Homework 4 (50 points) Due: October 04, 2024 11:59 pm
COMPSCI 733: Advanced Algorithms and Designs
Benzon Carlitos Salazar (salazarbc24@uww.edu)

Documentation: (5 points) Type your solutions using Latex (www.overleaf.com or https://www.latex-project.org/). Submit your solutions (pdf is enough) to Canvas.

Problem 1: (25 points)

(a) Let $G = (V, E)$ be an undirected simple graph with finite number of vertices. Prove that

$$2|E| = \sum_{v \in V} deg(v),$$

where $deg(v)$ is the number of edges incident with the vertex v.

Proof. Let $G = (V, E)$ be an undirected simple graph, where V is the set of vertices and E is the set of edges. For each vertex $v \in V$, let $\deg(v)$ denote the degree of vertex v, which is the number of edges incident to v.

Now, consider the sum of the degrees of all vertices in the graph:

$$\sum_{v \in V} \deg(v)$$

Each edge in the graph contributes exactly two to the total sum of degrees, since every edge is incident to exactly two vertices in an undirected graph. That is, for each edge $e = (u, v) \in E$, the vertices u and v both have their degrees increased by 1 due to the edge e.

Thus, the sum of the degrees of all the vertices is twice the number of edges in the graph:

$$\sum_{v \in V} \deg(v) = 2|E|.$$

□

(b) Let $G = (V, E)$ be an undirected simple connected graph. Prove that the BFS algorithms discussed in the class runs in time $O(|V| + |E|)$, if the graph is given by the adjacency list representation. (Hint: Use the Part (a) result.)

Proof. BFS starts at a source vertex and explores all vertices and edges in the graph. The graph is represented using an adjacency list, where for each vertex v, we store a list of its adjacent vertices. We now analyze the running time of BFS:

1. **Initialization:** BFS initializes data structures such as the queue, distances, and predecessors. This takes $O(|V|)$ time because we have to initialize each vertex individually.

2. **Exploring Vertices:** Each vertex is enqueued and dequeued exactly once during the BFS traversal. Since there are $|V|$ vertices, this step takes $O(|V|)$ time.
3. **Exploring Edges:** For each vertex $v \in V$, BFS iterates over all of its neighbors, i.e., all vertices adjacent to v. The time to explore the neighbors of a vertex v is proportional to the degree of v, denoted $\deg(v)$. Therefore, the total time to explore all neighbors of all vertices is

$$O\left(\sum_{v \in V} \deg(v)\right).$$

By the result from Part (a), we know that

$$\sum_{v \in V} \deg(v) = 2|E|.$$

Thus, the time spent exploring edges is $O(2|E|) = O(|E|)$.

4. **Total Time Complexity:** The total time is the sum of the time spent on initialization, vertex exploration, and edge exploration. Hence, the overall time complexity of BFS is

$$O(|V| + |E|).$$

□

(c) What is the complexity for Part (b) if an adjacency matrix is used?

Proof. When the graph is represented using an adjacency matrix, the BFS algorithm works as follows:

1. **Initialization:** Initializing the BFS queue and distance array takes $O(|V|)$, since there are $|V|$ vertices.
2. **Exploring Vertices:** Each vertex is enqueued and dequeued exactly once during BFS, which takes $O(|V|)$ time.
3. **Exploring Edges:** For each vertex $v \in V$, BFS must check whether there is an edge between v and each other vertex $u \in V$. Since the graph is represented by an adjacency matrix, this requires scanning through the entire row of the matrix corresponding to v, which takes $O(|V|)$ time for each vertex. Therefore, the total time for edge exploration is

$$O(|V|) \times |V| = O(|V|^2).$$

4. **Total Time Complexity:** The overall time complexity is the sum of the time spent on initialization, vertex exploration, and edge exploration, which gives

$$O(|V| + |V| + |V|^2) = O(|V|^2).$$

□

(d) A sequence of integers $d_1, d_2, \ldots, d_n$ is called graphic if it is the degree sequence of a simple undirected graph. Using basic graph properties including Problem 1a, determine whether each of these sequences is graphic. For those that are, draw a graph having the given sequence. For those that are not, provide reasons for why no graph has such a sequence.

(1) 5, 4, 3, 2, 1, 0
(2) 6, 5, 4, 3, 2, 1
(3) 2, 2, 2, 2, 2, 2
(4) 3, 3, 3, 2, 2, 2
(5) 3, 3, 2, 2, 2, 2
(6) 1, 1, 1, 1, 1, 1
(7) 5, 3, 3, 3, 3, 3
(8) 5, 5, 4, 3, 2, 1

Answer:

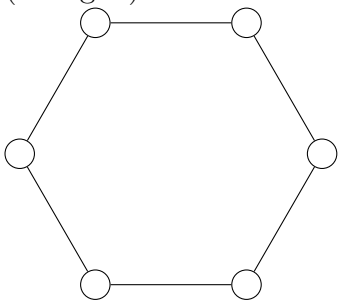
(1) 5, 4, 3, 2, 1, 0
Sum of degrees: $5 + 4 + 3 + 2 + 1 + 0 = 15$ (odd number).
Conclusion: This sequence is **not graphic**, because the sum of the degrees is odd, violating the Handshaking Lemma.

(2) 6, 5, 4, 3, 2, 1
Sum of degrees: $6 + 5 + 4 + 3 + 2 + 1 = 21$ (odd number).
Conclusion: This sequence is **not graphic**, because the sum of the degrees is odd, violating the Handshaking Lemma.

(3) 2, 2, 2, 2, 2, 2
Sum of degrees: $2 + 2 + 2 + 2 + 2 + 2 = 12$ (even number).
Havel-Hakimi check:

* $2, 2, 2, 2, 2, 2$
* Remove a 2, subtract 1 from the next two: $1, 1, 2, 2, 2$
* Remove a 2, subtract 1 from the next two: $1, 1, 1, 1$
* Remove a 1, subtract 1 from the next one: $0, 0, 0$.

Conclusion: This sequence is **graphic**. The corresponding graph is a 6-cycle (hexagon):

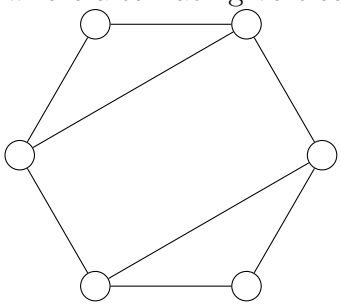


(4) 3, 3, 3, 2, 2, 2
Sum of degrees: $3 + 3 + 3 + 2 + 2 + 2 = 15$ (odd number).
Conclusion: This sequence is **not graphic**, because the sum of the degrees is odd, violating the Handshaking Lemma.

(5) 3, 3, 2, 2, 2, 2
Sum of degrees: $3+3+2+2+2+2=14$ (even number).
Havel-Hakimi check:

* $3,3,2,2,2,2$
* Remove a 3, subtract 1 from the next three: $2,2,1,1,2$
* Remove a 2, subtract 1 from the next two: $1,1,1,1$
* Remove a 1, subtract 1 from the next one: $0,0,0$.

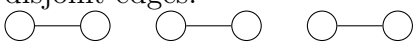
Conclusion: This sequence is **graphic**. One possible graph is a 6-vertex cycle where alternating vertices have degree 3 and 2.



(6) 1, 1, 1, 1, 1, 1
Sum of degrees: $1+1+1+1+1+1=6$ (even number).
Havel-Hakimi check:

* $1,1,1,1,1,1$
* Remove a 1, subtract 1 from the next one: $0,0,0,0,0$.

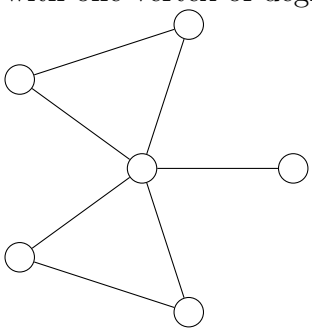
Conclusion: This sequence is **graphic**. The corresponding graph consists of 3 disjoint edges:



(7) 5, 3, 3, 3, 3, 3
Sum of degrees: $5+3+3+3+3+3=20$ (even number).
Havel-Hakimi check:

* $5,3,3,3,3,3$
* Remove a 5, subtract 1 from the next five: $2,2,2,2,2$
* Remove a 2, subtract 1 from the next two: $1,1,1,1$
* Remove a 1, subtract 1 from the next one: $0,0,0$.

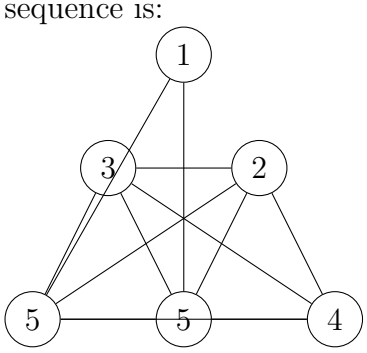
Conclusion: This sequence is **graphic**. One possible graph is a star graph with one vertex of degree 5:



(8) 5, 5, 4, 3, 2, 1
Sum of degrees: $5+5+4+3+2+1=20$ (even number).
Havel-Hakimi check:

* $5,5,4,3,2,1$
* Remove a 5, subtract 1 from the next five: $4,3,2,1,0$
* Remove a 4, subtract 1 from the next four: $2,1,0,0$
* Remove a 2, subtract 1 from the next two: $0,0,0$.

Conclusion: This sequence is **graphic**. A possible graph satisfying the degree sequence is:



Problem 2: (10 points)

A graph $G=(V,E)$, $|V|=n$, is bipartite if its vertices can be partitioned into two subsets $V=A\cup B$ such that all edges connect only vertices between the two sets (no two edges in the same set are connected).

Is the given statement True or False? If it is true, give a proof. If it is false, give a counter example.

(a) If a graph G is bipartite, then G is a tree.
Answer: False. Not all bipartite graphs are trees. A counterexample is a 4-vertex cycle graph.

(b) If a graph G is a tree, then it is bipartite.
Answer: True. Every tree is bipartite.

Proof. Let $G=(V,E)$ be a tree, which means G is a connected acyclic graph. We need to show that G is bipartite, i.e., that the vertex set V can be partitioned into two disjoint sets A and B such that every edge in E connects a vertex in A to a vertex in B, with no edge connecting two vertices within the same set.

1. Start by choosing an arbitrary vertex $v_0\in V$ and perform a breadth-first search (BFS) or depth-first search (DFS) traversal starting from v_0. During the traversal, we will color the vertices in two colors, say **color red** and **color blue**, which correspond to the two sets A and B we are trying to construct.
2. Assign vertex v_0 to set A (color red).
3. For each vertex adjacent to v_0, assign it to set B (color blue). More generally, for any vertex that is reached during the traversal:

- If the vertex is assigned to set A (color red), assign all of its neighbors to set B (color blue).
- If the vertex is assigned to set B (color blue), assign all of its neighbors to set A (color red).

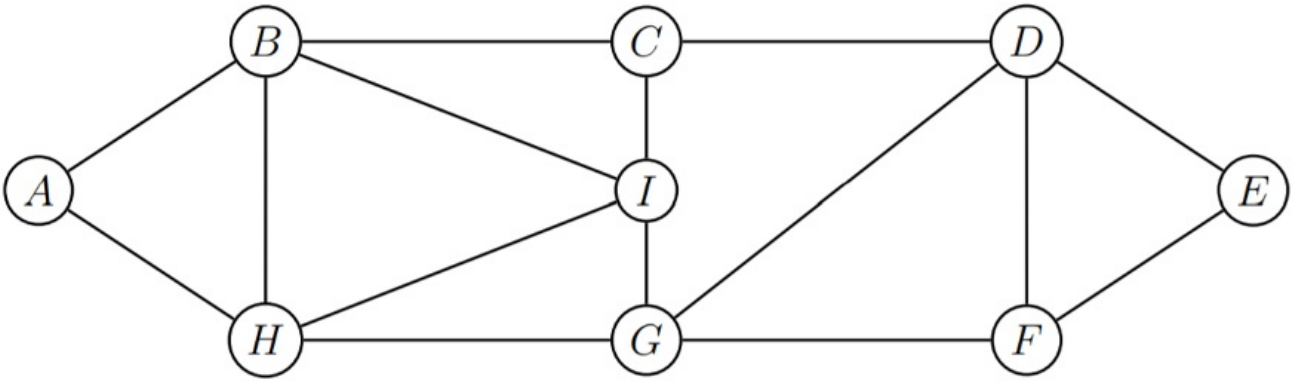
4. Continue this process for all vertices encountered during the traversal. Since G is a tree, it has no cycles, and this coloring scheme will not lead to any conflicts. Specifically, no two adjacent vertices will be assigned to the same set (i.e., both in A or both in B) because:
   - Every time a vertex is colored, all of its neighbors are assigned the opposite color.
5. Since the graph is acyclic, we will not encounter any conflicts or contradictions during the traversal. In other words, there is no point in the process where an edge would connect two vertices in the same set.

After coloring all the vertices of G, we have successfully partitioned the vertex set V into two sets A and B such that every edge connects a vertex in A to a vertex in B. Thus, the graph is bipartite. **Therefore,** every tree is bipartite. □

Problem 3: (10 points) Step through the following BFS algorithm (CLRS textbook) for the following given graph with the starting vertex "A" and fill in the entries in the given table as shown in the class lecture. You need to show any intermediate values too.

Algorithm 1 BFS(G, s)

```
1: for each vertex u ∈ G.V − {s} do
2:     u.color ← WHITE
3:     u.d ← ∞
4:     u.π ← NIL
5: end for
6: s.color ← GRAY
7: s.d ← 0
8: s.π ← NIL
9: Q ← ∅
10: ENQUEUE(Q, s)
11: while Q ≠ ∅ do
12:     u ← DEQUEUE(Q)
13:     for each v ∈ Adj[u] do
14:         if v.color == WHITE then
15:             v.color ← GRAY
16:             v.d ← u.d + 1
17:             v.π ← u
18:             ENQUEUE(Q, v)
19:         end if
20:     end for
21:     u.color ← BLACK
22: end while
```



Vertex v	Color $v.color$	Distance $v.d$	Predecessor $v.\pi$
A			
B			
C			
D			
E			
F			
G			
H			
I			

Answer:

Vertex	Color	Distance	Predecessor
A	GRAY	0	Nil
B	GRAY	1	A
C	WHITE	∞	Nil
D	WHITE	∞	Nil
E	WHITE	∞	Nil
F	WHITE	∞	Nil
G	WHITE	∞	Nil
H	GRAY	1	A
I	WHITE	∞	Nil

After processing B:

Vertex	Color	Distance	Predecessor
A	BLACK	0	Nil
B	BLACK	1	A
C	GRAY	2	B
D	WHITE	∞	Nil
E	WHITE	∞	Nil
F	WHITE	∞	Nil
G	WHITE	∞	Nil
H	GRAY	1	A
I	WHITE	∞	Nil

After processing H:

Vertex	Color	Distance	Predecessor
A	BLACK	0	Nil
B	BLACK	1	A
C	GRAY	2	B
D	WHITE	∞	Nil
E	WHITE	∞	Nil
F	WHITE	∞	Nil
G	GRAY	2	H
H	BLACK	1	A
I	GRAY	2	H

After processing C:

Vertex	Color	Distance	Predecessor
A	BLACK	0	Nil
B	BLACK	1	A
C	BLACK	2	B
D	WHITE	∞	Nil
E	WHITE	∞	Nil
F	WHITE	∞	Nil
G	GRAY	2	H
H	BLACK	1	A
I	GRAY	2	H

After processing G:

Vertex	Color	Distance	Predecessor
A	BLACK	0	Nil
B	BLACK	1	A
C	BLACK	2	B
D	WHITE	∞	Nil
E	WHITE	∞	Nil
F	GRAY	3	G
G	BLACK	2	H
H	BLACK	1	A
I	GRAY	2	H

After processing I:

Vertex	Color	Distance	Predecessor
A	BLACK	0	Nil
B	BLACK	1	A
C	BLACK	2	B
D	GRAY	3	I
E	WHITE	∞	Nil
F	GRAY	3	G
G	BLACK	2	H
H	BLACK	1	A
I	BLACK	2	H

After processing F:

Vertex	Color	Distance	Predecessor
A	BLACK	0	Nil
B	BLACK	1	A
C	BLACK	2	B
D	GRAY	3	I
E	GRAY	4	F
F	BLACK	3	G
G	BLACK	2	H
H	BLACK	1	A
I	BLACK	2	H

After processing D:

Vertex	Color	Distance	Predecessor
A	BLACK	0	Nil
B	BLACK	1	A
C	BLACK	2	B
D	BLACK	3	I
E	GRAY	4	F
F	BLACK	3	G
G	BLACK	2	H
H	BLACK	1	A
I	BLACK	2	H

After processing E:

Vertex	Color	Distance	Predecessor
A	BLACK	0	Nil
B	BLACK	1	A
C	BLACK	2	B
D	BLACK	3	I
E	BLACK	4	F
F	BLACK	3	G
G	BLACK	2	H
H	BLACK	1	A
I	BLACK	2	H