4. Inspired by the example of that great Cornellian, Vladimir Nabokov, some of your friends have become amateur lepidopterists (they study butterflies). Often when they return from a trip with specimens of butterflies, it is very difficult for them to tell how many distinct species they've caught—thanks to the fact that many species look very similar to one another.

One day they return with n butterflies, and they believe that each belongs to one of two different species, which we'll call A and B for purposes of this discussion. They'd like to divide the n specimens into two groups—those that belong to A and those that belong to B—but it's very hard for them to directly label any one specimen. So they decide to adopt the following approach.

For each pair of specimens i and j, they study them carefully side by side. If they're confident enough in their judgment, then they label the pair (i, j) either "same" (meaning they believe them both to come from the same species) or "different" (meaning they believe them to come from different species). They also have the option of rendering no judgment on a given pair, in which case we'll call the pair *ambiguous*.

So now they have the collection of n specimens, as well as a collection of m judgments (either "same" or "different") for the pairs that were not declared to be ambiguous. They'd like to know if this data is consistent with the idea that each butterfly is from one of species A or B. So more concretely, we'll declare the m judgments to be *consistent* if it is possible to label each specimen either A or B in such a way that for each pair (i, j) labeled "same," it is the case that i and j have the same label; and for each pair (i, j) labeled "different," it is the case that i and j have different labels. They're in the middle of tediously working out whether their judgments are consistent, when one of them realizes that you probably have an algorithm that would answer this question right away.

Give an algorithm with running time $O(m + n)$ that determines whether the m judgments are consistent.

- Again Beautiful question

- We have n butterflies and they are labelled "same" ← (i,j)
  or
  "different" (i,j)

- Both butterflies either belong to class A
  OR
  both in class B

i → A
j → B OR

i → B
j → A

- We are asked to check consistency of bifurcation

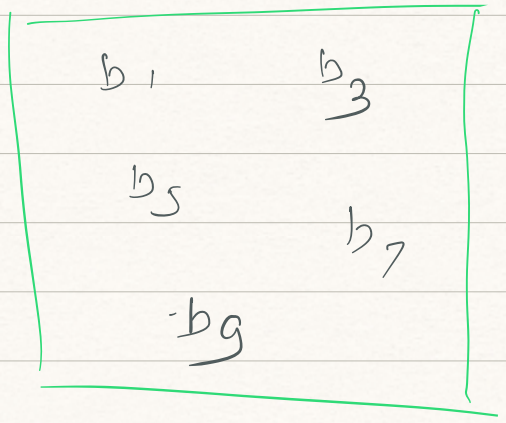
- As usual let's take example

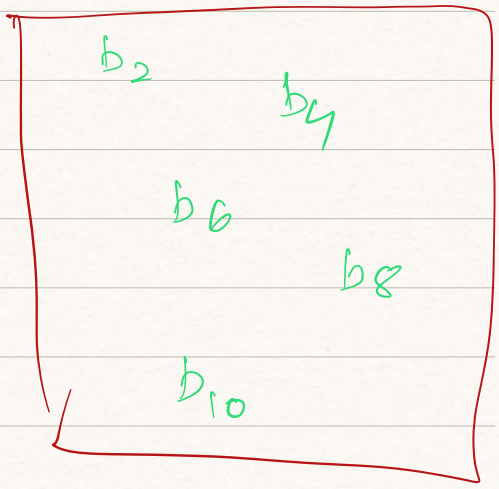
e.g.

- we have total $b_1, b_2, \ldots, b_{10}$
- odd butterflies in group A & rest in B

Secret :

- we are gonna use bipartite graph concept.
- Think how?



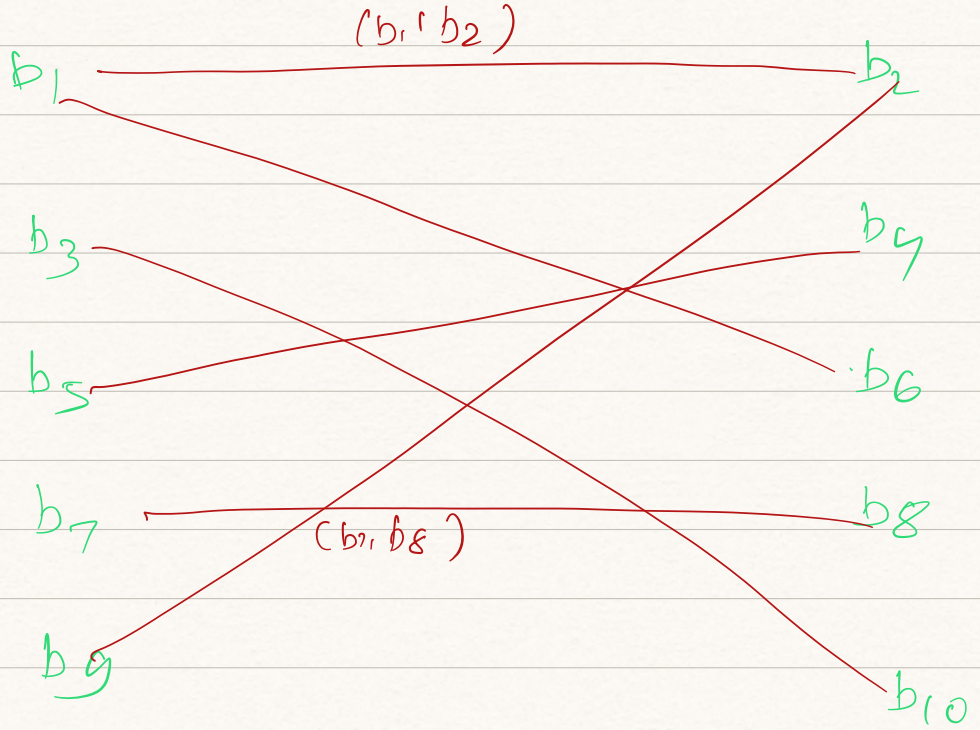


$(b_1, b_3) \rightarrow$ same
$(b_1, b_2) \rightarrow$ different
$(b_3, b_5) \rightarrow$ same
$(b_2, b_4) \rightarrow$ same
$(b_4, b_{10}) \rightarrow$ same

$(b_1, b_7) \rightarrow$ same
$(b_7, b_8) \rightarrow$ different

- Now, when we encounter
  - same (IGNORE)
  - different (e, (i, j))
    - draw an edge

At the end, if we get bipartite graph, we are good



- Now, let's discuss the algorithm.

- We have n butterflies and m samples. We can create a graph with n vertices and iterate through m edges.
- When sample says "same", we will ignore it temporarily & when we encounter "different"

  ⇓

  we will add edge to the graph.

- At the end, we should have bipartite graph & we can again verify by iterating through m edges.
  & Now, when we encounter same [(i,j)]
  ↳ we need to make sure that, it be i & j both have same tag.

- Time Complexity :

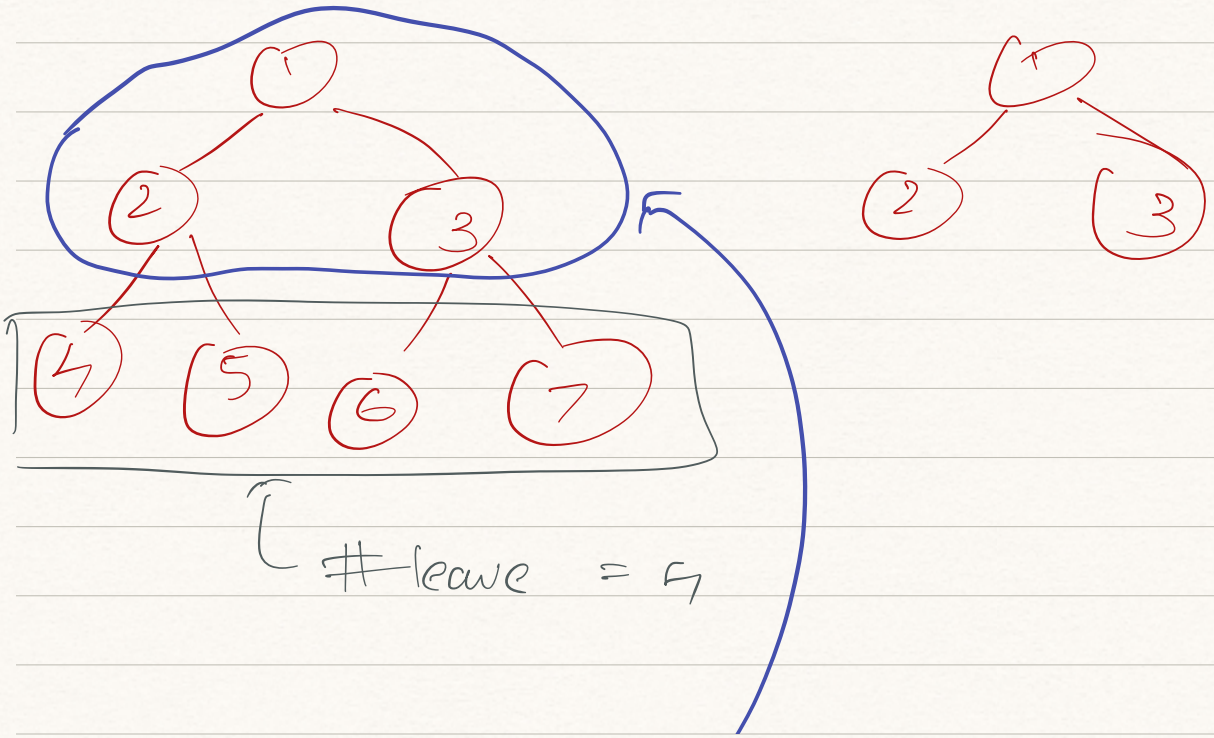
- $O(n+m)$ + $O(1) \cdot O(m)$

Bipartite Graph — verify for "same" label

- $O(n+m)$

5. A binary tree is a rooted tree in which each node has at most two children. Show by induction that in any binary tree the number of nodes with two children is exactly one less than the number of leaves.

- Let's understand the problem with example



/ Node with 2 childsem = 3

- When we have binary (complete) free tree, where each node has two children

total number of nodes (n)

$$n = 2^h - 1$$

- No. of leaves $= \frac{2^h}{2}$

$$= 2^{h-1}$$

- No. of nodes with 2 children

$$= \underbrace{(2^h - 1)}_{\text{total}} - \underbrace{(2^{h-1})}_{\text{leaf}}$$

$$= \underline{\underline{2^{h-1} - 1}}$$

$$\boxed{(2^{h-1} - 1) < 2^{h-1}}$$

this proof can be verified even without Assumption

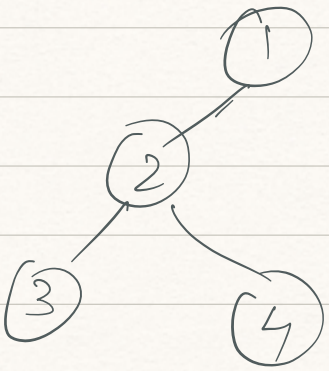
all nodes have 2 node

- Let's poove this with induction

i) tree with 1 root & 2 leaves
(with 2 children)

root < # leaves

ii) tree with 2 non-leaf & 2 leaf



node < # leaves
(with 2 children)

- Every node that we add, if we can have 2 children i.e. leaves of that node then everytime we add m nodes & it generates

$m+1$ leaves.

We can say -

6. We have a connected graph $G = (V, E)$, and a specific vertex $u \in V$. Suppose we compute a depth-first search tree rooted at u, and obtain a tree T that includes all nodes of G. Suppose we then compute a breadth-first search tree rooted at u, and obtain the same tree T. Prove that $G = T$. (In other words, if T is both a depth-first search tree and a breadth-first search tree rooted at u, then G cannot contain any edges that do not belong to T.)