Mahya Jamshidian
9525133

Homework 3

Problem 1

Since Tree width of a graph is the maximum in number of nodes present in all of bags in the minimum tree decomposition, at least one of the bags has K nodes $\longrightarrow$ number of bags are less than number of nodes minus k $\longrightarrow |V(T)| \leq |V(G) - k|$

For the latter theorem, we make use of proof by induction:// Base ; $|V(G)| = 1$ since tree width is 0 then $|E(G)$ ¡= 0 which is trivial.
for case $|V(G)|$ = d which is the inductive hypothesis, we add another node v, since the treewidth is still k, we only may add this vertice to bags with less than k vertices, Thus we may only add k edges connecting v to some bags.

Problem 2

MAX INDEPENDENT SET : f explanation : this function works as we hang our tree decomposition of of an spot and mark the highest node as root. f tries to explore the tree beneath one node as i starting from a seed like S. so it expands to other nodes from S through i's childrern. it chooses maximum independent set from the subtree i wich its intersection with i is S.
leaf function explaination : when the recursive function reaches some leaf, it checks if S(the expanding seed) is independent (k^2) then return its size if it and a very small O.W.
on nodes other than leaves, is S is an independant set, and i has $c_1, c_2, c_3, ..., c_l$ as its children. Now that we know S is independent, while using X_i childern for expanding the tree, we must preserve what is in S and not in the children. Then for every child i_j we choose a subset of X_{cj} where its intersection with its father is equal to S intersection with that child. this tries to keep the independence property while making a way to children. Note that, The algorithm checks if $W \cup S$ is independent to go with recursion step.

DOMINATING SET PROBLEM: function f explanation : we set the output to be the minimum size of sets at level i with value of g in which value of g for nodes in both X_i and D is equal to 1 and every node with value other than 2 is either in D or is adjacent in D.
forget node explanation : since there is one node in X_j that is not in X_i namely u, first we add (u,1) to g calling it g_1 and call g(j,g_1) and then we add (u,0) to g calling it g_1 and call g(j,g_1) and we assign to f(i,g) minimum of those calls. the first chooses the node and further expands it while the latter leaves that to the rest of the graph.
join node explanation : for g and a join node X_i for X_j1 and X_j2, which $X_i = X_j1 = X_j2$ every vertex that its value g is 0, can be ignored by one of the children;Thus, we add up the value of function f for one time ignoring some node u in j1 and another time in j2. note that the subtraction works as it will eliminate double counted vertices which had already been choosen (denoted by g value 1).

i believe that this max has to be modified to min.

introduce node explanation : Since X_j does not care for g-value of node, it simply calls X_j for j and other g values if we have chosen to ignore u in this sectoin or its neighbors have already been chosen. Also, if we have decided to choose u, we make sure all of its neighbors now are relaxed to value 2 from 0. and if we have to choose u and have not chosen any adjacent yet, we return a very high value since we are introducing u and it is not present in X_j, all of its neighbors are in X_j.

Problem 3

a

if $d \leq 2$ then solve (G,k) in polynomial time. $|E(G)| = dn/2$; if (G,k) is a yes instance and the solution is X. X is at most k, then at most dk edges are incident with it. On the other hand, G has at most n-k-1 edges (because it is a forest now).

$dn/2 \leq dk + n - k \longrightarrow (d/2 - 1)n \leq (d-1)k$
then n is bounded by o(k).

b

Now that we know r is constant, we make a few preprocessing step before we reach exhaustive search part.

Firstly, if $d \leq k$ there is no chance at getting k-cliques so it is an abvious No instance.

For every v in V(G):

—-For every subset of size k in N(v):
———check if it is a clique

the overall running time would be $n.k^{o(1)}.C^{d}_{k-1}$ since every node at most have d neighbors.

for parameter d+k, for every value of r' in (r+k)//2 to r+k: —- k is (k+r) - $K_{m,n}$ overall running time is: $n.(\Sigma^{(r+k)//2})_{i=0}(((r+k)//2)^{o(1)}.C^{(r+k)//2+i)}_{(r+k)//2-i})$ which is FPT.

Problem 4

For a given instance (G,K), we try to prove the following lemma,

Lemma : A graph G is Cluster iff it does not contain any path with 3 vertices as induced subgraph (P_3).

($\Longrightarrow$) Since all connected components of G are cliques, if they are 1 or 2 cliques, they do not contain any 3 vertices, then the case is trivial; If the connected component has more than 3 vertices, then every 3 vertices that have 2 pair of adjacent nodes, it must have the the third edge as every node must connect to very other node.

($\Longrightarrow$) Since all connected components of G are cliques, if they are 1 or 2 cliques, they do not contain any 3 vertices, then the case is trivial; If the connected component has more than 3 vertices, then every 3 vertices that have 2 pair of adjacent nodes, it must have the the third edge as every node must connect to very other node.

($\Longleftarrow$) in any graph, if every three node form a triangle, every node is connected to every other node, Thus we are dealing with a clique. in other words if in any graph, every three node form a triangle or is not a 3-vertice path induced subgraph, the graph is clique.

Now for the instance (G,k), one may found a family of sets S that contains any P_3 in G. The solution X has to be incident with every set in S.

Algorithm on input (G, k)

- 1) if G is cluster graph, return YES
- 2) if k = 0 return NO
- 3) find a P_3 in G
- 4) for v_1, v_2 and v_3 run the algorithm (G $\setminus$ v_i , k-1)

Checking graph for extracting P_3 and checking whether it is Cluster takes polynomial time and branchin algorithm goes deep as k level 3-childern tree.

Problem 6

Using 5 reduction rules introduced at class for FVS, we make a instance (G', k') from (G,k) which has following properties:

- each node has least degree of 3.
- Atleast half of the edges in $E(G^{'})$ has one endpoint in $X^{'}$

Now we choose one edge completely random from $E(G^{'})$; By second property, with probability larger than 1//2 it has one endpoint in $X^{'}$ and with probability larger than 1//4 that endpoint is v which is chosen randomely from 2 endpoint of chosen edge.

After finding v, we run the algorithm again for $(G^{'} \backslash \{v\},\, k^{'}-1)$.

using proof by induction, if Reduction Rule 5 is not triggered for returning NO for $k \leq 0$, basic step is described.

for inductive hypothesis, assume for instance $(G^{'} \backslash \{v\},\, k^{'}-1)$ recursive call, it has probability of $4^{-(k^{'}-1)}$ to give out X" as FVS, and probbaility of finding next vertex is 1//4 $\Longrightarrow$ overall probability for instance $(G^{'},\, k^{'})$ to find $X^{'}$ is $1//4.4^{-(k^{'}-1)} = 4^{-(k^{'})} \geq 4^{-(k)}$.
Now all we have to do is to prove third property:
If F is the produced forest by removing X from, it is enough to show $E(G) \backslash E(F) \geq |E(F)|$ and since F is forest it is enough to show $E(G) \backslash E(F) \geq |V(F)|$ and V(F) is composed of vertices with degree less than 1 :
have atleast two incident with X by their connected edges, since every node in G has atleast degree 3. and by tree properties, number of vertices with degree less than one is more than number of vertices with degree more than tree
$E(G) \backslash E(F) \geq \#v_{\leq 1} \geq \#v_{\geq 3} \geq V(F)$.

Probelem 7

a

Consider a nice tree decomposition of G, for any introduce node and tree-width at most k, it is sufficient to show G contains a vertex at most k to show every subgraphs of G noted by bags of tree decomposition has atleast one vertex of degree k and Thus is k-degenerate.

For a path from a bag with empty set to furthest forget node, if tree width is k then by adding each node by introduce nodes, even if we add every new edge somehow that new vertex is adjacent to one that already exists, at bag with size k+1, we at most have a vertex with degree k and by forgetting any of existing vertices, we may lose one degree of that vertex and new bag.

Therefore, every subgraph of G contains vertices at most k.

b

1) Treewidth of $K_{m,n}$ is at least min(m,n):

Since $K_{m,n}$ is min(m,n), it has treewidth atleast min(m,n).

2) Treewidth of $K_{m,n}$ is at most min(m,n), by choosing a subset S of size min(m,n) from V($K_{m,n}$) and choosing ever other $v_1, v_2, ..., v_{m+n-min(m,n)}$ to form a child of S with vertices $S \cup \{v_i\}$.