**Exp No.** 12 **Name:** PONNURI ANIRUDDHA

**Date**: 18/10/2022 **Reg No:** RA2112704010015

**Best fit and Worst fit memory management policies**

**Aim: --**

To implement Best fit and Worst fit memory management policies

**Procedure: --**

**BEST FIT MEMORY MANAGEMENT: --**

This method keeps the free/busy list in order by size – smallest to largest. In this method, the operating system first searches the whole of the memory according to the size of the given job and allocates it to the closest-fitting free partition in the memory, making it able to use memory efficiently. Here the jobs are in the order from smallest job to largest job.

Advantages of Best-Fit Allocation:   
Memory Efficient. The operating system allocates the job minimum possible space in the memory, making memory management very efficient. To save memory from getting wasted, it is the best method.

Best-Fit Allocation Benefits:   
It is a Slow Process. Checking the whole memory for each job makes the working of the operating system very slow. It takes a lot of time to complete the work.

**WORST FIT MEMORY MANAGEMENT: --**

In this allocation method, the process scans the entire memory, looking for the biggest hole or partition, and then it is allocated to that hole or partition. In order to find the biggest hole, the method must go over the entire memory, which takes time.

Worst-Fit Allocation Benefits

There will be significant internal fragmentation because this process selects the greatest hole or fragment. This internal fragmentation will now be sufficiently large to allow for the placement of additional smaller processes in the unused partition.

Worst-Fit Allocation Drawbacks:

It is a slow procedure since it first navigates across every memory partition before choosing the largest one, which takes a long time.

**CODE (BEST FIT): --**

*def bestFit(blockSize, m, processSize, n):*

*# Stores block id of the block allocated to a process*

*allocation = [-1] \* n*

*for i in range(n):*

*# Find the best fit block for current process*

*bestIdx = -1*

*for j in range(m):*

*if blockSize[j] >= processSize[i]:*

*if bestIdx == -1:*

*bestIdx = j*

*elif blockSize[bestIdx] > blockSize[j]:*

*bestIdx = j*

*# If we could find a block for*

*# current process*

*if bestIdx != -1:*

*# allocate block j to p[i] process*

*allocation[i] = bestIdx*

*# Reduce available memory in this block.*

*blockSize[bestIdx] -= processSize[i]*

*print("Process No. Process Size Block no.")*

*for i in range(n):*

*print(i + 1, " ", processSize[i],end = " ")*

*if allocation[i] != -1:*

*print(allocation[i] + 1)*

*else:*

*print("Not Allocated")*

*# Driver code*

*if \_\_name\_\_ == '\_\_main\_\_':*

*blockSize = []*

*processSize = []*

*m = int(input("Enter Number of memory holes:--"))*

*n = int(input("Enter Number of Processess:--"))*

*for i in range(m):*

*x=int(input("Enter memory hole size:--- "))*

*blockSize.append(x)*

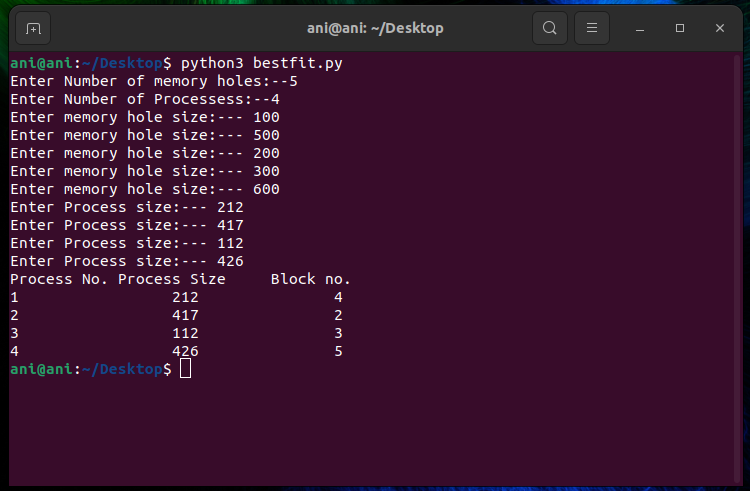
*for i in range(n):*

*x=int(input("Enter Process size:--- "))*

*processSize.append(x)*

*bestFit(blockSize, m, processSize, n)*

**OUTPUT (BEST FIT): --**

****

**CODE (WORST FIT): --**

*def worstFit(blockSize, m, processSize, n):*

*# Stores block id of the block allocated to a process*

*# Initially no block is assigned*

*# to any process*

*allocation = [-1] \* n*

*# pick each process and find suitable blocks*

*# according to its size ad assign to it*

*for i in range(n):*

*# Find the best fit block for*

*# current process*

*wstIdx = -1*

*for j in range(m):*

*if blockSize[j] >= processSize[i]:*

*if wstIdx == -1:*

*wstIdx = j*

*elif blockSize[wstIdx] < blockSize[j]:*

*wstIdx = j*

*# If we could find a block for*

*# current process*

*if wstIdx != -1:*

*# allocate block j to p[i] process*

*allocation[i] = wstIdx*

*# Reduce available memory in this block.*

*blockSize[wstIdx] -= processSize[i]*

*print("Process No. Process Size Block no.")*

*for i in range(n):*

*print(i + 1, " ",*

*processSize[i], end=" ")*

*if allocation[i] != -1:*

*print(allocation[i] + 1)*

*else:*

*print("Not Allocated")*

*# Driver code*

*if \_\_name\_\_ == '\_\_main\_\_':*

*blockSize = []*

*processSize = []*

*m = int(input("Enter Number of memory holes:--"))*

*n = int(input("Enter Number of Processess:--"))*

*for i in range(m):*

*x = int(input("Enter memory hole size:--- "))*

*blockSize.append(x)*

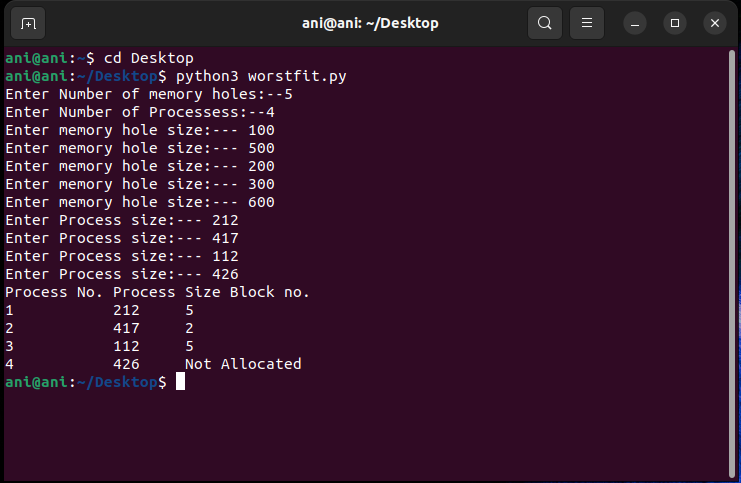
*for i in range(n):*

*x = int(input("Enter Process size:--- "))*

*processSize.append(x)*

*worstFit(blockSize, m, processSize, n)*

**OUTPUT (WORST FIT): --**



**RESULT: --**

The best fit and worst fit memory management policies are implemented using python programming.