① Show if G is disconnected

$\Rightarrow \bar{G}$ is connected

(other direction is equivalent logically)

Recall: $\forall u,v \in V(G)$: $\exists$ u,v-path $\Rightarrow$ connected

Consider any arbitrary $u,v \in V(G)$

Case 1: u,v in the same component

$\rightarrow \exists w \in V(G)$: w is in diff comp.

$\Rightarrow$ in $\bar{G}$, u and v are connected

via path $P = \{(u,w),(w,v)\}$

Case 2: u,v in different components

$\rightarrow$ in $\bar{G}$ $\exists e = (u,v) \in E(\bar{G})$

$\Rightarrow$ hence trivial path $P = \{(u,v)\}$

exists in $\bar{G}$ $\square$

② G connected $\Leftrightarrow$

$\forall\, V_1 V_2 \subseteq V(G)$ where $V_1 \cup V_2 = V(G)$ (union)

$\exists\, e = (u, v) : u \in V_1,\ v \in V_2$

($\Rightarrow$) contrapositive

$\exists\, V_1 V_2 \subseteq V(G)$ s.t. there is no $e = (u, v)$
$u \in V_1, v \in V_2$

$\Rightarrow G$ is disconnected

→ as there is no edge between any pair of vertices in V_1, V_2, there is therefore no path between them

$\Rightarrow$ they are separate components and G is disconnected ✓

($\Leftarrow$) using contrapositive again

→ as G is disconnected, $\exists$ at least

2 components

→ assign 1 component's verts to V_1

→ all other are in V_2

=> as V_1 is a disconnected comp, there are no edges to any vertex in V_2 □

(3) Weak induction on $V(G)$

Basis $P(1)$: o ← single vert spans itself, no cuts, so trivially contains all cuts ✓

I.H. on connected $P(k)$

↳ we have connected $S \subseteq P(k)$ where S has all cut vertices and edges

$P(k+1) = P(k) + v$

Case 1: v is attached via a single edge $e = (u, v)$ to some u

→ u is a cut vertex and e is a cut edge, all other existing a cut edge, all other existing cuts are otherwise unaffected

$\Rightarrow S' = S + (u, v)$ is a valid connected spanning subgraph for $P(k+1)$ ✓

Case 2: v is connected to some number of $\{u_1 u_2 u_3 \dots u_n\}$

→ no (v, u_i) is a cut edge and no u_i is a cut vertex unless it already was before

$\Rightarrow S' = S + (v, u_i)$ will be a valid spanning subgraph for $P(k+1)$ □

④ we have $e = (u, v)$

→ note: deleting u or v will also delete e, so $G-u$ or $G-v$ is disconnected

=> so u or v must be a cut vertex ✓

(both are unless $d(u)=1$ or $d(v)=1$)

→ every edge in some disconnecting set is incident on some vertex in a separating set of equal size

→ a vertex could have multiple incident edges in the same set

=> $K(G) \leq K'(G)$ □

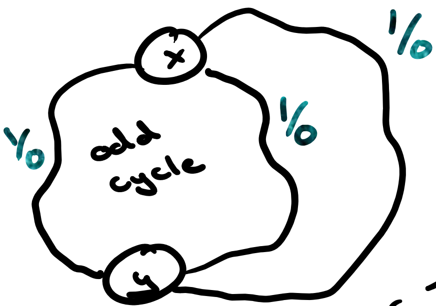
(5) Consider an ear decomposition of some block $b_i \in G$

b_i must have some odd cycle or is otherwise K_2

Q: can we add an open ear to our cycle?

A: No, consider parity of the

A: No, consider parity of the two paths of the cycle between the x, y endpoints of the ear and of the x,y-path of the ear itself



→ we have 3 x,y-paths

cycle { 1. at least one is odd
2. one is even

ear → 3. one is odd/even

$\Rightarrow$ No matter what, we can always combine 2 paths to create an even cycle, so no ears exist □

(6) $\forall v \in V(G): d(v)$ even $\Leftrightarrow \forall v \in B_i : d(v)$ is even

($\Leftarrow$) The degree of $v \in V(G)$ is the sum of degrees of that vertex in each of its B_i subgraphs

parity $\quad \Sigma \text{even} = \text{even}$

(⇒) Strong induction on # blocks

$$P(1): \underset{v \in V(G)}{\overset{\text{even}}{d(v)}} \overset{=}{=} \underset{v \in B_i}{\overset{\text{even}}{d(v)}} \checkmark$$

degrees trivially equal with only one block

P(n): we have G with P(n) blocks

$P(k) = P(n) - B_i$ ← some leaf block (only attached to one cut vertex → exists as block-cutpoint graph is tree)

We note that only some cut vertex v is affected by deletion of block

→ per degree sum formula, this vertex must have an even degree in B_i (all others are even)

1/0 → as even − even = even, all vertices
($d(v)$ in P(n), $d(v)$ in B, $d(v)$ in P(k))

1/0 parity 1/0 1/0

in P(k) have even degree

I.H. on P(k) gives us all degrees within blocks as even

⇒ P(k) → P(n) is trivial □

⑦ We'll consider various $\Delta(G)$ values

→ Assume G is connected, else $K(G) = K'(G) = 0$ trivially

• Assume G is connected, so trivially

$\Delta(G) = 1 \rightarrow G$ is K_2

$\kappa(G) = \kappa'(G) = 1$ ✓

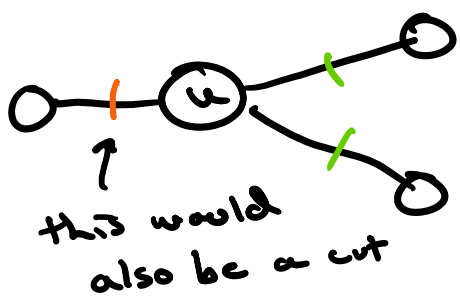
$\Delta(G) = 2 \rightarrow G$ is a path or cycle

path: $\kappa(G) = \kappa'(G) = 1$ cycle: $\kappa(G) = \kappa'(G) = 2$

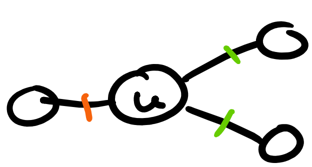
$\Delta(G) = 3 \rightarrow$ consider possible $\kappa'(G)$ values

$\kappa'(G) = 1 \rightarrow$ any endpoint of cut edge is a cut vertex, so $\kappa(G) = 1$

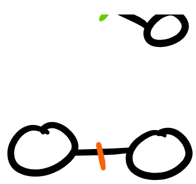
$\kappa'(G) = 2 \rightarrow$ we must have 2 edges on the cut with unique endpoints, other wise that implies a cut of 1, so $\kappa(G) = 2$



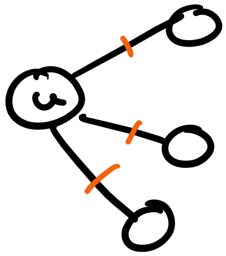
$\kappa'(G) = 3 \rightarrow$ any edge cut not using unique endpoints has two configurations



1. ← smaller cut of 2 implied, so this configuration can't

1. 

← smaller cut of 2 implied, so this configuration can't happen

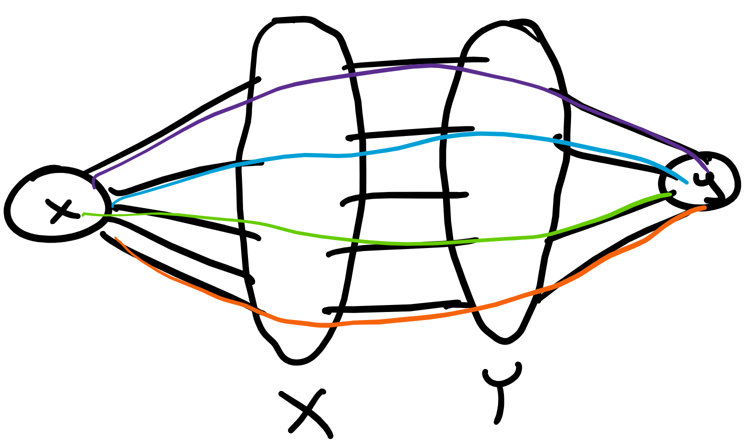
2. 

← vertex cut of 3 is implied, so $K'(G) = K(G) = 3$

→ Otherwise, our edge's endpoints are unique and therefore imply a vertex and edge cut of 3

$\Rightarrow$ All configurations have $K(G) = K'(G)$ □

⑧ Consider a construction as with our bipartite matching algorithm



Attach new x to all $u \in X$

Attach new y to all $v \in Y$

Now use Menger to give us $\lambda(x,y)$

Now use Menger to give us $\lambda(x,y)$ disjoint paths and therefore a vertex cut of size $K(x,y) = \lambda(x,y)$

→ our paths give us disjoint edges in the original bigraph, which we know must be a maximum match, given our construction

→ our cut is also a vertex cover, as an edge not covered by a cut vertex implies a larger match or a larger number of idps

⇒ min cut = max idps
⇕ ⇕
min cover = max match □

(9) Consider the same construction above, but:

* vertex x is source s
* vertex y is sink t

* vertex y is sink t

* direct all edges: s → X, Y → t (capacity 1)

  X → Y (capacity ∞)

→ this gives us a flow network, on which we can use the same logic as before (/and in WP7)

↳ follow unit of flow from s → X, X → Y, Y → t

↳ our flow paths from X → Y give us a max match due to unit capacity constraints on s → X and Y → t edges

→ max flow = min cut

↳ these are (s, X) & (Y, t) edges, we select the endpoints in X & Y, which are a min cover

(same logic as before, if they

(same logic as before, if they aren't a cover there would be a larger flow|match)

=> we have max flow = min cut

$\Updownarrow$ $\Updownarrow$ □

max match = min cover

(10) We know max flow = min cut

→ $k = \kappa'(s,t)$, so we have to remove at least k edges to disconnect s from t (assuming unit capacities)

→ and $\kappa'(s,t) = \lambda'(s,t)$ per Menger, so we have at least k edges from s to t (probably don't need this second statement)

=> removing any $k-1$ edges will leave at least 1 s,t-path □