29/9/23

Date:
Page: 1

Parametrized Complexity
Assignment 4.

1. Given → A graph $G(V, E)$ & an integer k.

Parameter → k.

Goal → ~~\$~~ Find $S \subseteq V(G)$ of size atmost k ($|S| \leq k$)
S.T. $G - S$ does not contain a cycle of length 7.

Reduction Rules.

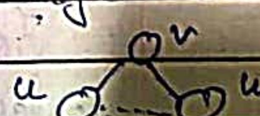
Rule 1. Degree ~~of~~ 0 & 1 vertices.

Forward
We can safely remove the vertices with degree 0 & 1 as they do not contribute in making of a cycle.

Backward.
Adding vertices of degree 0 & 1 in the reduced graph does not change the cyclic nature of graph as they don't contribute in cycle formation.

Rule 2. Degree 2 vertices.

If a vertex v has a degree 2 & is adjacent to vertices u & w then remove v & add an edge between u & w. It doesnot creates a new cycle



Date:
Page:

(2)

Rule 3 Vertices with degree >2

For vertices with $d>2$, we can remove all but two of its incident edges. This is because vertices with $d>2$ can't be part of a cycle of length 7.

Assume a vertex v with degree >2. As v can't be the part of cycle of length 7 it means that v must have atleast $d-1$ leaf neighbours. This is because in each cycle each vertex has exactly 2 neighbours.

If we have $d>2$ edges incident on v, we can keep only two of them & remove the rest. This ensures that v still has atleast $d-1$ leaf neighbours The removed edges do not contribute to any cycle So we can discard/ Remove them.

Observation

Forward.

Applying the reduction rules decrease the size of graph. Rule 1, 2, 3 will reduces the vertices of graph & Rule 4 reduces the #edges

Backward.

For any set $S \subseteq V(G)$ after applying the reduction rules adding back the vertices removed in Rule 1, 2 & 3

& restoring by rule 4 produces a graph that is valid / same as of given graph.

Kernal Size

1. Removing all the vertices with degree 0 does not change the value of k so no effect on kernal size.

2. Removing all the vertices with degree 1 we decrease the value of k by 1 for each vertices. This can remove atmost k vertices.

3. Removing a vertex of degree 2 & connecting its neighbours does not change value of k so no effect on kernal size.

4. For each vertex removed by rule 4 we remove atmost 2 edges. Therefore this rule can remove atmost 2k edges

Thus the kernal size ⇒ 2k + k + k + 4
⇒ 4k+4 atmost

Runtime Analysis

1. Scanning all vertices to identify isolated vertices takes O(V(G)) time.

2. Removing vertices of degree 1. This is also be done

(4)

in $O(V(G))$ time.

3. Handle vertices of degree 2. We iterate through the vertices of G once & apply this rule. & can be done in $O(V(G) + E(G))$ time.

4. Handle vertices of degree ≥ 2. This can be done in $O(V(G) + E(G))$ time.

Thus total time is $O(V(G) + E(G))$ & it runs in polynomial time.

(b) We have a kernel size of $4k+4$

The algorithm runs in $O(V(G) + E(G))$

→ In the worst case we have to do 2^k recursive calls.

→ For each time it is taking $O(n^2)$ time.

Thus Total time is $O(V(G) + E(G) + \underbrace{n^2 2^k}_{\text{branching}})$

As our kernel size is of $4k+4$ So enumerating each set will take 2^{4k+4} iteration which is bounded by $6 \cdot 27^k$ for $k \geq 6$

For each time deleting the cycle time taken is of $O(n^2)$ via DFS to detect cycles.

Overall Runtime is $O(6.27^k \cdot n^2)$ or

$\Rightarrow O(6.27^k \cdot n^{O(1)})$