Dis 2B: Graphs

Tuesday, 30 June 2020 5:48 PM

From previous section:

3 Hello World!

Determine the computability of the following tasks. If it's not computable, write a reduction or self-reference proof. If it is, write the program.

(a) You want to determine whether a program P on input x prints "Hello World!". Is there a computer program that can perform this task? Justify your answer.

(b) You want to determine whether a program P prints "Hello World!" before running the kth line in the program. Is there a computer program that can perform this task? Justify your answer.

(c) You want to determine whether a program P prints "Hello World!" in the first k steps of its execution. Is there a computer program that can perform this task? Justify your answer.

Problem: Halting Problem cannot be computed

suppose HP is solvable, then ∃ fn ~~TestH(P, x)~~ Y if P(x) N if P(x)

TH couldn't exist by Turing (Turing) argument.

Goal: a fn that checks if an ~~into~~ int is divisible by 4 exist:

TH

suppose. this doesn't exist

divisible_4 (x)

P HW

divisible_by_ (x)

def div_4(x)
if div_8 (x) is True
return T
else:
return F

(a) Uncomputable. We will reduce TestHalt to PrintsHW(P,x).

```
TestHalt(P, x):
  P'(x):
    run P(x) while suppressing print statements
    print("Hello World!")
  if PrintsHW(P', x):
    return true
  else:
    return false
```

If PrintsHW exists, TestHalt must also exist by this reduction. Since TestHalt cannot exist, PrintsHW cannot exist.

writing testHalt (so it seems like TH can solve HP)

← problem

HP

& TestHalt ∃ fn

Solver

Definitions

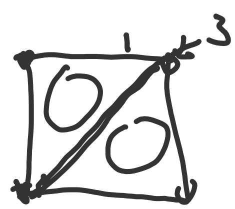
Tree = connected, acyclic graph
Connected graph G = (V, E) = for any 2 points u, v of G, there's a path from u to v

1 True or False

(a) Any pair of vertices in a tree are connected by exactly one path.

(b) Adding an edge between two vertices of a tree creates a new cycle.

(c) Adding an edge in a connected graph creates exactly one new cycle.



G undirected,
$\forall v \in V(G)$ have even degree
$\Updownarrow$
G has E-tour

2 Bipartite Graph

A bipartite graph consists of 2 disjoint sets of vertices (say L and R), such that no 2 vertices in the same set have an edge between them. For example, here is a bipartite graph (with $L = \{$green vertices$\}$ and $R = \{$red vertices$\}$), and a non-bipartite graph.

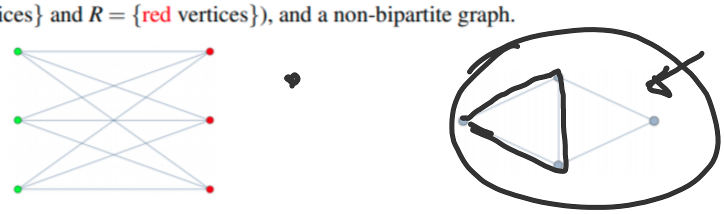


Figure 1: A bipartite graph (left) and a non-bipartite graph (right).

Prove that a graph has no tours of odd length if it is a bipartite (This is equivalent to proving that, a graph G being a bipartite implies that G has no tours of odd length).

Contrapositive

L: G bip $\Rightarrow$ R: G has no tours of odd len

$\neg R \Rightarrow \neg L$

$\exists 1$ tour of odd length, call one of them t

eg start from green (taking the tour t)

G → R → G →

1) L
2 case: i) G has a tour
ii) G has no tour
$\Rightarrow$ ✓

Direct

∴ it's bipartite
if you have a -
it need to end
the same "side"

$2n+1$th vertex is (odd) 'otherwi

Fact a)

A+B = even
and B is even $\Rightarrow$ A is even

4 Odd Degree Vertices

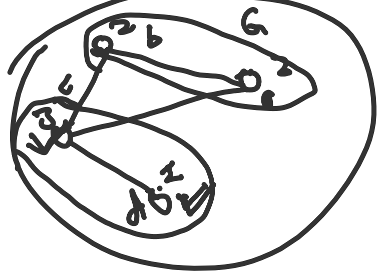
Claim: Let $G = (V, E)$ be an undirected graph. The number of vertices of G that have odd degree is even.

Prove the claim above using:

(i) Direct proof (e.g., counting the number of edges in G). *Hint: in lecture, we proved that* $\sum_{v \in V} \deg v = 2|E|$.

(ii) Induction on $m = |E|$ (number of edges)

(iii) Induction on $n = |V|$ (number of vertices)



1+3+2+2

=

4 + 4

$\sum_{v \in V} \deg$ is even

even

even

$\sum_{v \in V_{odd}(v)} \deg(v) + \sum_{v \in V_{even}(v)} \deg(v)$

even

even

must have even $|V_{odd}(v)|$

$G = (V, E)$

$V_{odd}(G)$: set of with odd deg vertices in G

$|V_{odd}(G)|$

$V = V_{odd}(G) \cup V_{even}(G)$

sum up the degrees of vertices of both sets

even $= \boxed{} +$ $\sum_{v \in V_{even}(G)} \deg(v)$ is even

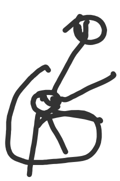
even

base: $m = 1$

o——o $\Rightarrow$ 2 ✓✓

Ind. S: consider when $m = 2k+1$, $G = (V, E)$

IH: $\boxed{\text{for all } G \text{ w/ } m = 2k, \ |V_{odd}(v)| \text{ is even}}$ ✓



take random $e \in E$, & remove rm. from E

$\Rightarrow \boxed{G_k = (V, E / \{e\})}$.

a) when you re-add (u,v) to G_k (& get back G)

look @ effect of "re-adding" (u,v) to G_k (G)

$\{u, v\}$ $m = 2k$ edge $\Rightarrow$ IH

u v

b)

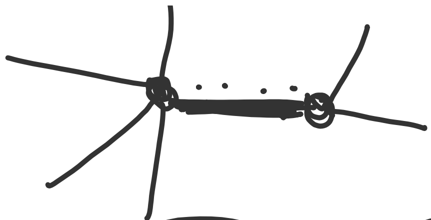
③ case: a) both u, v had even deg in G_k

b) one have even deg " "

c) none "

$G_k = (V, E / \{e = (u,v)\})$

1



(u,v) back?

G_k

WLOG
Suppose u had even deg:

deg u = 4 +1
even

deg v = 2 +1
even

v had odd deg

+1 to $|V_{odd}(G)|$
-1 to $|V_{odd}(G)|$

overall: $\frac{|V_{odd}(G)|}{\pm 2}$

overall: ± 0

Definitions #2

Walk: sequence of edges (edge repeats allowed)

Tour: Walk (edge repeats disallowed) + start and end at same vertex

Eulerian Tour: Tour + all edges visited once

Eulerian Walk: like Eulerian Tour, but don't have to start / end at same place

(Simple) Cycle:

c) e=(u,v) ie both had

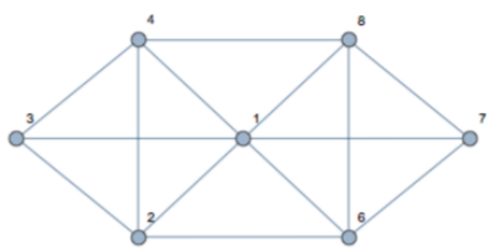
in $G_k = (V, E$

(u,v) u has even
v " even

$\Rightarrow$ -2 odd

$|V_{odd}(G)| =$

3 Eulerian Tour and Eulerian Walk



(a) Is there an Eulerian tour in the graph above? If no, give justification. If yes, provide an example.

(b) Is there an Eulerian walk in the graph above? An Eulerian walk is a walk that uses each edge exactly once. If no, give justification. If yes, provide an example.