Jilin Zheng // U49258796

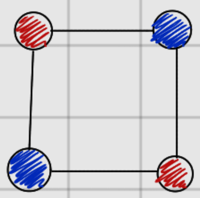
EC330 Homework #7

1. [Graphs 15 Points]
For which integer n > 0 values do the following graphs have the given chromatic number? Explain.

a. C_{2n}, chromatic number 2.
b. W_{2n}, chromatic number 3.
c. S_{2n} chromatic number 2.

a. C_{2n}, chromatic number 2

C_4

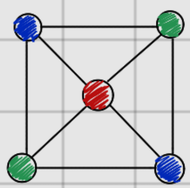


$n=2$

all even integers $n>0$ since alternating colors is legal

b. W_{2n}, chromatic number 3

W_5

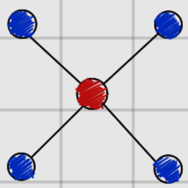


$n=\frac{5}{2}$

no integer values of n work, but fractions of $\frac{n}{2}$, with an odd n, satisfy the chromatic number

c. S_{2n}, chromatic number 2

S_4



all integers $n>0$ since only one color is needed for the leaves and only one other color is needed for the internal, center node

2. [Graph representation, 15 Points]
Consider directed graphs with V vertices and E edges.
For each of the following graph representations, provide:
- How long would it take (asymptotically, in terms of the number of vertices V and edges E) to compute both the <u>in-degree and out-degree of every vertex</u> in the graph. Explain.
- How much additional space the above would require. Explain.

a. Edge-list implemented as a linked list.
b. Adjacency-list implemented as a linked list of linked lists.
Include, for each vertex v, a list of vertices connected by an outgoing edge from v.
c. Incidence matrix implemented as a two-dimensional array. The incidence matrix of a directed graph is a $n \times m$ matrix B where n and m are the number of vertices and edges respectively, such that $B_{i,j} = -1$ if the edge e_j leaves vertex v_i, 1 if it enters vertex v_i and 0 otherwise.

a. Edge-list as linked list.

- Time to compute in- & out-degree: $\Theta(E)$ since we have to traverse the entire linked list
- Space required: $\Theta(V)$ since we need to track in- & out-degree of each vertex

b. Adjacency-list as a linked list of linked lists.

(V) (E)
- Time to compute in- & out-degree: $\Theta(V+E)$ since we have to traverse every vertex's adjacency list
- Space required: $\Theta(V)$ since we need to track in- & out-degree of each vertex

c. Incidence matrix as 2-D array.

- Time to compute in- & out-degree: $\Theta(VE)$ for both in- & out-degree since we have to traverse the matrix
- Space required: $\Theta(V)$ since we need to track in- & out-degree of each vertex

3. [Graph algorithms, 10 Points]
Give a linear-time algorithm for determining whether a given graph is bipartite.

Apply graph coloring. We must maintain a chromatic color of 2 for the graph to be bipartite.
Begin with any arbitrary vertex and color it one color, e.g. red.
Perform a Breadth-First Search, coloring adjacent vertices a second color, e.g. blue.
If we run into an already-colored vertex with the same color as the current color, the graph is NOT bipartite.
If we traverse the entire graph without problems, i.e. adjacent vertices have alternating colors, we have a bipartite graph.
The runtime is $\Theta(V+E)$, linear, as we visit every vertex & edge once in the worst case.

4. [Graph algorithms, 10 Points]
Design an efficient algorithm that computes a path in a given connected, undirected graph G that traverses every edge in G exactly once in each direction. Analyze its runtime.

Apply Depth-First Search, as covered in lecture.

DFS(G)

Color all vertices white $\Theta(V)$
$\pi(v)$=null $\Theta(V)$
for each white vertex v $\Theta(V)$
    DFS-Visit(G,V)

DFS-Visit(G,V)

Color V grey $\Theta(V)$
for each node u adjacent to V $\Theta(E)$ (adjacency list)
    if u is white $\Theta(V)$
        $\pi(u)=V$ $\Theta(V)$
        DFS-Visit(G,u) $\Theta(V)$
Color V black $\Theta(V)$

Aggregate Analysis: $\Theta(V+E)$