

**Lab no:** **Date: 2079/**

**Title: Write a program to simulate worst fit memory allocation strategy.**

**Worst Fit:**

Worst Fit allocates a process to the partition which is largest sufficient among the freely available partitions available in the main memory. If a large process comes at a later stage, then memory will not have space to accommodate it.

**Algorithm:**

Step 1: Input memory blocks and processes with sizes.

Step 2: Initialize all memory blocks as free.

Step 3: Start by picking each process and find the maximum block size that can be

assigned to current process i.e., find max(bockSize[1], blockSize[2],.....

blockSize[n]) > processSize[current], if found then assign it to the current

process.

Step 4: If not then leave that process and keep checking the further processes.

**Programming Language = C++**

**IDE = Dev-C++**

**Source Code:**

|  |
| --- |
| #include<bits/stdc++.h>  using namespace std;    // Function to allocate memory to blocks as per worst fit algorithm  void worstFit(int blockSize[], int m, int processSize[], int n)  {  // Stores block id of the block allocated to a process  int allocation[n];    // Initially no block is assigned to any process  memset(allocation, -1, sizeof(allocation));    // pick each process and find suitable blocks according to its size ad assign to it  for (int i=0; i<n; i++)  {  // Find the best fit block for current process  int p = -1;  for (int j=0; j<m; j++)  {  if (blockSize[j] >= processSize[i])  {  if (p == -1)  p = j;  else if (blockSize[p] < blockSize[j])  p = j;  }  }    // If we could find a block for current process  if (p != -1)  {    allocation[i] = p;    // Reduce available memory in this block.  blockSize[p] -= processSize[i];  }  }    cout << "\nProcess No.\tProcess Size\tBlock no.\n";  for (int i = 0; i < n; i++)  {  cout << " " << i+1 << "\t\t" << processSize[i] << "\t\t";  if (allocation[i] != -1)  cout << allocation[i] + 1;  else  cout << "Not Allocated";  cout << endl;  }  }  int main()  {  int m, n, processSize[20], blockSize[20],i;  cout<<"Enter the no of block\n";  cin>>m;  cout<<"Enter the size of each blocks\n";  for(i=0;i<m;i++)  {  cin>>blockSize[i];  }  cout<<"Enter the no. of process\n";  cin>>n;  cout<<"Enter the size of each process\n";  for(i=0;i<n;i++)  {  cin>>processSize[i];  }  worstFit(blockSize, m, processSize, n);    return 0 ;  } |

**Output:**

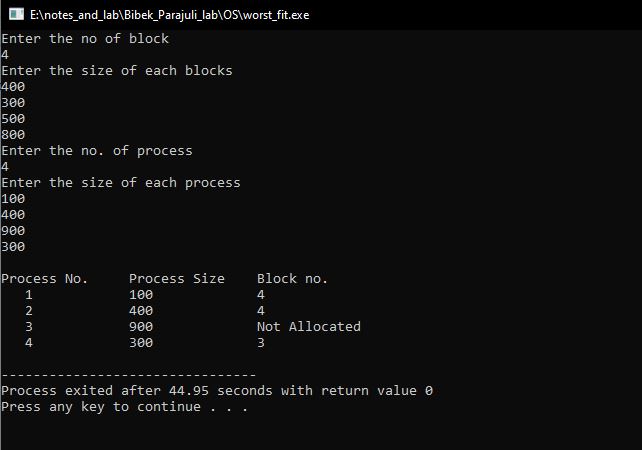
****

Fig: Worst Fit memory allocation strategy