To Perform Ringsum Operation on Given Two Sets of Positive Integers

Group Members

Aakash Anand (IIT2019015)

Parth Kataria (IIT2019016)

Shruti Nanda (IIT2019017)

Introduction

This paper contains the algorithms to perform only node operation of ringsum operation on two sets of positive integers which represents nodes of the two graphs.We will discuss time complexities of all algorithms and differences in them.

Definition - Given two graphs G1 = (V1, E1) and G2 = (V2, E2) we define the ring sum G1 ⊕ G2 = (V1 ∪ V2,(E1 ∪ E2) − (E1 ∩ E2)) with isolated points dropped. So an edge is in G1 ⊕ G2 if and only if it is an edge of G1 , or an edge of G2, but not both.

Node Operation-(V1 U V2)
Edge Operation-(E1 ∪ E2) − (E1 ∩ E2)

Algorithm Design

We have designed three algorithm For finding the union of two sets of positive Integers.

1. Brute-force
2. Union
3. Searching and sorting

1. Brute Force

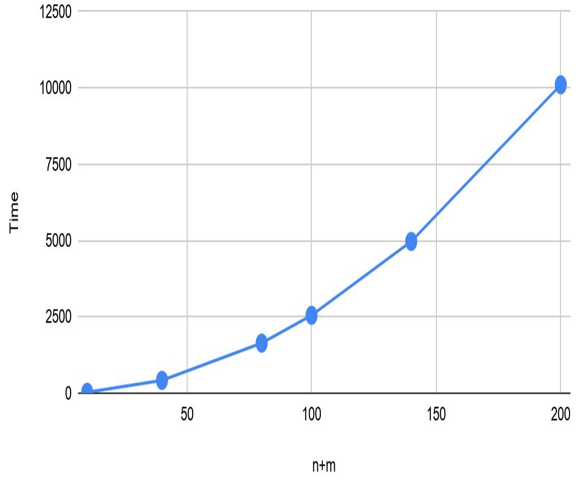
In this Algorithm, We print all the elements of the larger set and all elements of the smaller set which are not present in the larger set using brute-force searching.

Time Complexity

Best case : Ω(n+m)
Average case : θ(n+n*m)
Worst case : O(n+n*m)

Space Complexity: O(n)

Figure1:Time Complexity of Algorithm1



Pseudo Code for Brute force

```
void brute(int n, int m, int set1[], int set2[])
{
        int i,j;
        for(i=0;i<n;i++)
                printf("%d ",set1[i]);
        for(i=0;i<m;i++)
        {
                for(j=0;j<n;j++)
                        if(set2[i]==set1[j])
                                break;
                if(j==n)
                        printf("%d ",set2[i]);
        }
        printf("\n");
}
```

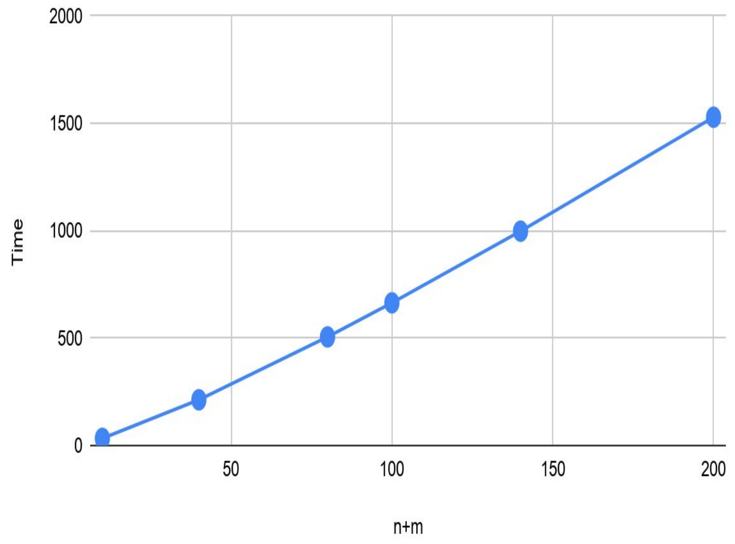
2. Union

In this algorithm we first sort the two arrays and use two pointers we compare elements of both the sets.

Time Complexity
Best case : Ω(n+m)
Average case : θ(n(logn + 1) +m(logm + 1))
Worst case : O(n(logn + 1) +m(logm + 1)

Space Complexity: O(n)

Figure2:Time Complexity of Algorithm2



Pseudo Code

```
int cmp(int a,int b)
{
        if(a<b)
                return a;
        else
                return b;
}

void union(int n,int m,int set1[],int set2[])
{
        qsort(set1,n,sizeof(int),cmp);
        qsort(set2,m,sizeof(int),cmp);
        int i = 0, j = 0;
while (i < m && j < n) {
      if (set1[i] < set2[j]) {
         printf("%d ", set1[i]);
         i++;
      }
      else if (set2[j] < set1[i]) {
              printf("%d ", set2[j]);
              j++;
         }
      else {
         printf("%d ", set2[j]);
         i++;
         j++;
      }
   }
   while (i < m)
   {
      printf("%d ", set1[i]);
      i++;
   }
   while (j < n)
         {
      printf("%d ", set2[j]);
      j++;
   }
}
```

3. Searching and Sorting

In this Algorithm We first find out which array is smaller and sort it in O(Nlogn) complexity. Then we will apply binary search on each element of another array in O(mlogn) and if the element is present in the first array then we will ignore it and print the element which is not present in the first array.

Time Complexity

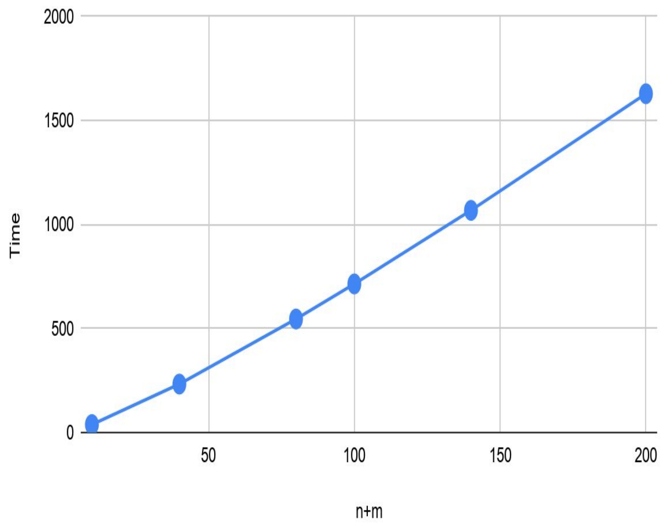
Best case : Ω(mlogm+n)

Average case : θ((m+n)log(min(m,n)))

Worst case : O((m+n)log(min(m,n)))

Space Complexity: O(n)

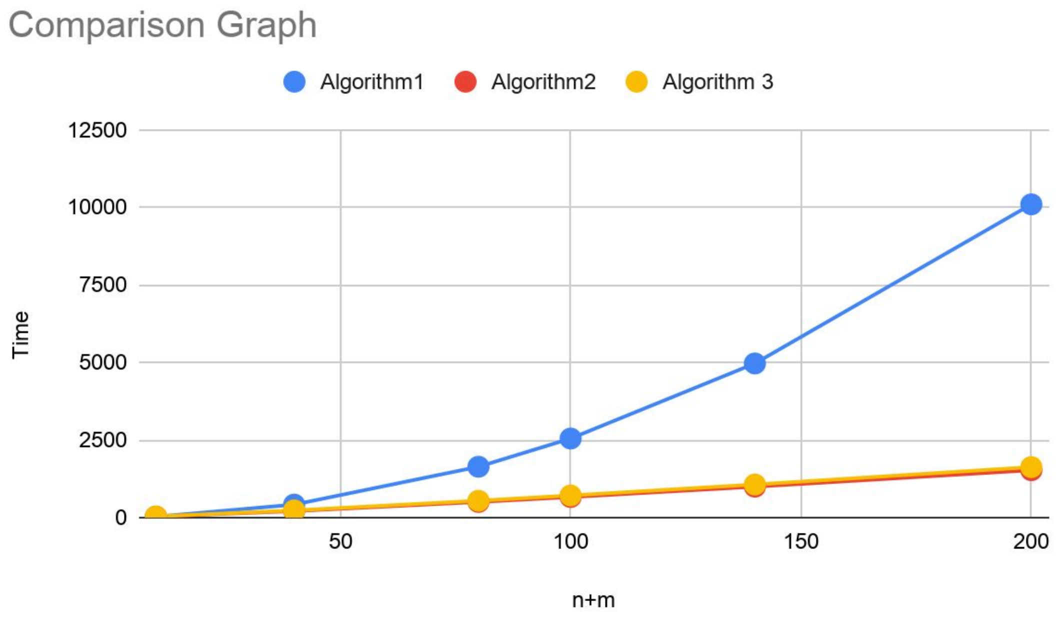
Figure 3:Time Complexity of Algorithm3



Pseudo code

```
Void union(int n, int m, int set1[], int set2[])
If m<n
  Sort(set2[]);
Printf(set2[]);
For i=0 to n
If Binary search(set[i])==false )
      Print(set[i]);
Else
  Sort(set1[]);
Printf(set1[]);
For i=0 to n
   If Binary search(set[i])==false  )
      Print(set2[i]);
```

Comparison of time Complexity of All three Algorithm



Conclusion

After doing the analysis of different algorithms we concluded that time complexity will be minimum in brute force and union algo when both of the arrays will be the same and for the worst case it will be minimum in Searching and Sorting Algorithm.

References:

https://www.geeksforgeeks.org/union-and-intersection-of-two-sorted-arrays-2/

https://www.google.com/search?q=Ringsum+operation+on+two+sets+of+positive+integers&oq=Ringsum+oper&aqs=chrome.1.69i57j69i59l2.7684j0j7&sourceid=chrome&ie=UTF-8