University of Waterloo
CS 341 Winter 2025
Written Assignment 2

Armin Jamshidpey, Mark Petrick, Collin Roberts

Due Date: Friday, February 7 at 11:59pm to Crowdmark
All work submitted must be the student's own.

- Make sure to read the Assignments section on the course webpage for instructions on submission and question expectations ("Instructions for Assignments"): `https://student.cs.uwaterloo.ca/~cs341/#Assignments`
- Space complexity is not required for this assignment.
- Assume all graphs are stored using adjacency list representation.

Question 1 [10 marks]

Jessica and Nolan are discussing whether it is possible to solve the Selection Problem in the worst case in the following way:

Jessica: I am sure we can solve the problem in $O(n)$ time in the worst-case using the algorithm quick-select. All we need to do is carefully choose the pivot. To do this, we can form groups of 3 entries of the input array A, then find medians of each of the $\frac{n}{3}$ groups in constant time. Once we get the $\frac{n}{3}$ medians, we find their medians recursively using quick-select. The median of medians will be used as our pivot. With this we can use the master method to show that the running time in the worst case is $O(n)$.

Nolan: No, I don't think so. Your idea is correct, and I agree with grouping into groups of size 3. But if we do this and write the runtime recursively, we get a recursive inequality for $T(n)$ which the master theorem cannot solve. When we have that recursive inequality for $T(n)$, if instead we use the substitution method, we can prove the cost will be $\omega(n^2)$.

Explain to Jessica and Nolan whether they are right or wrong. Your explanation must (i) include relevant definitions, formulas and justifications, and (ii) point out where Jessica's and Nolan' reasoning is correct/incorrect by proving the correct bounds.

Here are the definition of the problem and the pseudocode of quick-select. We are not concerned about n being divisible by 3 here.

Selection: given $A[0..n-1]$ and k in $\{0, \ldots, n-1\}$, find the entry that would be at index k if A was sorted.

```
quick-select(A, k)
A: array of size n,   k: integer s.t. 0 ≤ k < n
1.    p ← choose-pivot(A)
2.    i ← partition(A, p)                    i is the correct index of p
3.    if i = k then
4.        return A[i]
5.    else if i > k then
6.        return quick-select(A[0, 1, ..., i − 1], k)
7.    else if i < k then
8.        return quick-select(A[i + 1, i + 2, ..., n − 1], k − i − 1)
```

Question 2 [5 marks]

Given a graph where every edge has weight equal to 1 or 3, and a source vertex, design algorithm which finds the shortest path from the source vertex to every other vertex. Use BFS to give an efficient algorithm that has worst-case running time complexity $\Theta(|V|+|E|)$.

Question 3 [10 marks]

Given an undirected graph $G = (V, E)$, design a $O(|V| + |E|)$-time algorithm to output all cut edges of G.

An edge of an undirected graph is called a cut edge, if by removing it, the number of connected components of the graph increases.

A vertex of an undirected graph is called a cut vertex, if by removing it, the number of connected components of the graph increases.

Question 4 [10 marks]

Let G be an undirected graph with vertex set V and edge set E. The *degree* of vertex v is the number of edges incident to v. If U is a subset of V, then the graph *induced* by U is the graph with vertex set U, and with the edge set $\{(u, v) : u, v \in U, (u, v) \in E\}$, i.e., we throw away all the vertices outside U and all the edges incident to those vertices.

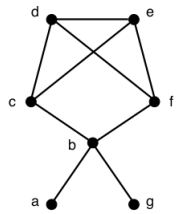
Given a graph G and a number k, a *degree-k-ordering* is an ordering $v_1, v_2, \ldots, v_n$ of the vertices V, such that for all $i \in \{1, \ldots, n\}$, v_i has degree at most k in the graph induced by $\{v_i, v_{i+1}, \ldots, v_n\}$.

Example 1. If G is a tree then it has a degree-k-ordering with $k = 1$ by repeatedly removing leaves from the tree.

Example 2. If G is a cycle then it has a degree-2-ordering.

1. [2 marks] Here is an example to show that it is not enough to sort the vertices by their degrees in the original graph. Consider the graph in the following figure. Ordering the vertices by their degrees in the whole graph gives the vertex ordering a, g, c, d, e, f, b because the corresponding degrees are $1, 1, 3, 3, 3, 3, 4$. This ordering is a degree-3-ordering because the degree of vertex c in the subgraph induced by vertices c, d, e, f, b is 3. More precisely, the degree of each vertex in the subgraph induced by removing the previous vertices is $1, 1, 3, 2, 1, 1, 0$.

   Find a degree-2-ordering for this graph.



2. [8 marks] Design a linear time algorithm that takes as input a graph G and a number k and finds a degree-k-ordering if one exists (otherwise the algorithm should declare that there is no such ordering).

   Note that linear time for a graph means $O(n + m)$ where n is the number of vertices and m is the number of edges. Justify correctness, and analyze the runtime.

Question 5 [10 marks]

Imagine in a city where all the streets are two-way streets. Perhaps because the streets are too narrow, the government would like to see if it is possible to make all streets one-way, so that there is still a way to drive from any intersection to any other intersection in the city. Show how to model this problem as a graph problem by providing a description of the problem in terms of a graph problem. Then design an algorithm to determine if it is possible, and when it is possible to output one solution to assign the directions. There exists an algorithm with runtime $O(n + m)$, where m is the number of streets in the city and n is the number of intersections in the city.

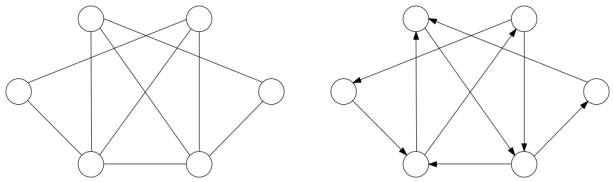


Figure 1: There is a solution in this example.

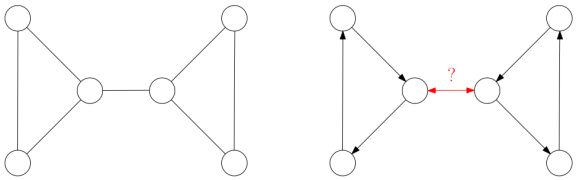


Figure 2: There is no solution in this example.