COL215 - Software Assignment 3

Name : Ujjwal Mehta | Entry No. : 2020CS10401

Name : Somaditya Singh | Entry No. : 2020CS10389

Description :

In this assignment we have created our own function to minimize the sum of products expression for a given kmap truth and don't care terms. In order to do this task, we have reused the code for comb_function_expansion from the previous assignment to expand a given '1' term for the corresponding kmap.

Explanation of Algorithm:

In order to minimize the sum of products expression for a given kmap which is done in the **opt_function_reduce function** , we have used the following algorithm i.e. first we generate the list of expanded terms with boolean term '1' as we did in software assignment 2 using the **comb_function_expansion** and then we create a frequency map named **terms_freq** where we store the boolean term '1' as key and the corresponding value as the number of times that particular term is covered in all the regions. Then we iterate over all the maximally expanded terms and then we check if removing that particular term will make the frequency of some '1' boolean term as 0, and if that is the case then we will store that term in our final minimized sum of terms else we will ignore it. Finally we will get our answer and then we will return it.

Analysis of Time Complexity:

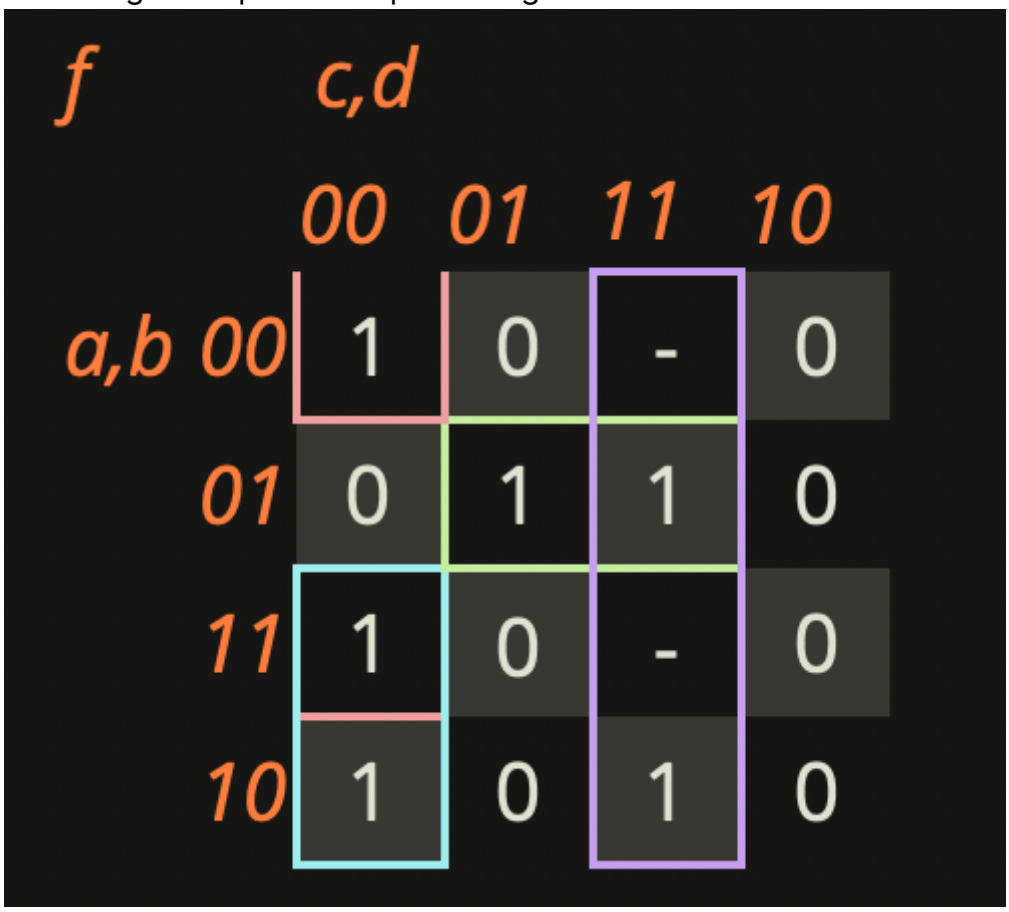
If we do the step by step analysis of our given algorithm then we can say that the overall time taken for our algorithm is the sum of times to first compute **comb_function_expansion** for the given kmap followed by the time to iterate these maximal terms and checking if they are essential in our region, hence we can say that our total time is given by :

```
Time Taken = Time of comb_function_expansion + O(n^2)[Time to iterate and
check maximal term]

Time Taken = O(n^2) + O(n^2) [time of getting maximal expansion terms is
O(n^2)]

=> Hence we can say that our total time complexity is:
Time Taken = O(n^2) where n is the combined size of Func_TRUE and Func_DC
```

Testing Strategy:

We used the following test cases to validate our code along with the sample test cases given in the assignment.

1. **Test Case 1**
   The image of input with optimal region is as follows:

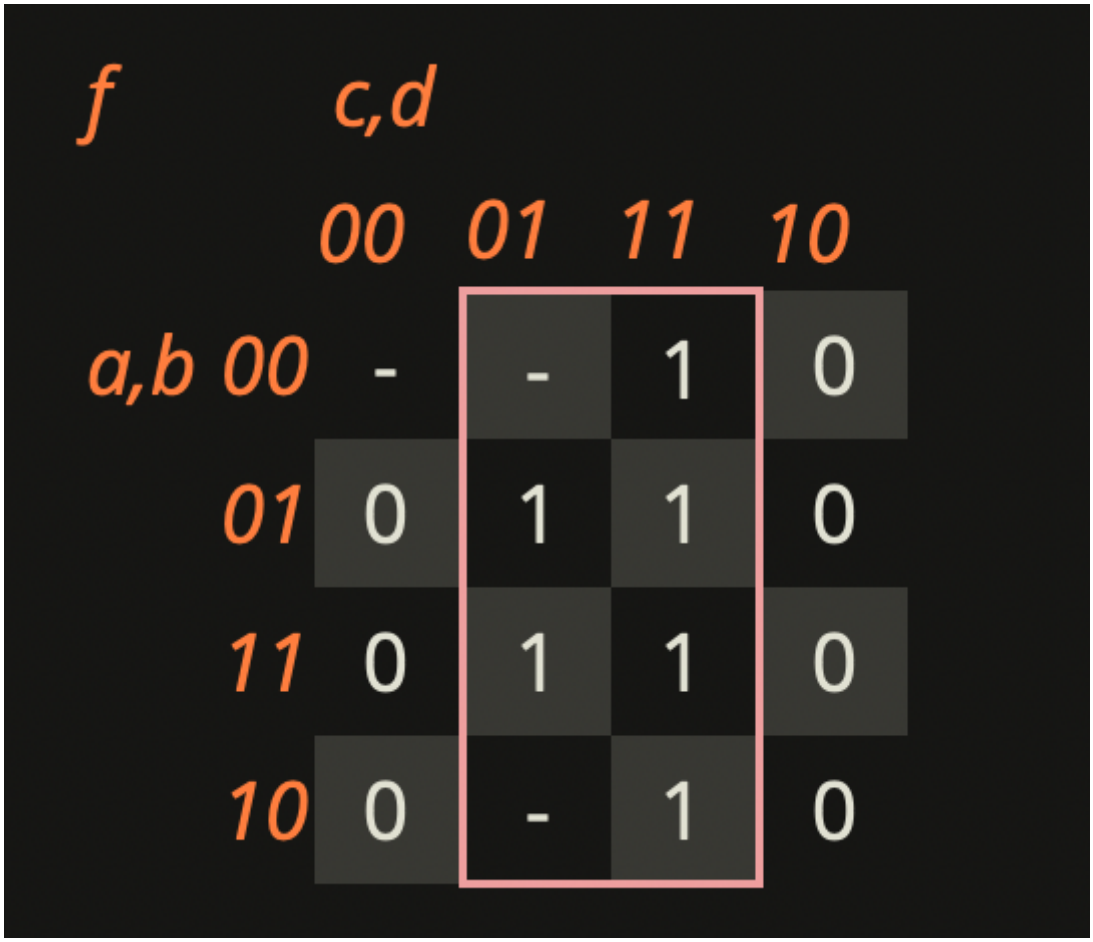


   Here the output of our code is :

```
["a'bd", 'cd', "ac'd'", "b'c'd'"]
```

And we can say that it consistent with given kmap since it contains the minimized sum of terms covering all '1'.

2. **Test Case 2**
   The image input with optimal region is as follows:



Here the output of our code is:

```
['d']
```

And we can say that here also it is consistent with given kmap since minimized term list size is 1 and it is covering all the '1' terms.

3. **Test Case 3**
   The image input with optimal region is as follows:

Here the output of our code is:

```
["c'd'e'", "cd'e", "be'"]
```

In this case also we getting minimized list size and all the '1' boolean terms are getting covered in these regions.

These cases validate that the `don't care` terms are appropriately chosen by our algorithm to minimise the logic. Also no term that was chosen by our algorithm had literals that were contained in other selected terms, even though there were multiple overlaps of the regions having all `1`, the way these tests were created.