ICCS313: Assignment 5

Natthakan Euaumpon
natthakaneuaumpon@gmail.com
November 2019

1: Problem1

(1)

```
def delete(table, elem):
    if (sizeof(table) == 0):
        return table
    else if (num(table)/sizeof(table) < (1/4)) //Check if the ratio between number of elem and tble size < 1/4
        Create new table called newTable with (1/2) * sizeof(table) slot
        insert all item to the newTable
        free(table) //free all items from table
        table = newTable
        sizeof(table) = (1/4) * sizeof(table)
    remove elem from table
    num(table) = num(table) - 1 //remove elem, total elem in the table decrease by 1
```

(2)

By accounting method we going to assign value into each operation inside the function.
Let elem be an elem needed to be delete and table be a table containing elem.
Charge \$2 per deletion of elem:
\$1 pays for the deletion of elem
\$1 pays for an emptying slot
WLOG, let assume that the table with size of k has no credit left in any slot and there is $\frac{k}{2}$ amount of item inside the table. Let m be the number of slots that contain items, when we perform a deletion of elem we pay \$2. One is use for deletion and another is save for each item in the slots. When our load factor ¡ $\frac{1}{4}$ we create a new table of size $\frac{k}{2}$. Then we copy the element from the old table to the new table. As we save \$1 for each non-empty slots, there would be enough credit to copy $\frac{k}{4}$ amount of item to a new table. This means there is no credit left over and the total credit will never be negative value.

2: Problem2

Claim: set-partition is MP-complete
set-partition $\in$ NP:
Given $A \subseteq S$ where A is a set of numbers and S which is also a set of number.
We can create a verifier to check if the solution run in polynomial time using and algorithm below.

 Natthakan Euaumpon
natthakaneuaumpon@gmail.com

```
1  def verifier(S,A):
2      sumS = 0
3      sumA = 0
4      for i in S:
5          sumS += i
6      for i in A:
7          sumA += i
8      if (sumS == 2*sumA):
9          return true
10     else:
11         return false
```

This algorithm time complexity is O(n). This is because we have 2 for-loops each run to all the elements in each set (line 4 and line 6). Let n be the length of S and m be the length of A. Other steps inside a loop take O(1). Instantiating an element in line 2 and 3 also take O(1). So $O(m \cdot 1) + O(n \cdot 1)2 \cdot O(1) = O(n)$. As n id greater than m. $\lim_{n \to \infty} \frac{n}{n} = 1$. This mean it is in $\Omega(n)$ which means it is O(n).
set-partition is NP-hard:
By reducing set-partition to sub-set sum. subset-sum is a known NP problem that an instance $< A, t >$ will be computed in polynomial time complexity.
Let subset-sum be the summation of S, now we claim that:
$subset - sum \leq_p set - partition$
Claim: C is a solution for subset-sum $\iff$ D is a solution for set-partition
($\Rightarrow$) Suppose C is a solution in subset-sum where C takes A and S. We know that set A and a target $\frac{subset-sum}{2}$. S can be partition into set A and A' where A' = S - A. This then form a set-partition solution for S.
($\Leftarrow$) Suppose D is a solution in set-partition which takes S and A. Set A is a set that sum to a number in which when we double it, it equals the the sum of S. Which is a subset-sum solution when it takes A as an input with a target $\frac{subset-sum}{2}$.

3: Problem3

Claim: Efficient-Recruiting is NP-Complete
Efficient-Recruiting $\in$ NP:
Given setK $\subseteq$ setM where setM is a set of all counselor and setK is a subset of setM.

Let map be a map of sports to key_i ans set_i which is a subset of applicant in setM.
We can create a verifier that take setK ans Map as an input to check if the solution is in polynomial time using an algorithm below.

```
//m = no. of applicant
def verifier(setK, map):
    if (sizeof(setK) <= m):
        return Null
    found = false
    for i in map.keyset: //O(n)
        subset = map.get(i)
        for j in subset: //O(seti)
            for k in setK: //O(k)
                if (k==j):
                    found = true
                    break
            if (found):
                continue
            return false
    return true
```

The running time is:

$$\sum_{i=0}^{n-1} set_i = O(n^2 k)$$

Where n is the total number of element in map. Efficient-Recruiting is NP-hard:
By reducing Efficient-Recruiting to vertex-cover. Vertex-cover for a graph G is a subset of vertices that covers all the edges in E. Let s be a size of counselors, the instance $< G, s >$ will be computed in polynomial time.
In order to remove this into a vertex-cover we needs to create a graph from Efficient-Recruiting problem. Let G be an undirected graph G = (V,E), where V are vertices and E are edges. Given M as a counselors vertices in V and N as a sports vertices in V, i_{th} vertices are connected to all vertices in M. Assumes there is at least one edge in every vertex in N that connects to vertex in M. $\bar{G} = (V - M'), \bar{E})$ where $\bar{E}$ are edges after M' is removed from V to create $|V - M'| = k$ which is the counselors we will be hiring.
$Vertex - cover \leq_p Efficient - Recruiting$
Claim: C is a solution for vertex-cover $\iff$ D is a solution for Efficient-Recruiting
($\Rightarrow$) Suppose C is a solution in vertex-cover of instance $< G, k >$. This mean for $k = |M| - |M'|$ we can form a vertex cover of size k and every edges are covered within it. This then form an Efficient-Recruiting solution for N sports and at most k counselors.

Natthakan Euaumpon
natthakaneuaumpon@gmail.com

($\Leftarrow$) Suppose D is a solution in Efficient-Recruiting of instance $< \bar{G}, k >$, this means $\bar{G}$ has at least one edge in N connecting to a vertex in $V - M' = V'$ and the size V' is k. From the solution, $\bar{G}$ can use V' vertices to connect to all vertices and can cover all the edges in $\bar{E}$. Which then forms a vertex-cover solution for graph $\bar{G}$ with a vertex cover of size k.