Assignment 4
CSCI 6101

Yansong Li
B00755354

Question 2 (Branching Algorithm)

(a) Goal : finding an algorithm that solves set cover problem in $O^*(f^n)$ time.

1. We arbitarily pick an element e with possible maximum frequency f, which means e exists in all subsets S_i $(1 \le i \le f)$. For the set cover problem, we need to cover every element $e \in U$, hence, the depth of the recursive parts is n. Also, since the possible maximum frequency of e is f, there could be f outcomes of which e in S_i is included in C.

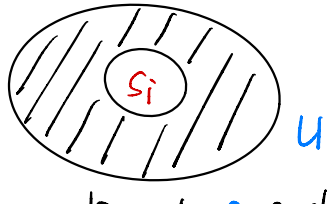
$e \in S_i \ (1 \le i \le f)$?

$\le n-|S_1|$ $\le n-|S_2|$ $\cdots$ $\le n-|S_f|$

for each outcome i in f

$C = C' \cup S_i$

where C' = minimum set cover of $U - S_i$



Since picking some elements e and constructing some graphs $U-S_i$ only takes linear time, which is $O(n+m)$. Meanwhile, the algorithm recurses on graph $U-S_i$ and there are at most f outcomes, so the recursive parts take at most $T(U-S_1)+\cdots+T(U-S_f)$ time. Hence, the total running time for set U would be

$$T(U) = O(n+m) + T(U-S_1) + \cdots + T(U-S_f)$$

Since we have the input universe size $|U|=n$, and we at least remove element e for each $U-S_i$, which is $n'=n-|S_i| \le n-1$. Hence,

$$T(n) \le O(n+m) + f*(T(n-1)) = O^*(f^n)$$

As a result, we have found an algorithm that solves the set cover problem in $O^*(f^n)$ time.

Validation:

Branching Vector $= (|S_1|, |S_2|, \dots, |S_f|)$

$\min(|S_i|) = 1$

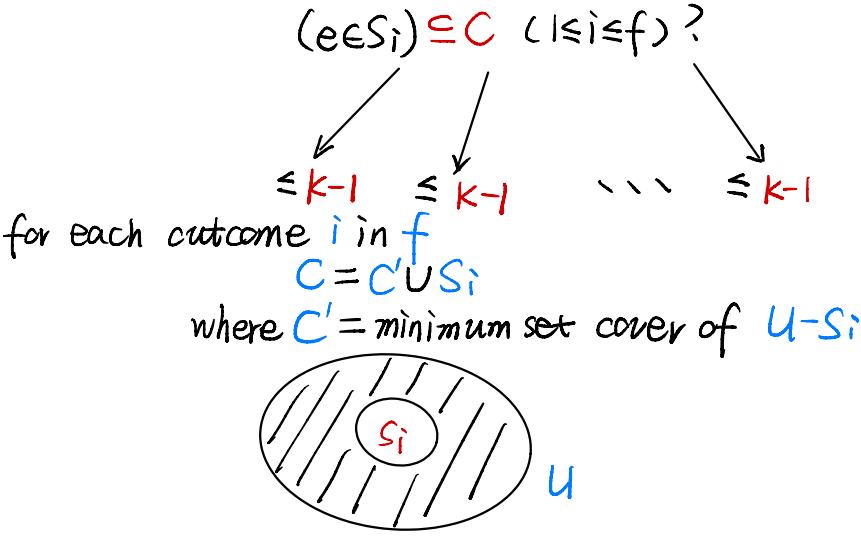
$\max(|S_i|) = n$

$$c^n \le c^{n-1} \cdot f \Rightarrow \frac{c^n}{c^{n-1}} \le f \Rightarrow c \le f$$

$$\Rightarrow T(n) = O^*(f^n)$$

(b) Goal: By assuming every set in S has weight 1, provide an algorithm that solves the set covering problem in $O^*(f^k)$ time.

We arbitarily pick an element e with possible maximum frequency f, which means it can exits in at most f subsets S_i. Since every subset S_i has the same weight 1, so when we decide to include an element e in a particular subset S_i to set cover C, we can also immediately include all the other elements in S_i to set cover C because we don't care which subset S_i contains which elements, we just looking for the minimum number of subsets that can cover U. Therefore, the depth of the recursive parts is bounded by $|S| = k$. And there are still f outcomes of which e in S_i will be included in C



Still, picking some elements e and constructing some graphs $U-S_i$ take only linear time $O(n+m)$. Similarly, the algorithm recurses on graph $U-S_i$ and there are at most f outcomes, so the recursive part take at most $T(U-S_1)+\cdots+T(U-S_f)$ time. Hence, the total running time for set U would be

$$T(U) = O(n+m) + T(U-S_1) + \cdots + T(U-S_f)$$

Since we have bounded the recursion depths to $|S|=k$, and we remove one entire subset S_i for each recursion, which is $k'=k-1$. Hence,

$$T(n) \le O(n+m) + f * (T(k-1)) = O^*(f^k)$$

As a result, we have found an algorithm that solves the set cover problem in $O^*(f^k)$ time when all subsets have weight 1.

Question 3 (Dynamic Programming)

The naive algorithm of finding an optimal valid vertex coloring runs in $O^*(n^n)$ time.
Goal: find a better solution of it runs in $O^*(3^n)$ time.

1. Find and list all subsets $S \subseteq V$ that are 1-colorable (can be colored with one color). since there are n vertices in G, $|V| = n$, and for each vertex v we can choose either include it or not in each subset $x \in S$. Therefore, there are 2^n subsets in S, $|S| = 2^n$, which means S has 2^n entries. Hence, constructing the table take $O^*(2^n)$ time.

2. For each 1-colorable subset $x \in S$, there is a set of remaining vertices $y = V / x$, we call of the sets of remaining vertices $y \in Z$. Since each subset $x \in S$ will result one set $y \in Z$, the size of Z equals to the size of S, $|Z| = |S| = 2^n$. Therefore, we have 2^n 1-colorable subsets x and 2^n remaining vertices sets y.

3. For each set of x and y, check whether the combination of them is k-colorable. since we already know all $x \in S$ are 1-colorable, in terms of finding a combination is k-colorable, we also need to find a set of remaining vertices y is $(k-1)$-colorable. Verifing a remaining vertices set y is $(k-1)$-colorable or not take polynomial time because vertex coloring is a NP-complete problem. Since $|Z| = 2^n$ and $|S| = 2^n$, so for each entry $x \in S$, it takes $O^*(2^n)$ time to verify all the combinations of $y \in Z$ with current x hav a k-colorable combination or not. Therefore, using S table to find a k-colorable combaniation takes

$\sum_{\phi \subseteq S \subseteq V} O^*(2^n)$. Therefore, the total time of finding a valid k-coloring takes $\sum_{\phi \subseteq S \subseteq V} O^*(2^n) + O^*(2^n) = \sum_{\phi \subseteq S \subseteq V} O^*(2^{n+1})$

$= \sum_{\phi \subseteq S \subseteq V} O^*(2^n)$

4. Next, we use some arithmetic opreations to obtain $O^*(3^n)$ from $\sum_{\phi \subseteq S \subseteq V} O^*(2^n)$.

$$\sum_{\phi \subseteq S \subseteq V} O^*(2^n) = \sum_{i=1}^{n} O^*\left(\binom{n}{i} 2^i\right)$$

$$< \sum_{i=0}^{n} O^*\left(\binom{n}{i} 2^i 1^{n-i}\right)$$

$$= O^*((2+1)^n)$$

$$= O^*(3^n)$$

$(x+y)^k = \sum_{i=0}^{k} \binom{k}{i} x^i y^{k-i}$

$x=2, y=1$

Hence, we have found an algorithm that solves vertex coloring problem in $O^*(3^n)$ time.