CS 4820, Spring 2014 **Homework 10, Problem 2**
Name: Piyush Maheshwari
NetID: pm489
Collaborators: None

2. *(10 points in total)*

2.a. *(5 points)* Consider the special case of unweighted set cover in which every element belongs to at most k sets. Design a polynomial-time $O(k)$-approximation algorithm for this problem.

2.b. *(5 points)* Consider the same specialization of weighted set cover and design a polynomial-time $O(k)$-approximation algorithm for it. (If you are confident in your solution of this part, you can simply write, "Special case of 2b" as your solution to 2a. In that case we'll simply double your score for this part of the problem so that it counts for 10 points instead of 5.)

Solution

2.a. Special case for 2.b

2.b. The problem is given a set U of n elements and a list of sets S_1 to S_m with weights w_1 to w_m, find a subset of these sets such that all elements are covered that the weights of the subsets chosen is minimized. The algorithm for this problem works on similar lines as the weight vertex cover algorithm presented in class. We will use the pricing method to approximate. We will assign a price to every element e such that for any set S_i we have $\sum_{e \in S_i}^{n} price(e) < W(S_i)$. This basically means that it's always cheaper to buy individual elements rather than buying the whole set.

Let U be the set of all elements which needs to be covered. Let OPT be the optimal solution for weighted set cover problem. This means that OPT pays for some sets which cover all the elements in the U. Since it is always cheaper to buy individual elements we have $OPT \geq \sum_{e \in U}^{n} price(e)$. This would be serve as the lower bound on OPT which would be used in approximating our solution.

Algorithm

Algorithm 1

```
1: S ← ∅                                              ▷ The solution set is initially empty
2: price(x) ← 0 $\forall x \in U$                                        ▷ Initial price is 0
3: s(i) ← 0 $\forall s_i$        ▷ s(i) represents the sum of prices of elements belonging to set $s_i$
4: while ∃ uncovered element y do
5:     $\delta$ ←min { w(i) - s(i) } $\forall i$ | y ∈ $S_i$
6:     price(y)←$\delta$
7:     s(i)← (s(i) + $\delta$) $\forall i$ | y ∈ $S_i$
8:     for all $\forall i$ | y ∈ $S_i$ do
9:         if s(i) = w(i) then Add i to S
   Output S
```

Proof of Correctness

First note that for each uncovered element y picked in the one iteration of the while, there is always one set s_i added to the set S which covers y. This is because we choose δ such that the condition in line 9 of the algorithm is true for atleast one set s_i. And since the algorithm does not terminate until we have covered all element, the set S which is finally output is a valid set cover. This also shows that the algorithm terminates since there are a finite number of elements in U and atleast one of them gets covered in one iteration.

Now we show that the output S is a $O(k)$ approximation of the optimal algorithm to solve the weighted set cover problem.

First note that $\sum\limits_{\forall s_i} s(s_i) = k * \sum\limits_{x \in U} price(x)$. This holds true because any time we set the price of an element as δ we increment the summation on the left hand side by k * δ. Also once we fix the price of an element while processing it, we don't change it again.

At termination, S is the set of all sets which constitute the set cover. Note that for all sets in S we have w(s_i) = s(s_i). This is because we only add a set s_i when this condition holds true on line 9 of the algorithm.

Let the set of all sets s_i be S_u

w(S) = $\sum\limits_{\forall s_i \in S} w(s_i)$
w(S) = $\sum\limits_{\forall s_i \in S} s(s_i)$
$w(S) \leq \sum\limits_{\forall s_i} w(s_i)$, since $S \subset S_u$
$w(S) \leq k * \sum\limits_{x \in U} price(x)$

$w(S) \leq k * w(OPT)$

This proves that our approximation algorithm is a k approximation.