**Name**: Afzal Karim Talukdar

**Regd no**: 11804221

**Section**: K18PD

**Roll no**: 38

**Email**: [afzaltalukdar35@gmail.com](mailto:afzaltalukdar35@gmail.com)

**GitHub**: <https://github.com/aktalukdar/simulationofcontiguousmemoryallocation>

**Code:**

#include<unistd.h>

#include<stdio.h>

int main(){

printf("1.Best Fit\t2.First Fit\t3.Worst Fit\nEnter your choice... ");

int n;

scanf("%d",&n);

int i,j,noOfBlocks,noOfProcess;

printf("\nEnter the number of memory blocks ");

scanf("%d",&noOfBlocks);

int block[noOfBlocks],BA[noOfBlocks];//BA-->Block Allocated

for(i=0;i<noOfBlocks;i++){

printf("Enter the size of block no %d ",i+1);

scanf("%d",&block[i]);

}

printf("\nEnter the number of processes ");

scanf("%d",&noOfProcess);

int process[noOfProcess],frag[noOfProcess],PAB[noOfProcess];//PAB-->Process Assign to block

for(i=0;i<noOfProcess;i++){

printf("Enter the memory size required for process %d ",i+1);

scanf("%d",&process[i]);

}

switch(n){

case 1://Best Fit

for(i=0;i<noOfProcess;i++){

int fr,min=100000,idx;//fr-->fragment, min-->Minimum fragment,idx-->block[idx],which produce minimum fragment

for(j=0;j<noOfBlocks;j++){

if(BA[j]!=1){//If block[j] is not allocated

if(process[i]<=block[j]){

fr=block[j]-process[i];

}

if(fr<min){

min=fr;

idx=j;

}

}

}

PAB[i]=idx;

frag[i]=min;

BA[idx]=1;

min=100000;

}

break;

case 2://First Fit

for(i=0;i<noOfProcess;i++){

for(j=0;j<noOfBlocks;j++){

if(BA[j]!=1){//If block[j] is not allocated

if(process[i]<=block[j]){

PAB[i]=j;

frag[i]=block[j]-process[i];

BA[j]=1;

break;

}

}

}

}

break;

case 3://Worst Fit

for(i=0;i<noOfProcess;i++){

int fr,max=0,idx;//fr-->fragment, max-->Maximum fragment,idx-->block[idx],which produce maximum fragment

for(j=0;j<noOfBlocks;j++){

if(BA[j]!=1){//If block[j] is not allocated

if(process[i]<=block[j]){

fr=block[j]-process[i];

}

if(fr>max){

max=fr;

idx=j;

}

}

}

PAB[i]=idx;

frag[i]=max;

BA[idx]=1;

max=0;

}

break;

default:

printf("\nInvalid input\n");

}

printf("\nProcess number\tAssigned Block\tProcess size\tBlock size\tFragement\n");

for(i=0;i<noOfProcess;i++){

printf("P%d\t\t%d\t\t%d\t\t%d\t\t%d\t\t\n",i+1,PAB[i]+1,process[i],block[PAB[i]],frag[i]);

}

return 0;

}

**Description:**

In this assignment I am going to write a program to simulate the following contiguous memory allocation techniques a) Worst-fit b) Best-fit c) First-fit.

1. Worst-fit: The algorithm used in this technique searches for free-space memory and selects the largest free partition and allocate to the processes.
2. Best-fit: The algorithm used in this technique searches for free-space memory and selects the smallest free partition which meets the requirement of the requesting process.
3. First-fit.: This technique searches for free-space memory and selects the first free partition which meets the requirement of the requesting process.

**Algorithm:**

1. Worst-fit:
2. Loop i=1 to i=number of process(n)
3. Loop j=1 to j=number of block(m)
4. If block is free and size is maximum then allocate to the process and return.
5. Else put the process in waiting

Complexity: O(m\*n)

1. Best-fit:
2. Loop i=1 to i=number of process(n)
3. Loop j=1 to j=number of block(m)
4. If block is free and size is minimum and meets the requirements then allocate and return
5. Else put the process in waiting

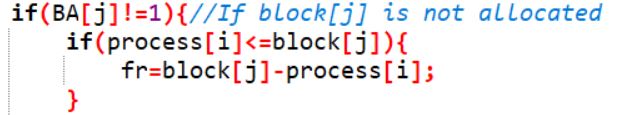
Complexity: O(m\*n)

1. First-fit:
2. Loop i=1 to i=number of process(n)
3. Loop j=1 to j=number of block(m)
4. If the block is free and meets the requirements then allocate and return.
5. Else put that process in waiting list.

Complexity: O(m\*n)

**Constraints:**

If a block is already allocated then it will not allow to allocate again.



**Have you made minimum 5 revisions of solution on GitHub?**

**Ans:** YES!

**GitHub Link:** <https://github.com/aktalukdar/simulationofcontiguousmemoryallocation>