

Lab no: Date: 2079/

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

Best Fit:

Best fit searches the entire list and takes the smallest hole that is adequate. Rather than breaking up a big hole that might be needed later, best fit tries to find a hole that is close to the actual size needed. The allocator places a process in the smallest block of unallocated memory in which it will fit.

**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 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.

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

**Source Code:**

|  |
| --- |
| //bestfit algorithm  #include<bits/stdc++.h>  using namespace std;  *// Function to allocate memory to blocks as per Best fit algorithm*  void bestFit(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));  for (int i=0; i<n; i++)  {  *// Find the best fit block for current process*  int bestIdx = -1;  for (int j=0; j<m; j++)  {  if (blockSize[j] >= processSize[i])  {  if (bestIdx == -1)  bestIdx = j;  else if (blockSize[bestIdx] > blockSize[j])  bestIdx = j;  }  }  if (bestIdx != -1)  {  *// allocate block j to p[i] process*  allocation[i] = bestIdx;  *// Reduce available memory in this block.*  blockSize[bestIdx] -= 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];  }  bestFit(blockSize, m, processSize, n);  return 0 ;  } |

**Output:**

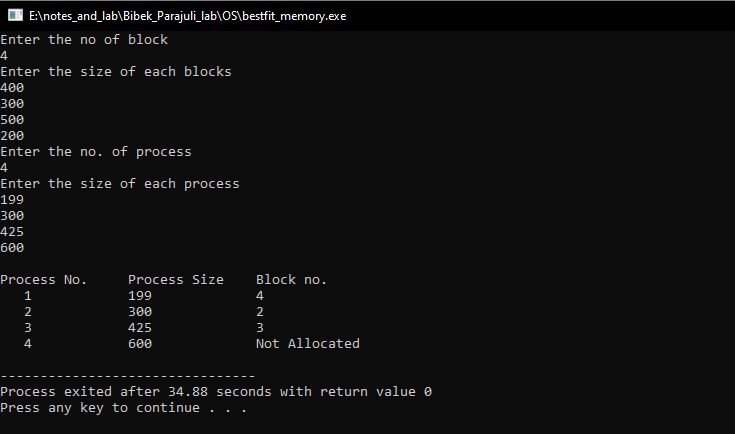


Fig: Best Fit memory allocation strategy