Assignment 5

Due by Monday, July 23, 11:59pm

Acknowledgments. Acknowledge all the sources you used to complete the assignment. DO NOT COPY! Please read `http://www.student.cs.uwaterloo.ca/~cs341/` for general instructions and policies.

For all algorithm design questions, you must give the algorithm, argue the correctness, and analyze time complexity.

1 NP completeness [10 marks]

Prove that the following problems are **NP**-complete.

(i) CliqueAndIS: Given a graph $G = (V, E)$ and a positive integer k, determine whether there is a clique of size at least k *and* an independent set of size at least k in G.

(ii) Subgraph: Given a graph $G = (V, E)$ and a graph $H = (V', E')$, determine if H is a subgraph of G—i.e., if there is a mapping π of the vertices in V' to the vertices in V such that for every $u, v \in V'$, $(\pi(u), \pi(v)) \in E$ if and only if $(u, v) \in E'$.

2 More NP completeness [10 marks]

Prove that the following problems are **NP**-complete.

(i) Clubs: Given

- a set of n persons that we associate with the numbers $\{1, 2, \ldots, n\}$,
- a collection $\mathcal{C}$ of clubs, where each club is a subset $C \subseteq \{1, 2, \ldots, n\}$ that represents the persons who are members of the club, and
- a positive integer k,

determine if there is a set $S \subseteq \{1, 2, \ldots, n\}$ of size $|S| \leq k$ such that every club contains at least one of the persons in S.

(ii) EvenSplit: Given n integers $a_1, \ldots, a_n$, determine whether there is a set $S \subseteq \{1, 2, \ldots, n\}$ for which

$$\sum_{i \in S} a_i = \sum_{i \in \{1,2,\ldots,n\} \setminus S} a_i.$$

3 And even more NP-completeness. . . or not? [10 marks]

In the CLIQUE3 problem, we are given a graph $G = (V, E)$ with maximum degree 3 and a positive integer k; we must determine if G has a clique of size at least k or not. (A graph G has *maximum degree* d if every vertex in G is incident to at most d edges.)

(i) Prove that CLIQUE3 $\in$ **NP**.

(ii) Here's a claimed proof that CLIQUE3 is **NP**-complete. Explain why the argument is incorrect.

> We showed in part (i) that CLIQUE3 is in **NP**. We know from lectures that CLIQUE is **NP**-complete. All that remains is to show that there is a polynomial-time reduction from CLIQUE3 to CLIQUE. Let F be the (trivial) algorithm that takes in a graph G with vertices of degree at most 3 and a parameter k, and leaves both as-is. The algorithm F runs in polynomial time and gives a transformation from inputs of the CLIQUE3 problem to inputs of the CLIQUE problem, and the answer to these inputs is always identical. Therefore, this is a valid polynomial-time reduction and CLIQUE3 is **NP**-complete.

(iii) In the VERTEXCOVER3 problem, we are given a graph $G = (V, E)$ with maximum degree 3 and a positive integer k; we must determine if G has a vertex cover of size at most k or not. It is known that VERTEXCOVER3 is **NP**-complete, and for this question we may use this fact without proof.

Here's another claimed proof that CLIQUE3 is **NP**-complete. Explain why the argument is incorrect.

> We already showed in part (i) that CLIQUE3 is in **NP**. We complete the proof that it is **NP**-complete by giving a polynomial-time reduction from VERTEXCOVER3 to CLIQUE3. Let F be the algorithm that transforms the input (G, k) into the input $(G, n - k)$. The algorithm F has polynomial-time complexity. And $C \subseteq V$ is a vertex cover in G if and only if $V \setminus C$ is a clique in G, so G has a vertex cover of size $\leq k$ if and only if it has a clique of size $\geq n - k$ and therefore our transformation gives polynomial-time reduction from VERTEXCOVER3 to CLIQUE3.

(iv) Prove that CLIQUE3 $\in$ **P**.

4 Almost acyclic graphs [10 marks]

In the AlmostDAG problem, we are given a directed graph $G = (V, E)$ and a positive integer k; we must determine if we it is possible to remove at most k edges from E to obtain a directed acyclic graph.

Prove that AlmostDAG is **NP**-complete.

Hint. You should consider using a reduction from VertexCover. See Piazza for a more detailed hint, if required.

5 Programming question [10 marks]

In the CONSTRAINEDAPSP problem, we are given a directed weighted graph $G = (V, E)$ with positive edge lengths $w : E \to \mathbb{R}^{>0}$ and a subset $S \subseteq V$ of vertices; for each pair of vertices $u, v \in V$, we must determine the length $L(u, v)$ of the shortest path from u to v *that visits at least one of the vertices in S along the path*. When no such path exists for a given pair u, v, the answer is ∞.

(i) Design and analyze an algorithm that solves the CONSTRAINEDAPSP problem. You should aim for an algorithm with time complexity $O(n^3)$ or better. Ideally, your algorithm will have time complexity $o(n^3)$ when $|S| = o(n)$. If that's the case, provide the time complexity analysis of the algorithm in terms of both $|S|$ and n.

(ii) Implement the algorithm you obtained in part (i).

Input and output. The input consists of several lines. The first line contains n (the number of vertices) and q (the number of queries). The set of vertices is $V = \{1, 2, \ldots, n\}$. The second line contains k distinct integers between 1 and n that specify the subset $S \subseteq V$.

The following n lines contain n integers each, with the jth entry of line i representing the length $w(i, j)$ of the edge (i, j) in the graph G. When there is no directed edge from i to j in G, the corresponding entry is $w(i, j) = -1$. The diagonal entries $w(i, i)$ are set to 0.

The following q lines contain two vertex numbers u and v each.

The output must have q lines (one per query) that contain two numbers—the length $L(u, v)$ of the shortest constrained path from u to v, and a vertex from S visited along this path. (If there are multiple such vertices, you may output any of them.) If there is no constrained path from u to v in G, the output for the query $u\,v$ is the pair of values `-1 -1`.

Sample input

```
5 3
2 4
0 1 2 3 4
2 0 3 3 1
1 2 0 1 -1
1 2 3 0 -1
5 5 -1 1 0
1 3
3 1
3 4
```

Sample output

```
4 2
2 4
1 4
```