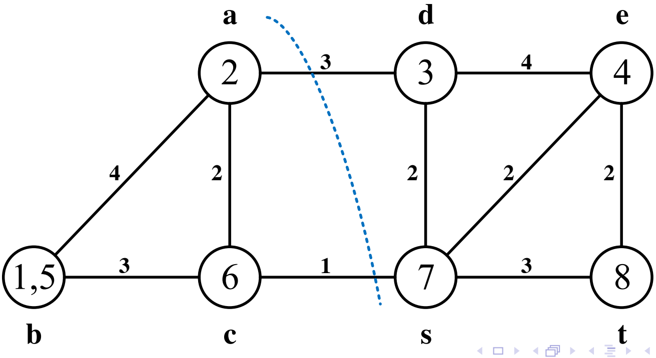
The two vertices are replaced by a new vertex and any edges from the two vertices to a remaining vertex are replaced by an edge weighted by the sum of the weights of the previous two edges, while edges joining the merged nodes are removed.



Merge

Definition

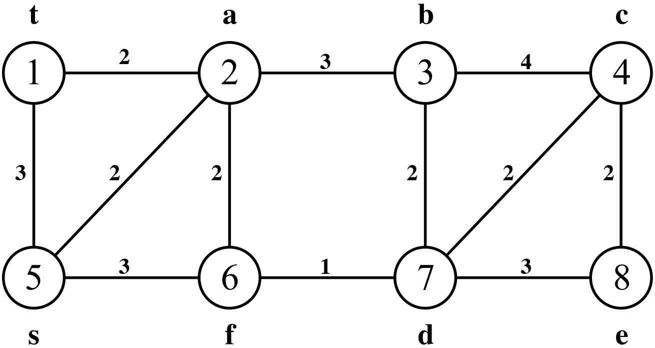
The two vertices are replaced by a new vertex and any edges from the two vertices to a remaining vertex are replaced by an edge weighted by the sum of the weights of the previous two edges, while edges joining the merged nodes are removed.



Stoer-Wagner Algorithm (2)

```
MinimumCut(G, w, a)
while (| V | > 1) do
    MinimumCutPhase(G, w, a)
    if the cut-of-the-phase is lighter than the current minimum cut
    then
        store the cut-of-the-phase as the current minimum cut
    end
end
```

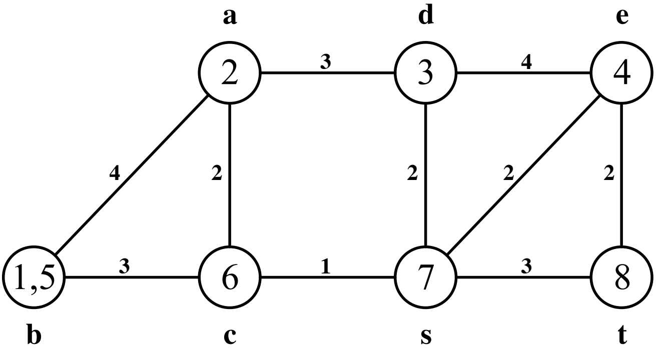
After the 1^{st} MinimumCutPhase(G, ω, a), $a = 2$



vertex ordering: a, b, c, d, e, f, s, t

cut-of-the-phase: $\{1\}, \{2, 3, 4, 5, 6, 7, 8\}$ $\quad \omega = 5$

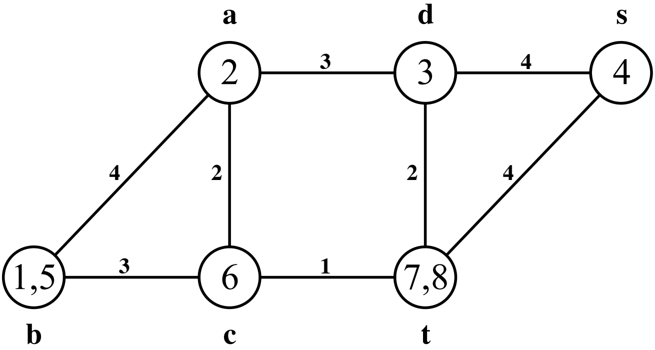
After the 2^{nd} MinimumCutPhase(G, ω, a), $a = 2$



vertex ordering: a, b, c, d, e, s, t

cut-of-the-phase: $\{8\}, \{1, 2, 3, 4, 5, 6, 7\}$ $\quad \omega = 5$

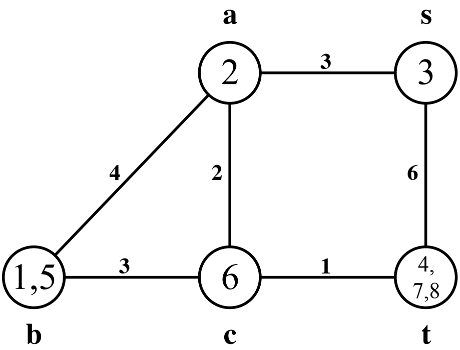
After the 3^{rd} MinimumCutPhase(G, ω, a), $a = 2$



vertex ordering: a, b, c, d, s, t

cut-of-the-phase: $\{7, 8\}, \{1, 2, 3, 4, 5, 6\}$ $\quad \omega = 7$

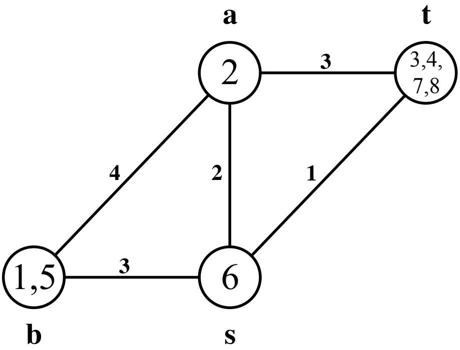
After the 4^{th} MinimumCutPhase(G, ω, a), $a = 2$



vertex ordering: a, b, c, s, t

cut-of-the-phase: $\{4, 7, 8\}, \{1, 2, 3, 5, 6\}$ $\quad \omega = 7$

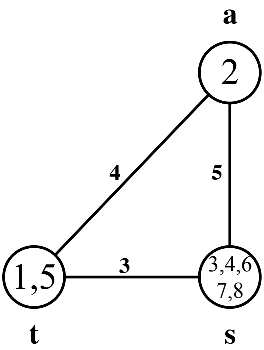
After the 5^{th} MinimumCutPhase(G, ω, a), $a = 2$



vertex ordering: a, b, s, t

cut-of-the-phase: $\{3, 4, 7, 8\}, \{1, 2, 5, 6\}$ $\quad \omega = 4$

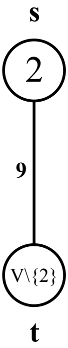
After the 6^{th} MinimumCutPhase(G, ω, a), $a = 2$



vertex ordering: a, s, t

cut-of-the-phase: $\{1, 5\}, \{2, 3, 4, 6, 7, 8\}$ $\omega = 7$

After the 7^{th} MinimumCutPhase(G, ω, a), $a = 2$



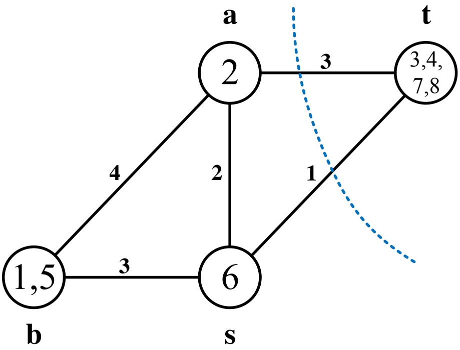
vertex ordering: s, t

cut-of-the-phase: $\{2\}, \{1, 3, 4, 5, 6, 7, 8\}$ $\quad \omega = 9$

Cut-of-the-phase

cut-of-the-phase	ω
$\{1\};\{2,3,4,5,6,7,8\}$	5
$\{8\};\{1,2,3,4,5,6,7\}$	5
$\{7,8\};\{1,2,3,4,5,6\}$	7
$\{4,7,8\};\{1,2,3,5,6\}$	7
$\{\mathbf{3},\mathbf{4},\mathbf{7},\mathbf{8}\};\{\mathbf{1},\mathbf{2},\mathbf{5},\mathbf{6}\}$	**4**
$\{1,5\};\{2,3,4,6,7,8\}$	7
$\{2\};\{1,3,4,5,6,7,8\}$	9

The Minimum Cut of the Graph G



vertex ordering: a, b, s, t

cut-of-the-phase: $\{3, 4, 7, 8\}, \{1, 2, 5, 6\}$ $\quad \omega = 4$

Outline

Lemma

*Each **cut-of-the-phase** is a **minimum*** **s-t** ***cut*** *in the current graph, where* **s** *and* **t** *are the two vertices added last in the phase.*

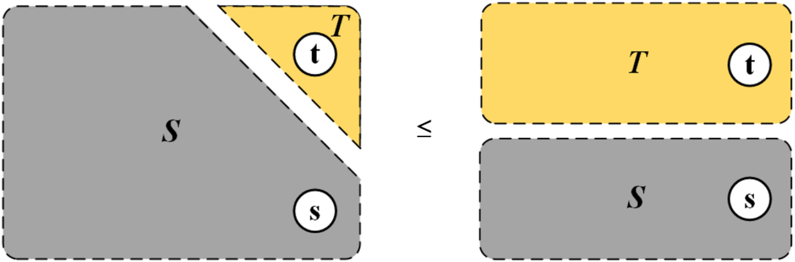
Theorem

*The **lightest** of these **cuts-of-the-phase** is the **minimum cut** of G.*

Assuming that the lemma holds, the theorem can be proved by a simple case distinction. Thus our proof is focused on the claimed property of the *cut-of-the-phase*.

Outline

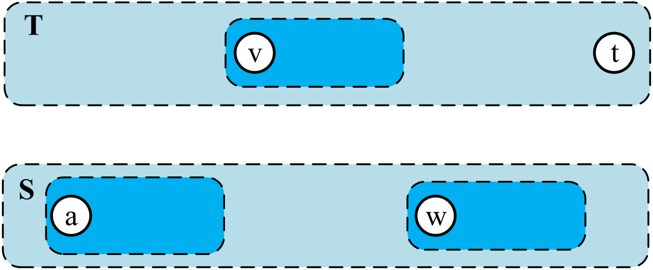
Proof



$\omega(A_t, t) \leq \omega(C_t)$

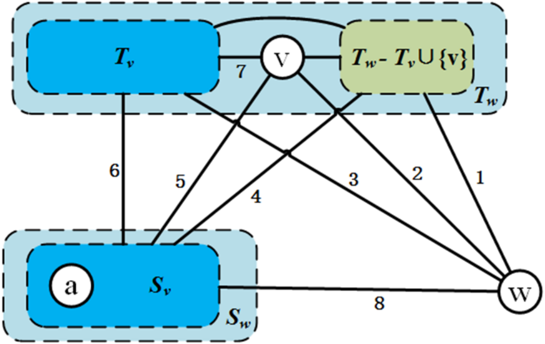
Definition

We call a vertex v *active* when v and the vertex added just before v are in different parts of the cut.



active vertices: $a, \ldots, v, w, \ldots, t$

Proof

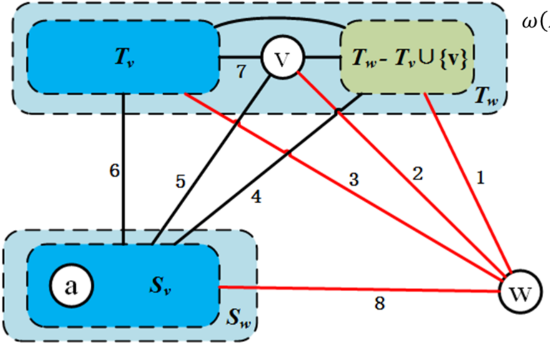


$$A_v = S_v \cup T_v$$

$$A_w = S_w \cup T_w$$

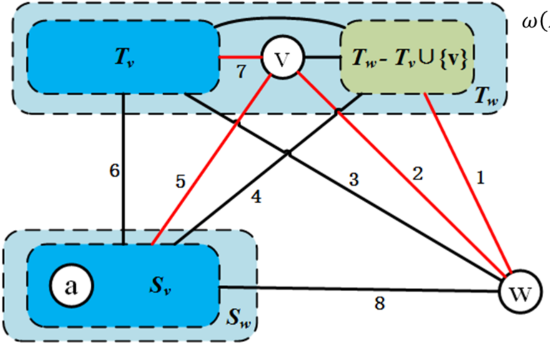
$$S_w = S_v$$

Proof



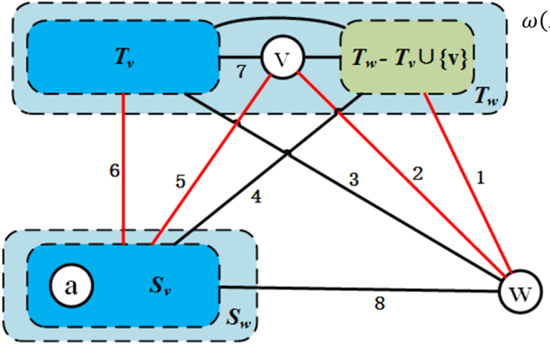
$$\omega(A_w, w) = (1+2) + (3+8)$$
$$= \omega(A_w \backslash A_v, w) + \omega(A_v, w)$$

Proof



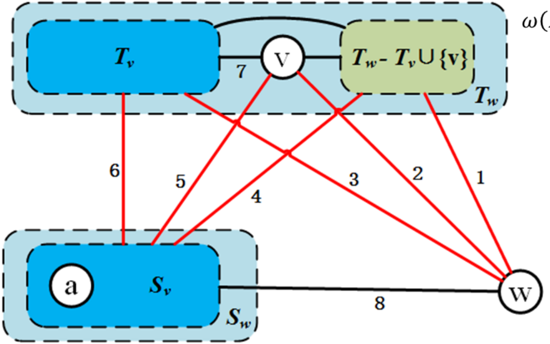
$$\begin{aligned}\omega(A_w, w) &= (1+2)+(3+8)\\ &= \omega(A_w \backslash A_v, w) + \omega(A_v, w)\\ &\leq \omega(A_w \backslash A_v, w) + \omega(A_v, v)\\ &= (1+2)+(5+7)\end{aligned}$$

Proof



$$\begin{aligned}
\omega(A_w, w) &= (1+2) + (3+8) \\
&= \omega(A_w \backslash A_v, w) + \omega(A_v, w) \\
&\leq \omega(A_w \backslash A_v, w) + \omega(A_v, v) \\
&= (1+2) + (5+7) \\
&\leq \omega(A_w \backslash A_v, w) + \omega(C_v) \\
&= (1+2) + (5+6)
\end{aligned}$$

Proof



$$\omega(A_w, w) = (1+2) + (3+8)$$
$$= \omega(A_w \backslash A_v, w) + \omega(A_v, w)$$
$$\leq \omega(A_w \backslash A_v, w) + \omega(A_v, v)$$
$$= (1+2) + (5+7)$$
$$\leq \omega(A_w \backslash A_v, w) + \omega(C_v)$$
$$= (1+2) + (5+6)$$
$$\leq \omega(C_w)$$
$$= (1+2+3+4+5+6)$$

Outline

MinimumCut Algorithm

MinimumCut(G, w, a)

while $(|V| > 1)$ **do**

  MinimumCutPhase(G, w, a)

  if the *cut-of-the-phase* is lighter than the current minimum cut **then**

    store the *cut-of-the-phase* as the current minimum cut

  end

end

MinimumCutPhase Algorithm

MinimumCutPhase(G, w, a)

$A \leftarrow \{a\}$
while $(A \neq V)$ **do**
  add to A the most tightly connected vertex
end
store the *cut-of-the-phase*
shrink G by merging the two vertices added last

MinimumCutPhase Algorithm

MinimumCutPhase(G, w, a)

$A \leftarrow \{a\}$
while $(A \neq V)$ **do**
    add to A the most tightly connected vertex
end
store the *cut-of-the-phase*
shrink G by merging the two vertices added last

Proposition

The running time for a single phase is $O(|E| + |V| \log |V|)$.

Proof.

1. All vertices that are not in A reside in a priority queue.

$\square$

Proposition

The running time for a single phase is $O(|E| + |V| \log |V|)$.

Proof.

1. All vertices that are not in A reside in a priority queue.
2. Whenever a vertex v is added to A,
   the priority queue does *ExtractMax* and *IncreaseKey* operations.

$\square$

Proposition

The running time for a single phase is $O(|E| + |V| \log |V|)$.

Proof.

1. All vertices that are not in A reside in a priority queue.
2. Whenever a vertex v is added to A,
   the priority queue does *ExtractMax* and *IncreaseKey* operations.
3. $|V|$ *ExtractMax* $|E|$ *IncreaseKey*

$\square$

Proposition

The running time for a single phase is $O(|E| + |V| \log |V|)$.

Proof.

1. All vertices that are not in A reside in a priority queue.
2. Whenever a vertex v is added to A,
   the priority queue does *ExtractMax* and *IncreaseKey* operations.
3. $|V|$ *ExtractMax* $|E|$ *IncreaseKey*
4. Fibonacci heaps
   ExtractMax $O(\log |V|)$
   IncreaseKey $O(1)$
   Time for one phase is $O(|E| + |V| \log |V|)$.

□

Proposition

The overall running time of Stoer-Wagner algorithm for minimum cut in undirected graphs is $O(|V||E| + |V|^2 \log |V|)$.

Proof.

1. $|V| - 1$ phases
2. Time for single phase $O(|E| + |V| \log |V|)$
3. Overall running time $O(|V||E| + |V|^2 \log |V|)$

$\square$

Thank you!

Q & A