Mahya Jamshidian
9525133

Take Home Exam 3

Problem 1

We will design some reduction ruled while constructing our kernel;
Problem instance (U,F,K): By providing a random coloring for Universe U, at least half of the family F are non-monochromatic $\Longrightarrow$ if $|F| \geq 2k$ Return YES.

Reduction Rule 1 : for $S \in F$ if $|F| = 1$ (U,F,K) $\longrightarrow$ (U,F $\setminus$ S, k)

Proof : Since it does not have 2 vertices to be actually non-monochromatic.

Reduction Rule 2 : if (U, F , K) is a yes instance and there is a set $S \in F$ s.t. $|F| \geq 2k$ $\rightarrow (U, F\setminus$ S , k-1): *proof* : then we choose k-1 sets from k non-monochromatic sets and choose two vertices with different colors from each. Now, there is at most 2k-2 vertices in S has colors so we at least have still two vertices which we can assign colors to them arbitarily. so by removing S from universe and decresing k by one, (U,F $\setminus S, k-1) is still a YES instance.$

Family size: 2k
Universe size: 2k(sets) * 2k(members of each set)

Problem 2

For grah g there exist a tree decomposition denoted by T(G) and with tw(T(G)) = k.

Let X_is be the bags provided by tree decomposition and V_i to be union over all nodes available in subtree at level i rooted by X_i. We later provide some partitioning through function g for V(G) into tree subcategories 0,1,2 in which for v in V(G):

- if g(v) = 0 : v is present in the matchin set V(M).
- if g(v) = 1 : v is not present in V(M) and yet it has to be.
- uf g(v) = 2 : v is not present in V(M) and it does not have to be.

Now for every i in V(T(G)) and every function g from $X_i \rightarrow \{0, 1, 2\}$:
f(i,g) = $\{M : 2|M| = |V(M)| and \forall v \in V(M) g(v) = 0 and for (v_i, v_j) in E(M) : v_{i,j} \notin V(M)$ $\setminus\{$v$_{i,j}\}\}$

W.l.o.g we can assume that T(G) is a nice tree decomposition, with 3 type of nodes (Forget, Introduce, Join).

For a forget node i, $X_i = X_j \setminus \{u\}$:

We add an entry in g for g(u) = 0 interpreted as $g^{'}$ and run f(j,$g^{'}$)

Again, We add an entry in g for g(u) = 1 interpreted as $g^{''}$ and run f(j,$g^{''}$)

f(i,g) = maxf(j,$g^{''}$),f(j,$g^{'}$)
For a join node i, $X_i = X_{j_1} \cup X_{j_2}$, for every v in X_i :

- We modify entry in g for g(v) = 0 interpreted as $g^{'}$ and run f(j_1 ,$g^{'}$) + f(j_2 ,g)
- Again, We add an entry in g for g(u) = 1 interpreted as $g^{''}$ and run f(j,$g^{''}$)
- f(i,g) = maxover the output of previous parts output - $|g^{-1}(0)|$

For an introduce node i, $X_i = X_{j_1} \cup \{u\}$:

- if g(u) = 2 then we run $f(j, g\|_{X_j}$ since it does not make any difference
- if g(u) = 1 for every v in $N(u) \cap g^{-1}(1)$: $max_v(f(j, g^{'}))\ g^{'}(v) = 0$
- if g(u) = 0 and N(u) $\subseteq g^{-1}(0)$ retrun -inf
- for every v in $N(u) \cap g^{-1}(1)$: $1 + max_v(f(j, g^{''}))\ g^{'}(v) = 0$

Run f(r,g) for all g : $X_r \to 0, 1$

the over all running time is equal to $2^{\|X_r\|} * n^{O(n)}$ for adjacency checking.

Problem 3

Suppose A is an FPTAS algorithm for P.
Set $\epsilon = 1/(k + 1)$ if $A_P(I) \geq k$ then accept since $OPT_P(I) \geq k$ as well. if $A_P(I) \leq k$ then reject.

proof: $OPT_P(I) = A_P(I) * (1 + 1/(k + 1)) \leq (k(k + 1))/(k + 1) \leq k$