Algorithmique de base

Master 1, Université de Rennes

04/10/2024 – TD 5

Exercise 1. Let G be un undirected graph with adjacency matrix A.

1. Prove that there must exist two distinct vertices u, v such that $\deg(u) = \deg(v)$.
2. Prove that the degree of the i-th vertex is equal to $A^2_{i,i}$. What do the entries in the i, j position of A^2 represent in G?
3. Show that that $A^k_{i,j}$ is the number of walks starting at vertex i, ending at vertex j and having length k.

Exercise 2. 1. Show that every closed odd walk in a graph contains an odd cycle.

Let G be a graph. A set of pairwise adjacent vertices in G is called a *clique* and a set of pairwise non-adjacent vertices is called an *independent set* (or a *co-clique*). An undirected graph $G = (V, E)$ is called **bipartite** if the vertex set V can be partitioned into two independent subsets A, B.

2. Show that a graph is bipartite if and only if it has no odd cycles.
3. Prove that every graph on 6 vertices either has a clique or an independent set of size 3.

Exercise 3. Let Q_n be the hypercube graph, defined as the graph whose vertex set is $\{0, 1\}^n$ and two vertices are adjacent if and only if the corresponding binary vectors differ from each other in exactly one position (for example, $(1, 0, \ldots, 0)$ is adjacent to $(1, 1, 0, \ldots, 0)$).

2. Determine the number of edges in this graph.
3. What is the maximum size of a clique in Q_n?
4. Prove that Q_n is bipartite. Find the bipartition of Q_n.
5. Determine the diameter of Q_n (the largest distance between any pair of vertices).
6. Determine all values of n for which Q_n has an Eulerian circuit.

Exercise 4. 1. Let G be a directed graph (with a finite number of vertices). Show that if G has no vertex with outdegree zero, then it has a cycle.

2. Give an example of a directed graph with negative-weight edges for which Dijkstra's algorithm produces incorrect answers. Why doesn't the proof of correctness go through when negative-weight edges are allowed?

Exercise 5. 1. Show that we can use a depth-first search of an undirected graph G to identify the connected components of G, and that the depth-first forest contains as many trees as G has connected components.
