Name: Shawn Louis

SE COMPS

Roll No.: 31

Experiment no 10

Aim : Write a program to implement dynamic partitioning placement algorithms - Best Fit algorithm.

Theory :

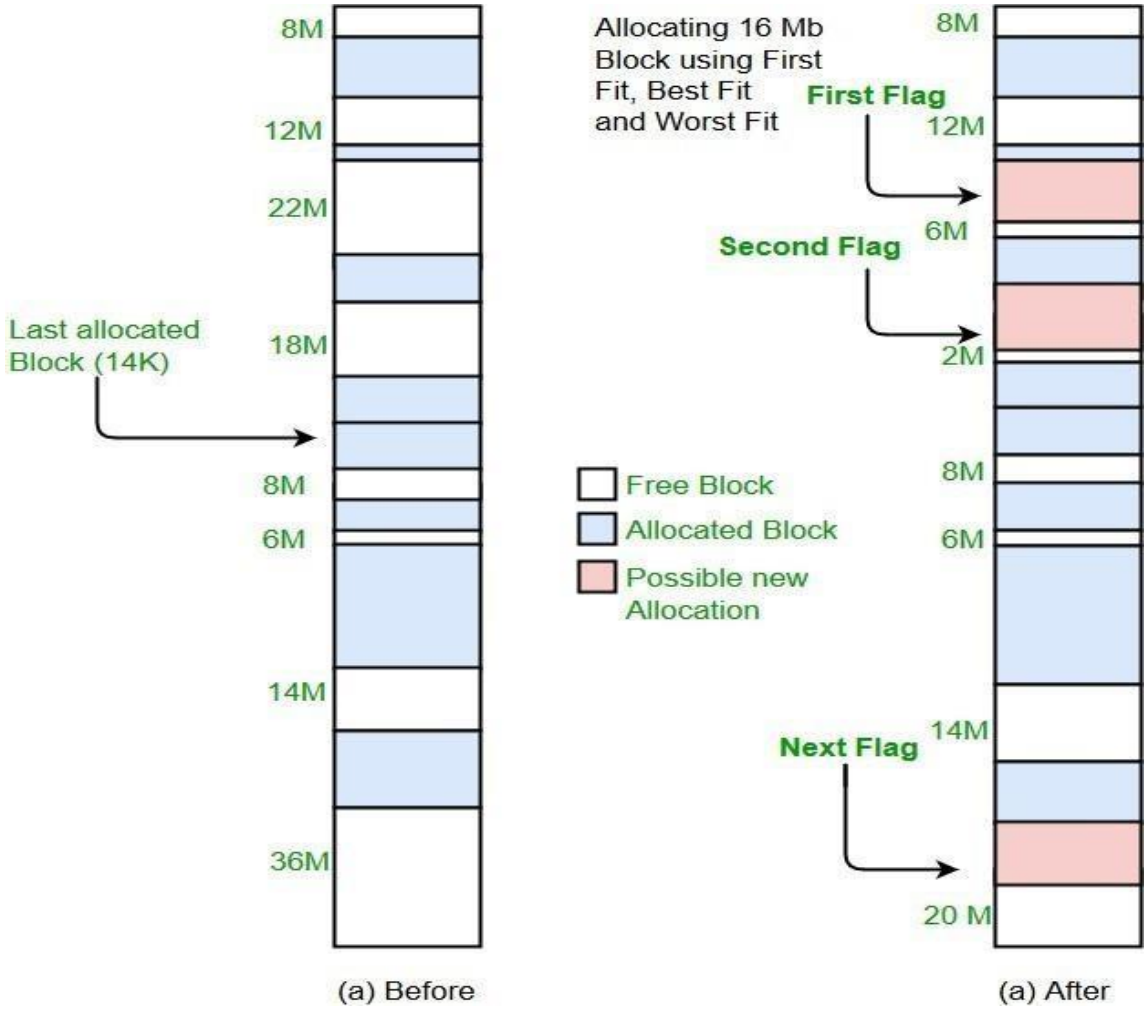
The best fit deals with allocating the smallest free partition which meets the requirement of the requesting process. This algorithm first searches the entire list of free partitions and considers the smallest hole that is adequate. It then tries to find a hole which is close to actual process size needed.

Advantage

Memory utilization is much better than first fit as it searches the smallest free partition first available.

Disadvantage

It is slower and may even tend to fill up memory with tiny useless holes.



Algorithm :
```
1- Input memory blocks and processes with sizes.
2- Initialize all memory blocks as free.
3- Start by picking each process and find the
   minimum block size that can be assigned to
   current process i.e., find min(bockSize[1],
   blockSize[2],.....blockSize[n]) >
   processSize[current], if found then assign
   it to the current process.
5- If not then leave that process and keep checking
   the further processes.
```

Output :

```
#include<stdio.h>

void main()
 {
    int fragment[20],b[20],p[20],i,j,nb,np,temp,lowest=9999;
    static int barray[20],parray[20];

    printf("\n\t\t\tMemory Management Scheme - Best Fit");
    printf("\nEnter the number of blocks:");
    scanf("%d",&nb);
    printf("Enter the number of processes:");
    scanf("%d",&np);

    printf("\nEnter the size of the blocks:-\n");
    for(i=1;i<=nb;i++)
    {
       printf("Block no.%d:",i);
       scanf("%d",&b[i]);
    }

    printf("\nEnter the size of the processes :-\n");
    for(i=1;i<=np;i++)
    {
       printf("Process no.%d:",i);
       scanf("%d",&p[i]);
    }

    for(i=1;i<=np;i++)
    {
       for(j=1;j<=nb;j++)
       {
          if(barray[j]!=1)
          {
             temp=b[j]-p[i];
             if(temp>=0)
                if(lowest>temp)
                {
                   parray[i]=j;
                   lowest=temp;
```

```
            }
        }
    }

    fragment[i]=lowest;
    barray[parray[i]]=1;
    lowest=10000;
  }

  printf("\nProcess_no\tProcess_size\tBlock_no\tBlock_size\tFragment");
  for(i=1;i<=np && parray[i]!=0;i++)
    printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d \n",i,p[i],parray[i],b[parray[i]],fragment[i]);
}
```

Conclusion :

Thus, best fit algorithm was implemented in C language and its advantages and disadvantages were pondered upon. Memory utilization, speed and time utilization were taken into consideration and concluded that this algorithm is better when compared to first fit algorithm.