Problem Sheet 1

Sourabh Aggarwal
111601025@smail.iitpkd.ac.in

Unless mentioned, assume given graph to be $G = (V, E)$.

1. 1

1.1. (a)

1.1.1. Disjoint Compression

Suppose we have a solution with $k+1$ vertices (say S), then iterate through all subsets of S, let the current subset be $\overline{R} = S \setminus R$ which is part of our new solution. We want a vertex set from $V \setminus S$ of size at most $|R| - 1$ to be part of our new solution. Clearly R with respect to itself should be an independent set as otherwise no edge in it can be covered with $V \setminus S$, if that is not the case, disregard this subset.

For any edge between R and $V \setminus S$, clearly only choice left is to take the vertex in $V \setminus S$ to be in our new solution. Also note that there are no edges between vertices of $V \setminus S$, thus we have covered all edges.

Therefore this procedure will give solution of size $\leq k$ iff such a solution exist, o/w we have a no instance.

Time complexity is $\mathcal{O}^*(2^{k+1}) = \mathcal{O}^*(2^k)$ [*Time to iterate over all subsets is solely exponential and other things can be done in polynomial time*]

1.1.2. Iterative Compression

Consider any $k+2$ vertices and the subgraph associated with it, choose any one vertex from this to form an independent set with respect to itself and other $k+1$ vertices as our vertex cover for this subgraph. Clearly as all these edges are part of our original graph and these edges **can only** be covered by vertices in this subgraph thus if this subgraph doesn't have a vertex cover of size $\leq k$ then the original graph doesn't have one either.

Now in this subgraph we can find a vertex cover of size $\leq k$ using disjoint compression, then we have an indepndent set of size 2 and a vertex cover of size k, now add any remaining vertex of the original graph into our vertex cover, increasing its size by 1. Again apply disjoint compression and proceed similarly to get finally a vertex cover of size $\leq k$.

Time complexity is $T_{\text{disjoint compression}} \times n^{\mathcal{O}(1)} \in \mathcal{O}^*(2^k)$

1.2. (b)

Since complement of an IS is always a VC, therefore given graph has IS of size $\geq k$ iff it has a VC of size $< n - k$ which is therefore my reduction.

1.3. (c)

According to my reduction and iterative compression algorithm, we will have running time as $2^{n-k}n^{\mathcal{O}(1)}$ which is clearly not $2^k n^{\mathcal{O}(1)}$

2. 2

2.1. (a)

2.1.1. Condition is neccessary

Assume that graph (V, E) is acyclic.

Claim: $\exists v \in V; \text{Indeg}\, v = 0$

Proof: Assume not, that implies for each vertex $\text{Indeg}\, v > 0$. Thus start from any arbitrary vertex v and go to vertex u s.t. $(u, v) \in E$. Repeat the same procedure from u. At end we must repeat an already seen vertex by pigeon hole principle, thus giving us a cycle $\Rightarrow\Leftarrow$.

Thus pick that vertex with zero Indeg (remove it from the graph) and add it as a first vertex for our topological ordering. Repeat the same procedure with the resultant graph, and result will be a valid topological ordering as for each edge (u, v), u is coming before v.

2.1.2. Condition is sufficient

Assume that graph (V, E) has a topological ordering $(t_1, t_2, \ldots, t_{|V|})$.

For a cycle to exist there must be an edge (t_j, t_i) such that $j > i$ which is not possible.

2.2. (b)

First direction follows trivially.

For other direction, consider shortest cycle $S = \{v_{i_1}, v_{i_2}, \ldots, v_{i_{|S|}}\}$ (obviously $|S| > 3$), then, either their is an edge (v_{i_1}, v_{i_3}) or (v_{i_3}, v_{i_1}). In former case we get a shorter cycle $\{v_{i_1}, v_{i_3}, \ldots, v_{i_{|S|}}\}$ whereas in later we get a triangle $\{v_{i_1}, v_{i_2}, v_{i_3}\}$ both of which are not possible. $\Rightarrow\Leftarrow$

2.3. (c)

Consider two topological orderings $v_{i_1}, v_{i_2}, \ldots, v_{i_{|V|}}$ and $v_{i'_1}, v_{i'_2}, \ldots, v_{i'_{|V|}}$

Let the first point of difference occur at index k, i.e. $v_{i_1} = v_{i'_1}, v_{i_2} = v_{i'_2}, \ldots, v_{i_{k-1}} = v_{i'_{k-1}}$ and $v_{i_k} \neq v_{i'_k}$. Thus for the first topological sorting, there is an edge $(v_{i_k}, v_{i'_k}) \in E$ whereas in second there is $(v_{k'}, v_k)$ which is absurd.

2.4. (d)

2.4.1. Condition is neccessary

Assume that the set of arcs form a minimal feedback arc set, that implies, removing them we get a DAG which has a topological ordering $v_{i_1}, v_{i_2}, \ldots, v_{i_{|V|}}$. Since our feedback arc set is minimal, that implies adding any edge of it we get a cycle and hence its orientation is backward, that is, it is of the form (v_{i_j}, v_{i_k}) s.t. $j > k$. Thus reversing these edges will give us a DAG as it will have the same topological ordering. Minimality follows because each edge was used to break at least one cycle which is not broken by removing other arcs, thus not reversing it would result in the formation of cycle.

2.4.2. Condition is sufficient

It is clear that the given set form feedback arc set. Minimality follows because each arc if not reversed is part of at least one cycle which is not broken after removing other arcs, thus, this arc must as well be removed.

2.5. (e)

Take any 3 length cycle, $\{v_{i_1}, v_{i_2}, v_{i_3}\}$, from solutions of previous parts, it follows that we must reverse at least one of the edge and add it to our solution set, thus, reducing k by 1. That implies we have three options:-

- $(V, E \setminus (v_{i_1}, v_{i_2}) \cup (v_{i_2}, v_{i_1}), k-1)$
- $(V, E \setminus (v_{i_2}, v_{i_3}) \cup (v_{i_3}, v_{i_2}), k-1)$
- $(V, E \setminus (v_{i_3}, v_{i_1}) \cup (v_{i_1}, v_{i_3}), k-1)$

Branching this way until k becomes 0, we have $3^k n^{\mathcal{O}(1)}$ algorithm.

3. 3

3.1. (a)

Proof by contradiction

Assume length of minimum cycle is $> 2\lceil \log n \rceil$ (*Note:* We must have a cycle as in forest, there exist a vertex of degree ≤ 1). Let v be a part of a shortest cycle. Now do BFS from this vertex, initially we will see at least 3 new neighbors but doing BFS from these neighbors we can see only at least 2 new neighbors as we have already seen one of its neighbor (its parent). Since length of minimum cycle is $> 2\lceil \log n \rceil$, we will see new neighbors till $1 + 3 + 3 \cdot 2 + 3 \cdot 2^2 + \cdots + 3 \cdot 2^k$ where $2(k+2) > 2\lceil \log n \rceil$ for even length cycle, for odd length cycle we will have $2(k+1) + 1 > 2\lceil \log n \rceil \to k + 1 > \lceil \log n \rceil - 1/2$ whereas in even case we had $k + 1 > \lceil \log n \rceil - 1$. Proceeding with even case, we have the sum equal to $3 \cdot 2^{k+1} - 2 > 3 \cdot 2^{\lceil \log n \rceil - 1} - 2 > 3/2 \cdot 2^{\log n} - 2 > 3n/2 - 2$ since this must be less than total number of vertices, we have $n > 3n/2 - 2$, i.e., $4 > n$. But since minimum degree is 3, we have $n \geq 4$. $\Rightarrow\Leftarrow$. Similarly for odd case we will arrive at $n < 1.79$, giving us again a contradiction.

3.2. (b)

Using those 5 reduction rules as discussed in class, we will have for each vertex $v \in V$, $\deg v \geq 3$. Thus we can apply the above theorem, let $\{v_{i_1}, v_{i_2}, \ldots, v_{i_l}\}$ be a cycle where $l \leq 2\lceil \log n \rceil$. Thus now we have a branching algorithm for each vertex as we must put at least one vertex from this cycle into our solution set so as to break this cycle, applying wherever possible our reduction rules to new instances.

Thus time complexity is $(\log n)^{\mathcal{O}(k)} n^{\mathcal{O}(1)}$.

It is not fixed-parameter tractable as $(\log n)^{\mathcal{O}(k)}$ does not depend solely on k.

3.3. (c)

Was unable to do ☹.