# Induction

***Induction is a powerful proof technique that is widely used in computer science and mathematics. It has many variations, and we shall look at some of them.***

- Ordinary induction over N.
- Course-of-values induction over N.

***Induction over** N*

- Imagine an infinite sequence of dominoes standing on an infinite table, with Domino $n$+1 standing just behind Domino $n$, and someone pushes Domino 0.
- Then Domino 0 falls, causing Domino 1 to fall, causing Domino 2 to fall, causing Domino 3 to fall...
- What about Domino 1010100?
- It will eventually fall.
- Indeed it is obvious that *each domino will fall*.

This is the idea behind induction over N.

- Let $P$ be a property of natural numbers.

- Suppose that *P*(0)—this is called the *base case*.
- Suppose also that, for any natural number N, the statement *P*(*n*) → *P*(*n* + 1)
  - This is called the *inductive step*, and the hypothesis *P*(*n*) is called the *inductive hypothesis*
  - From these two facts, we may conclude that *every* natural number satisfies *P*, even big ones like 1010100.

***Example using induction to prove summation formula***

***Let's prove 0 + . . . + (n − 1) =*** $\frac{1}{2}n(n-1)$ *by induction on n* $\in$ N

(n − 1)n by induction on n $\in$ N

- Clearly this is true for *n* = 0, since the sum of no numbers is defined to be 0
- Assuming it's true for *n*, let's show that it's true for *n* + 1.

***Proof***

0 + · · · + ((*n* + 1) −1) = 0 + · · · + (*n* −1) + *n*

= $\frac{1}{2}n(n-1)+n$ (by the inductive hypothesis)

= $\frac{1}{2}n^2-\frac{1}{2}n+n$

= $\frac{1}{2}n^2+\frac{1}{2}n$

= $\frac{1}{2}n(n-1)$

= $\frac{1}{2}(n+1)((n+1)-1)$ as required.

$\epsilon$

***Variations***

Here are some variations. Let *P* be a property of natural numbers.

- Suppose we have proved that *P* holds for 0, 1 and 2, and also that, if it holds for *n*, *n* + 1 and *n* + 2, then it also holds for *n* + 3. We now know that *P* holds for all natural numbers.

- Suppose we have proved that *P* holds for 1 and 3, and also that, if it holds for *n* and *n* + 2, then it also holds for *n* + 4. We now know that *P* holds for all odd natural numbers.
- Suppose we have proved that *P* holds for 1, and also that, if it holds for *n*, then it also holds for 2*n*. We now know that *P* holds for every power of 2.

***Course-of-values induction***

- When we give a proof by ordinary induction, the inductive step proves that *P*(1) follows from *P*(0), that *P*(2) follows from *P*(1), that *P*(3) follows from *P*(2), and so on.
- But surely, when proving *P*(3), it should be acceptable to assume not just *P*(2) but also *P*(1) and *P*(0). This thinking leads to *course-of-values induction* (also called "strong induction").
- *The principle is as follows*
    - Let *P* be a property of natural numbers.
    - Suppose that, for any natural number *n*, the statement *P*(*n*) holds if *P* holds for all natural numbers less than *n*.
        - (The latter assumption is called the *inductive hypothesis*.)
    - **This means that:**
        - *P*(0)
        - if *P*(0), then *P*(1)
        - if *P*(0) and *P*(1), then *P*(2)
        - if *P*(0) and *P*(1) and *P*(2), then *P*(3)
        - etc.
    - From this fact, we may conclude that *every* natural number satisfies *P*, even big ones like 1010100.

***Example:***

*The merge sort algorithm is the following recursively defined algorithm for sorting a list p.1*

- If the length of *p* is 0 or 1, return *p*.
- If the length of *p* is 2*k* where *k* > 0, then sort the left part of length *k*, and sort the right part of length *k*, and merge the results.

- If the length of $p$ is $2k + 1$ where $k > 0$, then sort the left part of length $k$, and sort the right part of length $k + 1$, and merge the results.

How do we know that this algorithm terminates, and returns a list that is a sorted version of $p$?

- By course-of-values induction on the length of the list.
- In each of the three cases, it is easy to see that the algorithm yields a sorted version of $p$, assuming that it works correctly on shorter lists.
- The version given here is intended to return the sorted version of the list.
- The version for sorting an *array* with in-place update is slightly different, but the idea is the same.

***Example***

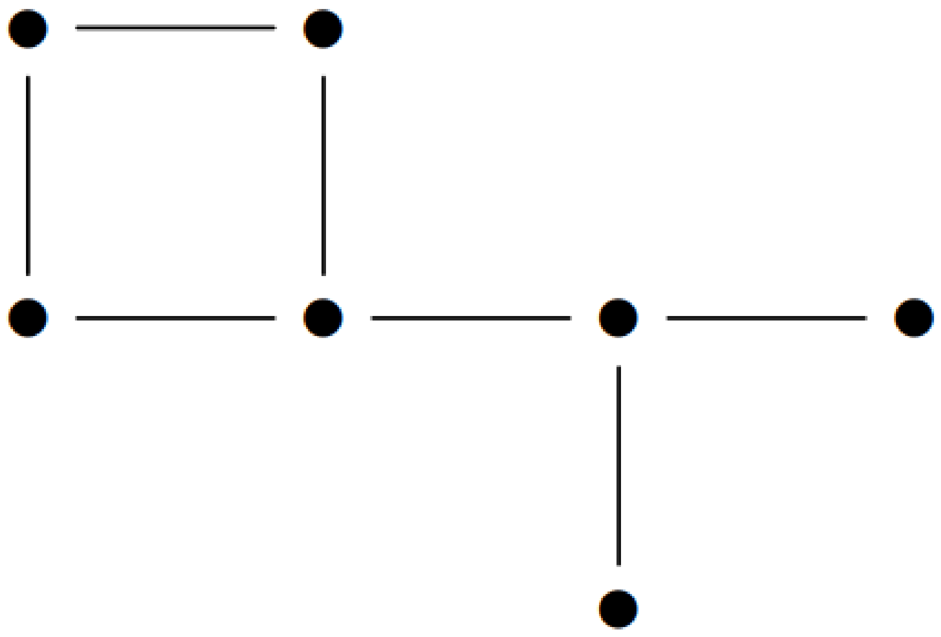
An undirected graph $G$ is *connected* when it has at least one vertex

AND

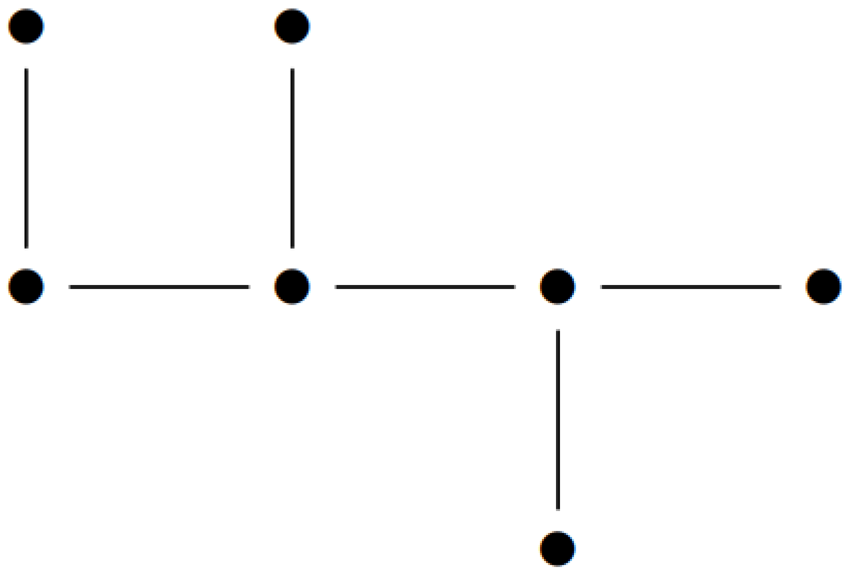
An undirected graph $G$ is *connected* when there is a path between any two vertices

Show that, if $G$ is connected, then $V(G) \leqslant E(G) + 1$, where $V(G)$ is the number of vertices and $E(G)$ the number of edges.

For example, if $G$ is



then $V(G) = E(G) = 7$, whereas if $G$ is

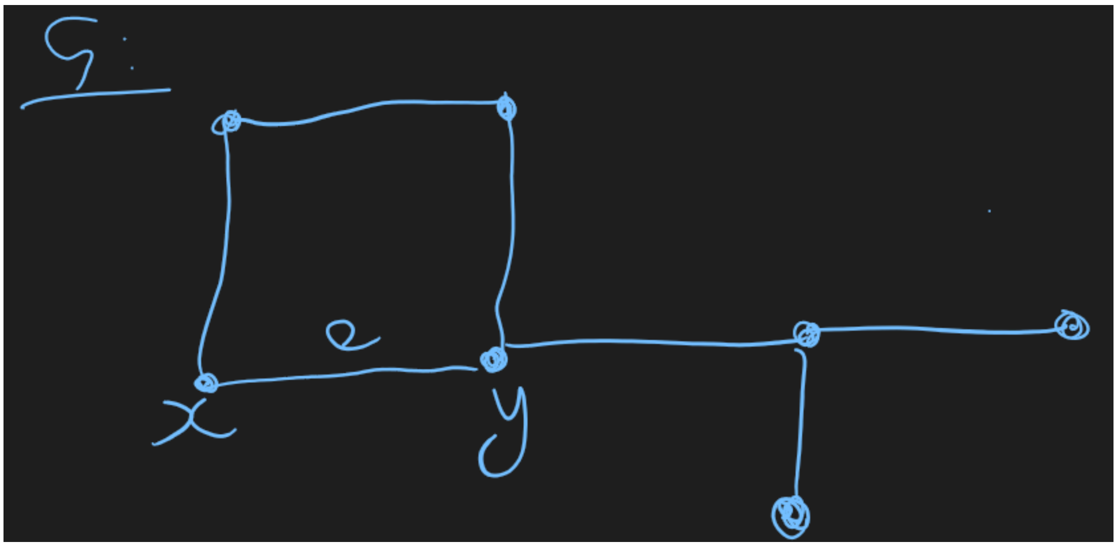


then $V(G) = 7$ and $E(G) = 6$.

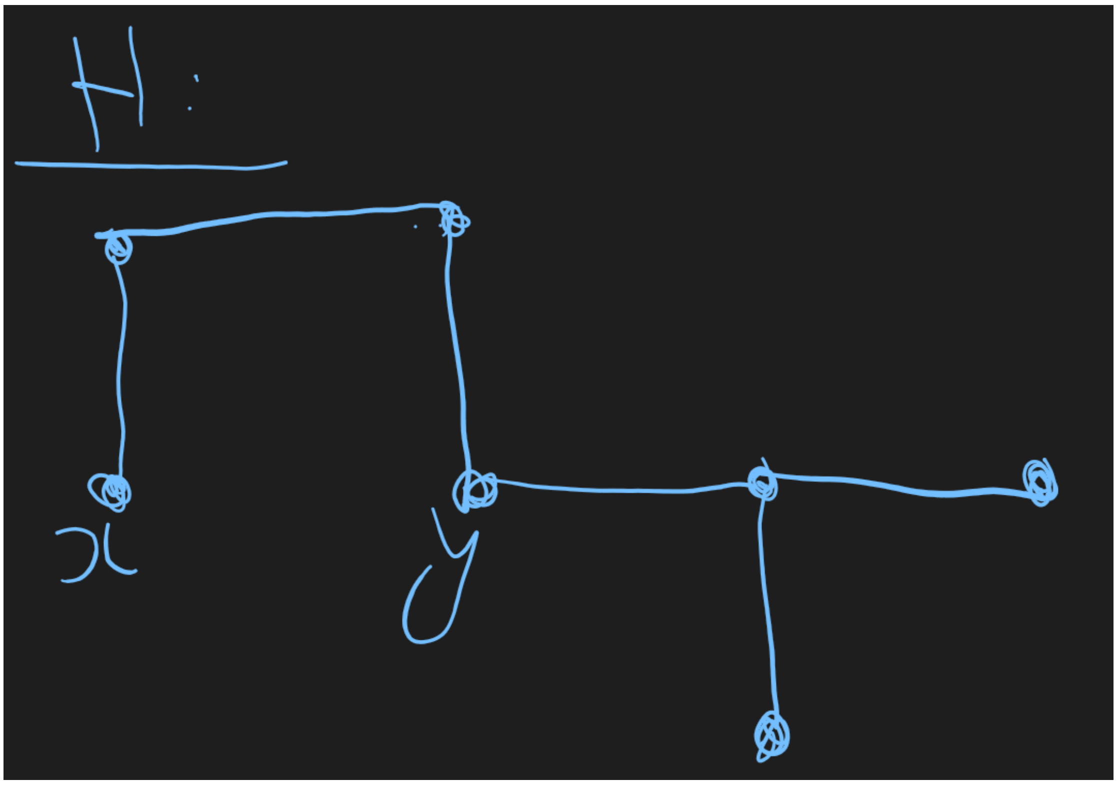
***Our proof proceeds by induction on E(G)***

- If $E(G) = 0$, then there can only be one vertex, so the property holds.
- So suppose that $E(G) > 0$

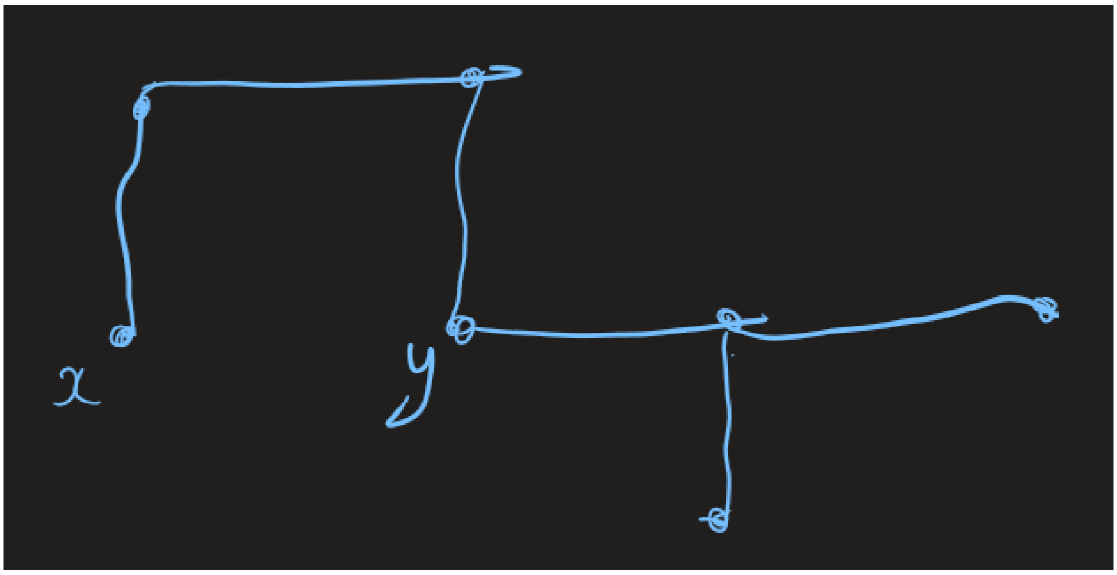
- Pick an edge *e* from *x* to *y*.



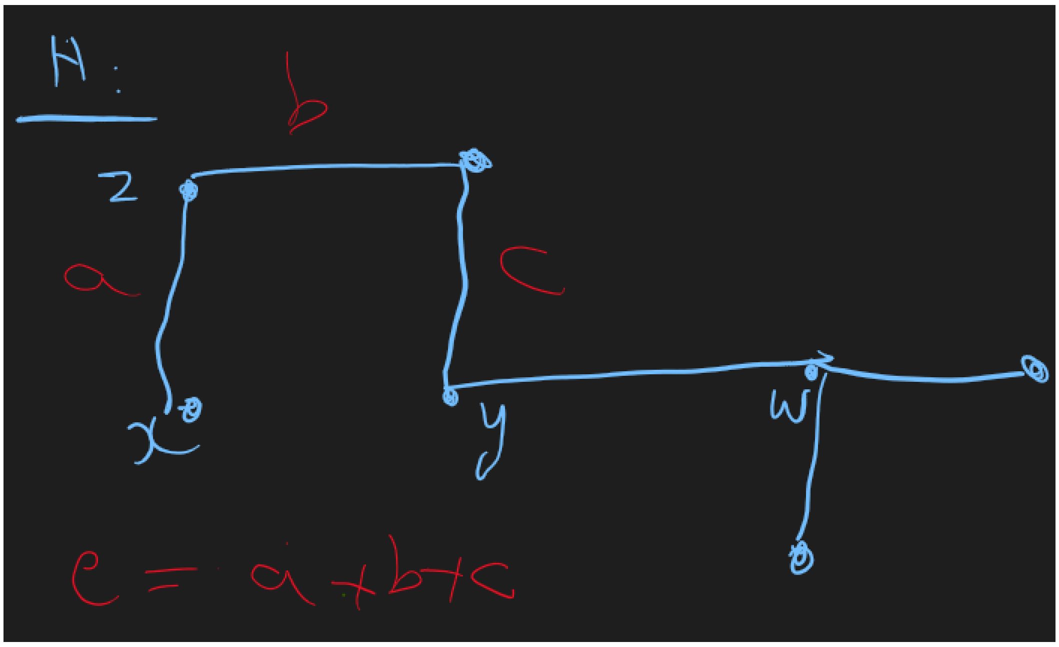
- Let *H* be *G* with the same vertices, but *e* removed.
- V(H) = V(G) but E(H) = E(G) - 1



Suppose there's a path *p* from *x* to *y* in *H left after removing e*



- Then *H* is connected.
- For any nodes *z* and *w*, take a path in *G* from *z* to *w*, then replace *e* in this path by *p*
    - e represented the entirety of the connection from x to y therefore:

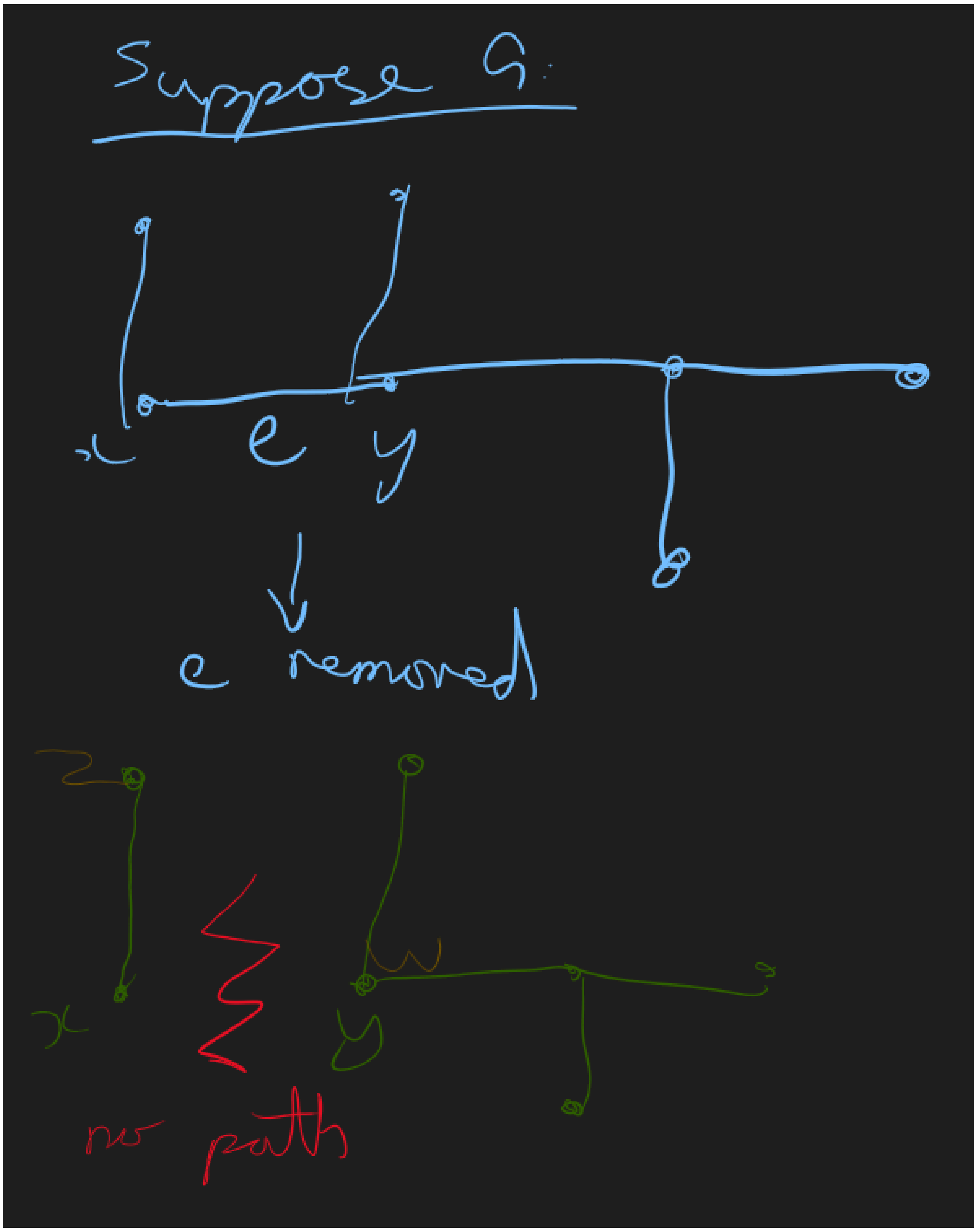


- The inductive hypothesis is that the statement $V(H) \leqslant E(H) + 1$ is true for smaller graphs. Since H has one edge less than G /Since *E*(*H*) = *E*(*G*) −1,
    - We apply the inductive hypothesis {Which only holds for connected graphs} to *H* giving $V(H) \leqslant E(H) + 1$

- So:

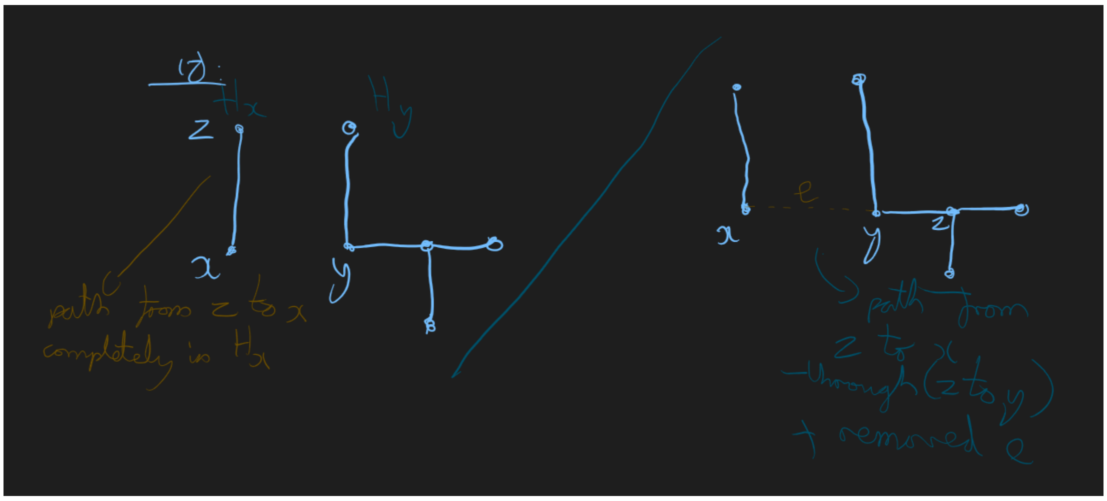
$$V(G) = V(H) \leqslant E(H) + 1 = E(G) < E(G) + 1$$

On the other hand, suppose there's no path from *x* to *y* in *H*.



- Then *H* consists of two seperate connected components

- The part $H_X$ that's connected to *x*
- The part $H_Y$ that's connected to *y*
- For any vertex *z*, there's a path in *G* from *z* to *x* with no cycles.
- It either lies entirely in $H_X$, in which case $z \in H_X$, or consists of a path in *H* from *z* to *y* followed by the edge *e* in which case $z \in H_Y$



(i):
$H_x$
$H_y$
$z$
$x$
$y$
path from $z$ to $x$ completely in $H_x$
$e$
$x$
$y$
$z$
path from $z$ to $x$ through ($z$ to $y$) + removed $e$

We can't have both $z \in H_X$ and $z \in H_Y$, as this would give a path

from *x* to *y*.)

Since $H_X$ and $H_Y$ are each connected graphs and have each have fewer edges than *G*, we can apply the inductive hypothesis to them, giving $V(H_X) \leqslant E(H_X) + 1$, and $V(H_Y) \leqslant E(H_Y) + 1$.

Thus

$$V(G) = V(H_X) + V(H_Y) \leqslant E(H_X) + E(H_Y) + 2 = E(H) + 2 = E(G) + 1$$

as required.

Notice that, in this example, we don't quote the inductive hypothesis at the start of the inductive step. Instead, we put some work into obtaining two smaller graphs, and only then do we apply the inductive hypothesis to each of them.