Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Submitted To :

Professor GOVINDHA.K
Data Structures and Algorithms
CSE2003 Review 3

Submitted By :

P.VAMSI KRISHNA (19BEC0841)

L57+58

BIN PACKING

Analysis Of Worst-Fit Heuristics

ABSTRACT :

This study examined the efficiency of worst fit algorithms in bin packing scenario. We have first studied the bin packing problem statement and then have done research on the different types of algorithms available for the bin packing problem. The best has always been regarded as the Best Fit Algorithm. But it was found that the perfect algorithm depends on the situation in which it is used. We put the entire focus on the Worst Fit Algorithm and its types. We have examined three types of algorithms in the worst fit scenario – Worst Fit Algorithm, Worst Fit Decreasing Algorithm, Almost Worst Fit Algorithm. These algorithms have first been studied and then been coded in terms of memory management. After that same input to all the three algorithms have been given and efficiency of one over the other is tested. Hundreds of sample inputs have been The data obtained has been graphed and the variations are observed depending on variation in input box sizes.Worst fit requires searching the entire free list at each memory request which is possibly done by priority queue implementation.

AIM :

The aim of this project is to prove the efficiency of different types of worst fit algorithms in Bin packing and also to conclude which is the most efficient one.The required data structure in c language is used in the algorithm.we also provide a comparison graph.

Objectives :

The objectives of this project are as follows :

- Study Of different types of Worst fit Algorithms.
- Developing an algorithms for different types.
- Comparing the different algorithms with same input.
- Developing a flow chart.
- Developing comparison graphs.

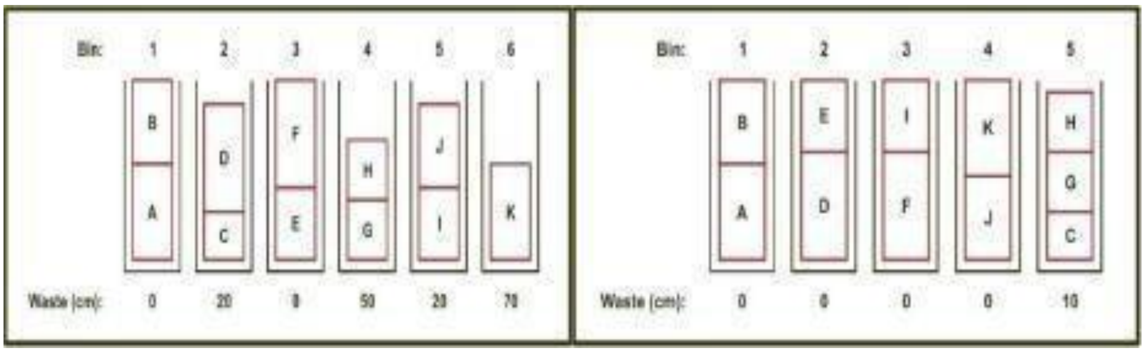
Applicability :

- There are many applications where one can put the worst fit algorithm to use.
- Worst fit algorithms sorts things in such a way that we obtain almost an equal distribution of weights in each bin or whatever it is.
- Although Best fit mostly predominant, worst fit also occupies it’s decent position when it comes to balancing by equal distribution in the memory.
- Real life examples using worst fit :

- ★ Packing of luggage in a bag.we need equal weight in order to maintain balance while picking it up and also while rolling it on the ground.Hence, worst fit is pretty suitable there.
- ★ When it comes to things like container loading and packaging of valuable items in transportation also find the worst fit most suitable and helps in increasing reliability.
- ★ One of the craziest applications that we could think of is usage of this in online sites for shopping.
- ★ When we are putting in our first search in the website without any previous searches or filters, the site doesn't have any option but to display all the the things related to the search giving equal weightage to each and everything.
- ★ These are all the places exactly where worst fit Algorithm steps in and finds it's place.

<u>Introduction</u>:

Given a positive integer number of bins m of capacity W and a list of n items of integer sizes w1, w2 wn (0 ≤ wi ≤ W), the problem is to assign the items to the bins so that the capacity of the bins is not exceeded and the number of bins used is minimized.The bin packing problem is to pack arbitrary sized and heterogeneous bins into a container leaving as little empty space as possible. Packing of bins into container have remained as a problem whose solution could be improved by devising new approach which consider major constraints namely boundary crossing constraint, overlapping constraint, stability constraint, orientation constraint, weight constraint and load bearing constraint. 42 In this research work, a GA has been used to solve the 3D bin packing problem. The genetic fitness function is used to minimize the unused space inside the container. In this work, various genetic parameters are analyzed to produce optimal solution.



Fig(1.1)

Alternate Methods and draw backs :

Usually, for Bin packing we try to minimize the number of bins used or we can also say it as the maximum size or total size of the accepted items. The problem also consists of the NP-hard problem. The present system uses mostly best fit and also in some of the cases first fit.
When we use the first fit we will get to a problem of The maximum resource bin packing problem, here the number of bins used were maximized, which is completely opposite of the classical bin packing problem.

The other method used is, Multi-dimensional Bin packing problems with Guillotine Constraints. In this method, it determines a problem like, if a set of multi-dimentional rectangular boxes can be orthogonally packed into a rectangular bin while satisfying the requirement the packing should be guillotine cuttable. Here, the unrestricted problem is known to be NP-hard.

The other method is using a Hybrid P-system and CUDA
Architecture, which is used for Heuristic for the variable sized bin packing problem. It reduces the complexity of implementing the psystem and also the number of bins used but it fails in cases of short data.

There are many different types of methods used to solve this problem and the problem NP-hard was not fully satisfied.

Proposed Model :

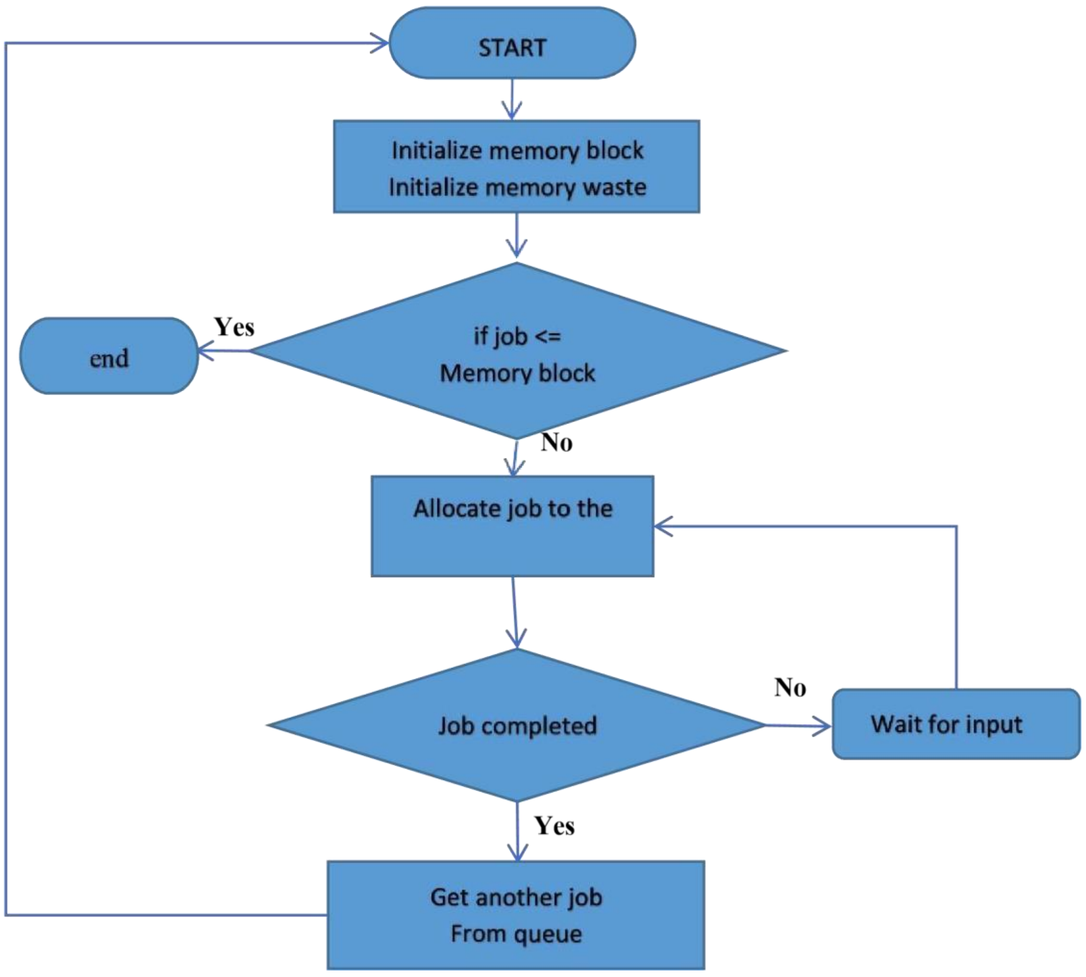
- This algorithm is not widely used , but comes into play in the cases where the best fit fails.
- Our model will be using the different kinds of worst fit Algorithms.
- Worst fit, Almost worst fit and worst fit decreasing are the different Algorithms used.
- While using the worst fit it skips the largest box in all the cases and manages to get the most efficient distribution.
- It uses more number of bins with equal distribution.
- Occupancy of space is equal for all the bins.

- The space will not be wasted much when it comes into the play.
- The advantage in using this is, it may waste the individual space of the bin, but saves the overall space.
- It's not very much complex and it is also the modification of best fit itself.
- It can be as an alternative for some cases which uses the best fit and where best fit can't work efficiently.

Proposed System Architecture :

Here we are comparing three different algorithms, but for each algorithm the basic concept is same and which is defined using the following flow chart.

FLOW CHART



Fig(1.2)

Implementation Details :

We will be implementing the three different algorithms where the user is asked for the required inputs. The process is as follows

After executing the program the user is asked for

- Enter the number of blocks
- Enter the number of files

And after this, it will ask the user to define the

- Size of the each block
- Size of the each file

- After all the inputs were provided the, based on the algorithm it provides us an output

Testing of Algorithm and Output :

1. **WORST FIT :**

```
#include<stdio.h>
#include<conio.h> #define max 25 void
main() {
int frag[max],b[max],f[max],i,j,nb,nf,temp,highest=0;
static int bf[max],ff[max]; printf("Worst Fit\n"); printf("Enter the number of
blocks:"); scanf("%d",&nb);
printf("Enter the number of files:"); scanf("%d",&nf); printf("Enter the size of
blocks:-\n"); for(i=1;i<=nb;i++) { printf("Block%d:",i); scanf("%d",&b[i]);
}
printf("Enter the size of files:-\n"); for(i=1;i<=nf;i++)
{ printf("File%d:",i); scanf("%d",&f[i]);
}
for(i=1;i<=nf;i++) { for(j=1;j<=nf;j++)
        { if(bf[j]!=1)
                { temp=b[j]-f[i]; if(temp>=0) if(highest<temp
                        )
                        { ff[i]=j;
                                        highest=temp;
                        }
                } } frag[i]=highest; bf[ff[i]]=1; highest=0;
}
printf("\nFile_no:\tFile_size\tBlock_no:\tBlock_size\tFragment"); for(i=1;i<=nf;i++)
printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d",i,f[i],ff[i],b[ff[i]],frag[i]); getch();
}
```

2. **Worst Fit Decreasing :**

```
#include<stdio.h>
#include<conio.h> #define max 25 void
main()
{
int frag[max],b[max],f[max],i,j,nb,nf,temp,highest=0; static int
bf[max],ff[max];
printf("\n\tWorst Fit Decreasing"); printf("\nEnter  the
        number of blocks:"); scanf("%d",&nb);
        printf("Enter the number of files:"); scanf("%d",&nf);  printf("\nEnter the size of
        theblocks:-\n");  for(i=1;i<=nb;i++)
        { printf("Block %d:",i);  scanf("%d",&b[i]);

        }
        printf("Enter the size of the files :-\n");   for(i=1;i<=nf;i++)
        { printf("File %d:",i); scanf("%d",&f[i]);

        }
        for(i=1;i<=nf;i++)
        {
           for(j=i+1;j<nf+1;j++)
           { if(f[i]<f[j])
                 { temp=f[i]; f[i]=f[j];  f[j]=temp
                       ;

                 }
           }
        }
        for(i=1;i<=nf;i++)  { for(j=i;j<=nb;j++)
           { if(bf[j]!=1)  //if bf[j] is not allocated
                 { temp=b[j]-f[i]; if(temp>=0) if(highest<temp
                       )
                       { ff[i]=j;
                       highest=temp;


                       }  }
           } frag[i]=highest; bf[ff[i]]=1;  highest=0;

     }
     printf("\nFile_no:\tFile_size\tBlock_no:\tBlock_size\tFragment");  for(i=1;i<=nf;i++)
     printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d",i,f[i],ff[i],b[ff[i]],frag[i]);  getch();  }
```

3. <u>Almost Worst Fit</u> :

```
#include<stdio.h>
#include<conio.h> #define max 25
void main()
```

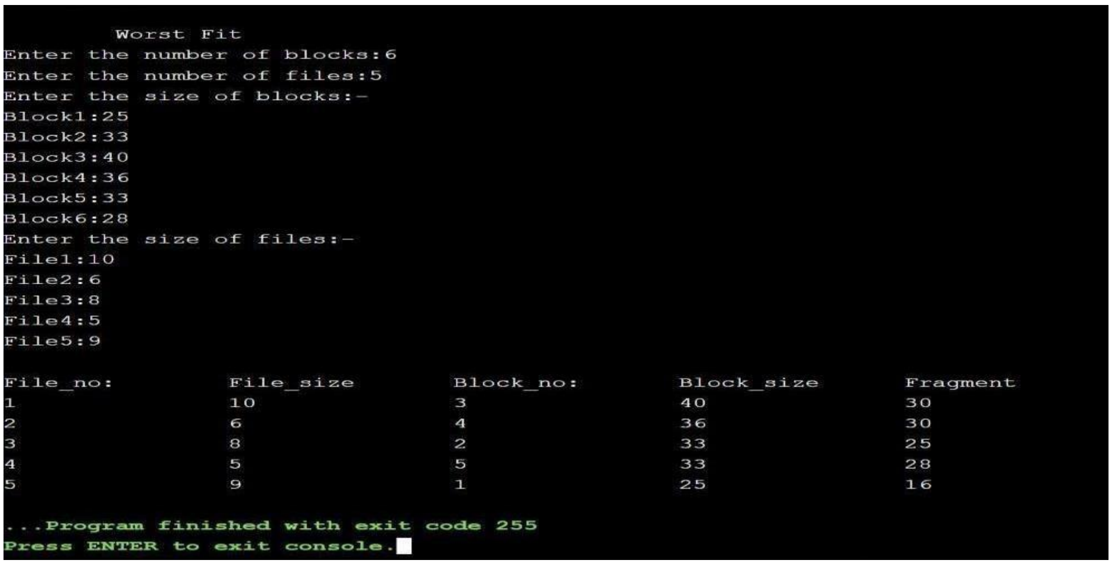
```
{ int frag[max],b[max],f[max],i,j,nb,nf,temp,highest=0; static int
        bf[max],ff[max]; printf("\n\tAlmost Worst Fit"); printf("\nEnter the
        number of blocks :"); scanf("%d",&nb);
        printf("Enter the number of files :"); scanf("%d",&nf); printf("\nEnter the
        size of the blocks:\n");
        for(i=1;i<=nb;i++)
        { printf("Block %d:",i); scanf("%d",&b[i]);
        }
        printf("\nEnter the size of the files:-\n"); for(i=1;i<=nf;i++)
        { printf("File%d:",i); scanf("%d",&f[i]);
        }
        for(i=1;i<=nb;++i) {
        for(j=i+1;j<=nb;++j) { if(b[i]<b[j]){ int
        a=b[i]; b[i]=b[j]; b[j]=a;
                }
            } } for(i=1;i<=nf;i++)
        {
            for(j=2;j<=nb;j++)
            { if(bf[j]!=1)
                { temp=b[j]-f[i]; if(temp>=0){
                    ff[i]=j; highest=temp;
                    frag[i]=highest; bf[ff[i]]=1; highest=0; break; } else{
                    if(bf[1]!=1){ if((b[1]-f[i]>0)){ ff[i]=1; highest=(b[1]f[i]);
                    bf[1]=1; frag[i]=highest; bf[ff[i]]=1; highest=0; break;
                                    }
                        }
                    }
                }
            } } printf("\nFile_no:\tFile_size:\tBlock_size:\tFragment")
        ; for(i=1;i<=nf;i++) printf("\n%d\t\t%d\t\t%d\t\t%d",i,f[i],b[ff[i]],frag[i]);
        getch();
}
```

○ Sample that we used for testing is the following

The main aim is to compare the variants of the Worst Fit Heuristics. So the code is put into execution and the results are compared. The inputs fed to the code were: 10, 6, 8, 5 and 9 are the boxes sizes that should be filled and the bins size inputs are 25, 33, 40, 36, 33 and 28. And the second input set taken for the boxes is 105, 95, 70, 90, 85, 110, 80 and 75. Sizes of the bins are: 100, 120, 135, 115, 125, 105, 140, 110 and 130.

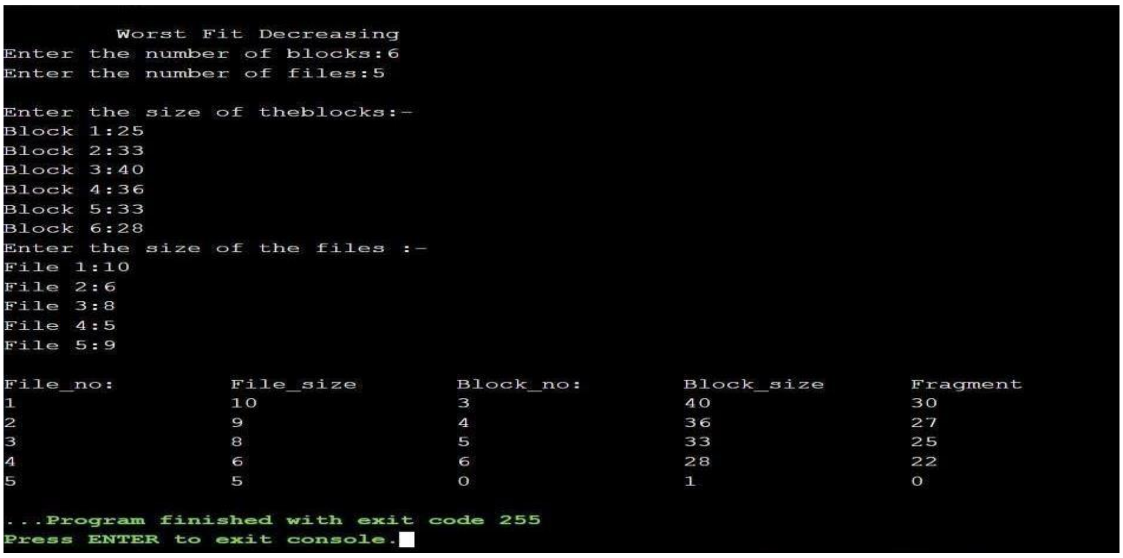
These are two of the many samples that we tested to make the conclusions stated below

Outputs and comparison graph for the first set of inputs :



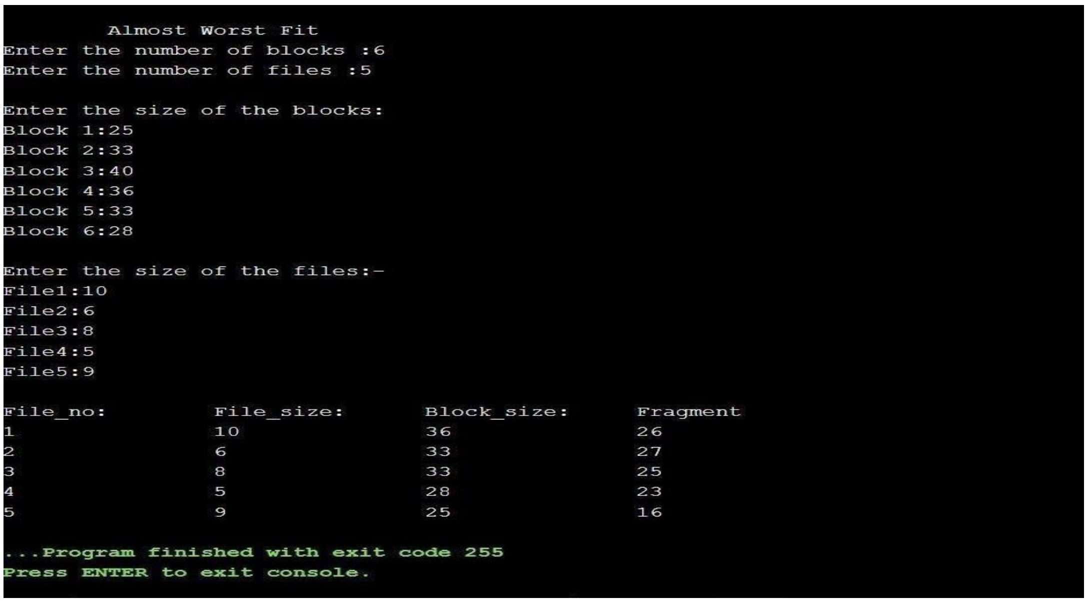
```
        Worst Fit
Enter the number of blocks:6
Enter the number of files:5
Enter the size of blocks:-
Block1:25
Block2:33
Block3:40
Block4:36
Block5:33
Block6:28
Enter the size of files:-
File1:10
File2:6
File3:8
File4:5
File5:9

File_no:        File_size       Block_no:       Block_size      Fragment
1               10              3               40              30
2               6               4               36              30
3               8               2               33              25
4               5               5               33              28
5               9               1               25              16

...Program finished with exit code 255
Press ENTER to exit console.
```

Fig(a)-Worst fit



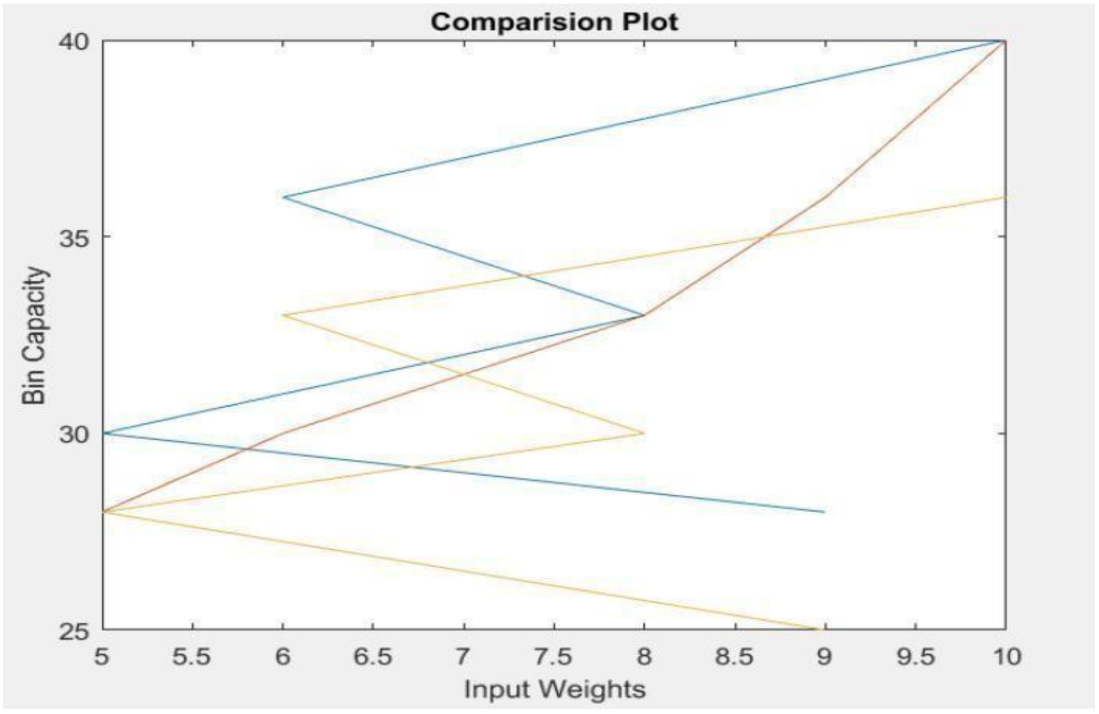
```
        Worst Fit Decreasing
Enter the number of blocks:6
Enter the number of files:5

Enter the size of theblocks:-
Block 1:25
Block 2:33
Block 3:40
Block 4:36
Block 5:33
Block 6:28
Enter the size of the files :-
File 1:10
File 2:6
File 3:8
File 4:5
File 5:9

File_no:        File_size       Block_no:       Block_size      Fragment
1               10              3               40              30
2               9               4               36              27
3               8               5               33              25
4               6               6               28              22
5               5               0               1               0

...Program finished with exit code 255
Press ENTER to exit console.
```

Fig(b)-Worst fit Decreasing



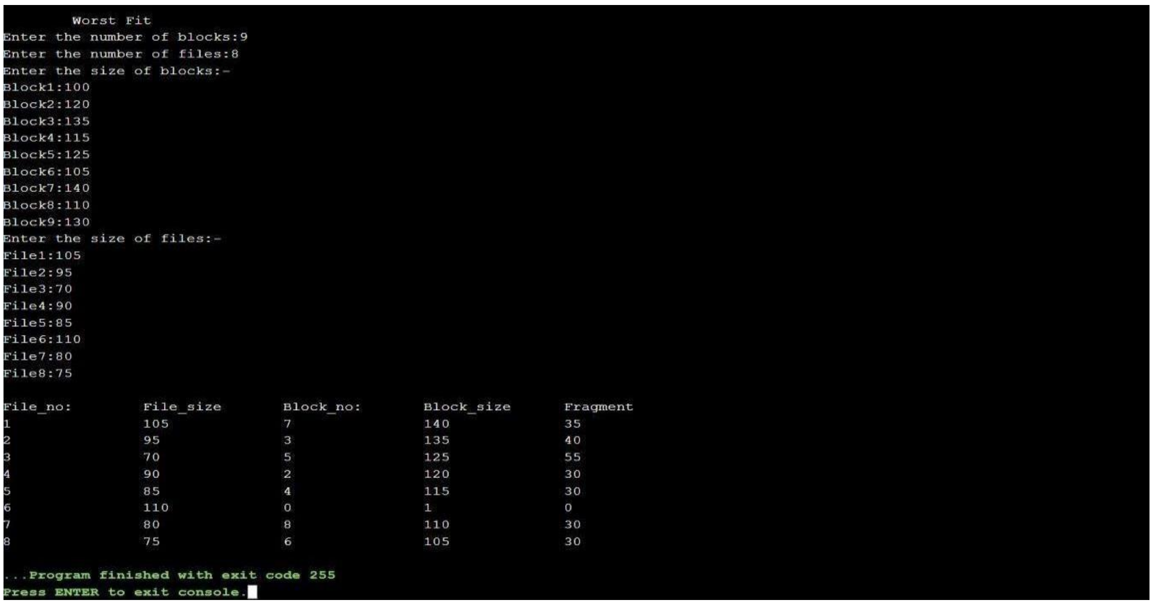
```
        Almost Worst Fit
Enter the number of blocks :6
Enter the number of files :5

Enter the size of the blocks:
Block 1:25
Block 2:33
Block 3:40
Block 4:36
Block 5:33
Block 6:28

Enter the size of the files:-
File1:10
File2:6
File3:8
File4:5
File5:9

File_no:        File_size:      Block_size:     Fragment
1               10              36              26
2               6               33              27
3               8               33              25
4               5               28              23
5               9               25              16

...Program finished with exit code 255
Press ENTER to exit console.
```

Fig(c)-Almost Worst fit



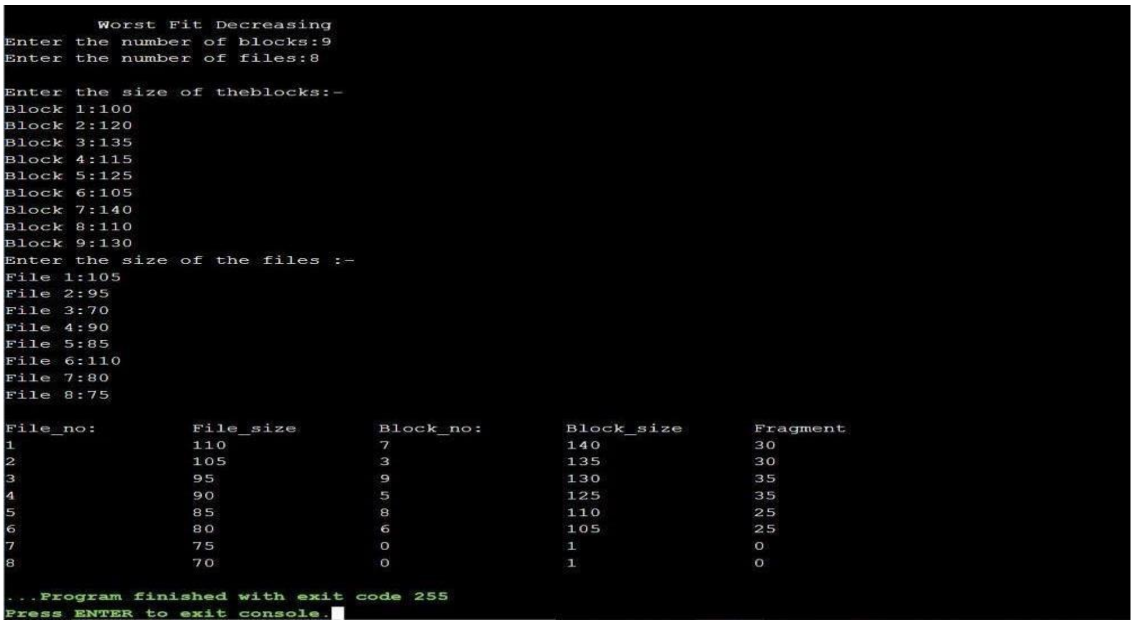
Fig(1.3)

∘ Output and comparison graphs for 2nd set of inputs



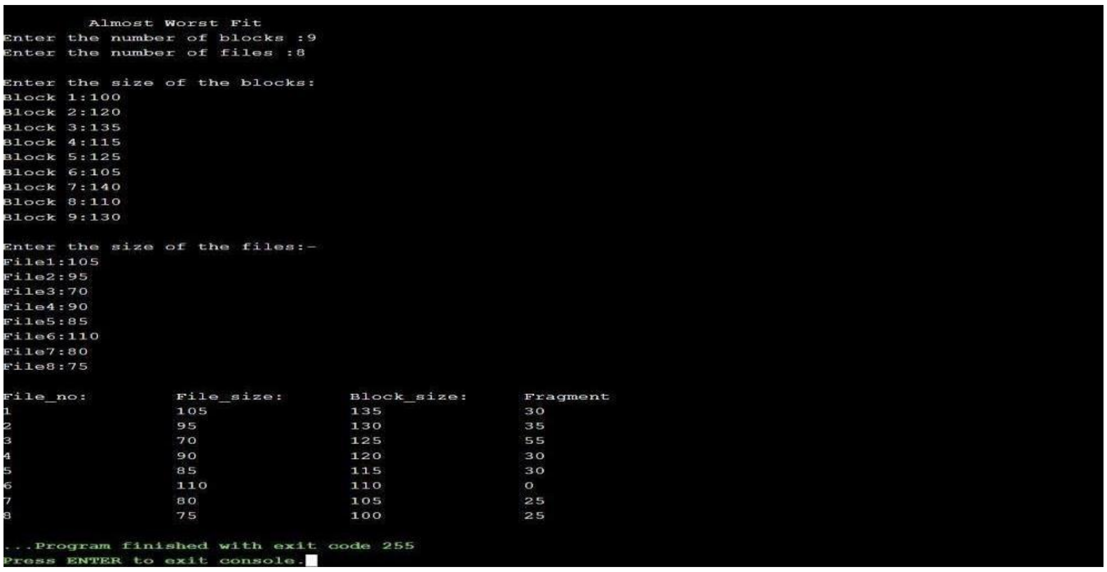
```
        Worst Fit
Enter the number of blocks:9
Enter the number of files:8
Enter the size of blocks:-
Block1:100
Block2:120
Block3:135
Block4:115
Block5:125
Block6:105
Block7:140
Block8:110
Block9:130
Enter the size of files:-
File1:105
File2:95
File3:70
File4:90
File5:85
File6:110
File7:80
File8:75

File_no:        File_size       Block_no:       Block_size      Fragment
1               105             7               140             35
2               95              3               135             40
3               70              5               125             55
4               90              2               120             30
5               85              4               115             30
6               110             0               1               0
7               80              8               110             30
8               75              6               105             30

...Program finished with exit code 255
Press ENTER to exit console.
```

Fig(d)-Worst fit



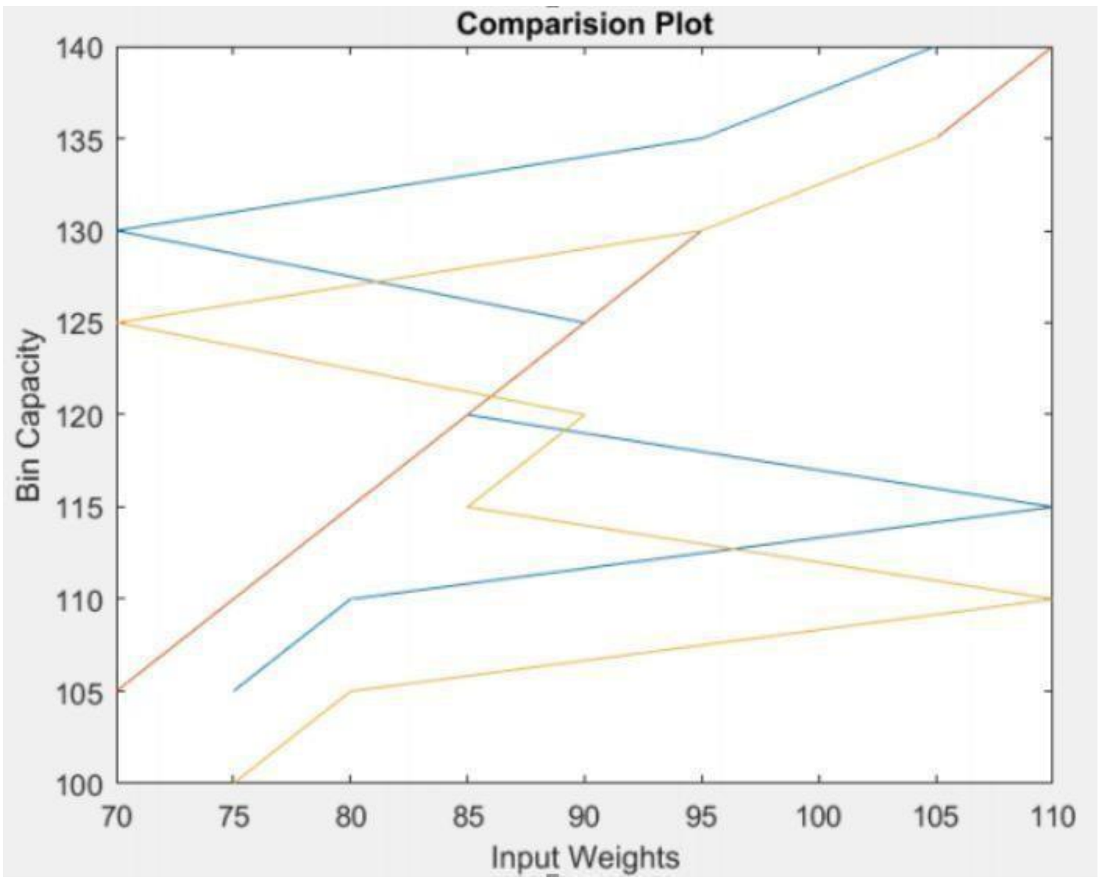
```
        Worst Fit Decreasing
Enter the number of blocks:9
Enter the number of files:8

Enter the size of theblocks:-
Block 1:100
Block 2:120
Block 3:135
Block 4:115
Block 5:125
Block 6:105
Block 7:140
Block 8:110
Block 9:130
Enter the size of the files :-
File 1:105
File 2:95
File 3:70
File 4:90
File 5:85
File 6:110
File 7:80
File 8:75

File_no:        File_size       Block_no:       Block_size      Fragment
1               110             7               140             30
2               105             3               135             30
3               95              9               130             35
4               90              5               125             35
5               85              8               110             25
6               80              6               105             25
7               75              0               1               0
8               70              0               1               0

...Program finished with exit code 255
Press ENTER to exit console.
```

Fig(e)-Worst fit decreasing



```
        Almost Worst Fit
Enter the number of blocks :9
Enter the number of files :8

Enter the size of the blocks:
Block 1:100
Block 2:120
Block 3:135
Block 4:115
Block 5:125
Block 6:105
Block 7:140
Block 8:110
Block 9:130

Enter the size of the files:-
File1:105
File2:95
File3:70
File4:90
File5:85
File6:110
File7:80
File8:75

File_no:        File_size:      Block_size:     Fragment
1               105             135             30
2               95              130             35
3               70              125             55
4               90              120             30
5               85              115             30
6               110             110             0
7               80              105             25
8               75              100             25

...Program finished with exit code 255
Press ENTER to exit console.
```

Fig(f)-Almost Worst fit



Fig(1.4)

<u>Comparing between the Algorithms</u> :

<u>In case I:</u>

Worst Fit is better than Worst Fit Decreasing by 2.30938% Almost Worst Fit is better than Worst Fit by 9.75513%

Almost Worst Fit is better that Worst Fit Decreasing by 12.0577%

<u>In Case II</u>*:*

Worst Fit is better than Worst Fit Decreasing by 0.895049% Almost Worst Fit is better than Worst Fit by 4.22275%

Almost Worst Fit is better than Worst Fit Decreasing by 5.11731%

<u>Conclusion</u> :

From the observation, it is clear that for the sample input data mentioned here, **"Almost worst fit is the best algorithm and is the most efficient one"**. It occupies the most space in all the bins compared to the remaining scenarios and hence is effective. This is because the inputs that we have given for the bins are much higher than the box sizes that we mentioned. Hence when almost worst fit is used, it skips the largest box in all the cases and manages to get the most efficient distribution. In case if some of the inputs are given equal to or higher that of the bins size, then almost worst fit case would be helpless. In that case, worst fit decreasing and worst fit algorithms will be the best ones to be used. In case of worst fit decreasing algorithm, if the inputs are given in such a way that most of them match the sizes of the bins, then worst fit decreasing algorithm will prove to be the best one. This is because while searching for the emptiest bin, the algorithm kind of arranges the bins in decreasing order of bin sizes. Also the boxes sizes is also arranged in decreasing order in worst fit decreasing scenario. Hence if the values match, then this algorithm would be perfect one to be applied. In all other cases, the normal worst fit algorithm would be the best suited.

<u>References</u> :

1) bitstreams.com
2) Wikipidea
3) Google
4) Korte, Bernhard; Vygen, Jens (2006).