Discussion 2a

Yunzhi Zhang

February 8, 2019

Path, Cycle, Tour

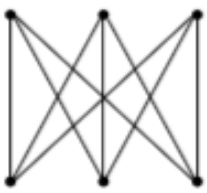
- Path: a sequence of edges $\{(v_1, v_2), (v_2, v_3), \ldots, (v_{n-1}, v_n)\}$, where $v_1, \ldots, v_n$ are distinct.
- Cycle: a path such that $v_n = v_1$.
- Walk: a sequence of edges $\{(v_1, v_2), (v_2, v_3), \ldots, (v_{n-1}, v_n)\}$, where $v_1, \ldots, v_n$ are not necessarily distinct.
- Tour: a walk such that $v_n = v_1$, with no repeated edges.

Eulerian

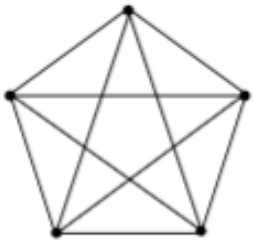
- Connected Graph: a graph is connected if there is a path between any two distinct vertices.
- Eulerian walk: a walk that uses each edge exactly once.
- Eulerian tour: an Eulerian walk that starts and ends at the same vertex.

Type of Graphs

- Planar Graph: a graph is planar if it CAN (but not necessarily always) be drawn in the plane without crossings.

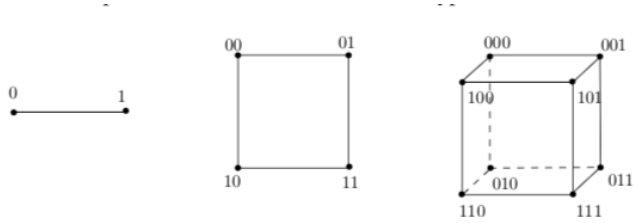


(v)



(vi)

  Important examples: $K_{3,3}, K_5$.
- Complete Graph: any distinct pair of vertices are connected by an edge.
- Trees: the following definitions are EQUIVALENT and handy:
  1. connected, no cycles
  2. connected, $|V| - 1$ edges
  3. connected, removal of any single edge disconnects the graph
  4. no cycles, addition of any single edge creates a cycle



- Hypercube: n vertices, $n2^{n-1}$ edges.

Bipartite Graph (P3)

Prove that a graph is bipartite if and only if it has no tours of odd length.

The followings are two possible procedures to find a bipartition in a graph with no odd tours. By explicitly describing a way to find a bipartition such a graph, we proved the "if" direction.

```
ROUTINE BiPartite1(G = (V, E)):
    # As on official solution.

    repeat:
        pick an uncolored vertex v from V
        color v with green
        for all uncolored u in V such that u, v is connected by some path:
            if some path from u, v is of even length:
                color u with green
            else:
                color u with red
    until all vertices have been colored
```

```
ROUTINE BiPartite2(G = (V, E)):
    # Credit to the brilliant students who introduced this solution to me on section!

    SUBROUTINE Traverse(v):
        for all u in V such that u is adjacent to v and u is not colored:
            if u has not been colored yet:
                assign to u with the opposite color of v
                Traverse(u)

    repeat:
        pick an uncolored vertex v from V
        color v with green
        Traverse(v)
    until all vertices have been colored
```

Proof of Correctness:
If when we call Traverse(v), we see a neighbor of v, namely u, such that u has been assigned the same color as v, then we note that there is a path u-u_1-u_2-u_3-$\cdots$-u_n-v, where the path length is $n+1$ and n

must be odd by observing the pattern that only odd-indexed u_i is colored with red.
Also note that v is a neighbor of u, so there is a tour with length $n + 2$ which is odd. But this is not possible given the setting of the graph, so BiPartite2 always gives a valid bipartition.

A side note is that if we modify either BiPartite1 or BiPartite2 a little bit, we obtain a procedure that tells us whether the input graph G is bipartite. These two procedures are essentially the same idea, with different ways of describing it. The subroutine "Traverse" in BiPartite2 is indeed deep first search which will be covered soon. You are welcome to talk to me if interested in further explanation!