

# Memory Placement Strategies

## 1. First Fit -

```
import java.util.Scanner;

public class FirstFit{

    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);

        System.out.println("Enter number of
processes: ");

        int numP = sc.nextInt();
        int arr[] = new int[numP];
        for(int i=0; i<numP; i++){

            System.out.println("Enter memory
requirement of process " + (i+1)+" ": );
            arr[i] = sc.nextInt();
        }

        System.out.println("Enter number of memory
blocks: ");

        int numB = sc.nextInt();
        int memory[] = new int[numB];
        for(int i=0; i<numB; i++){

            System.out.println("Enter size of memory
block " + (i+1)+" ": );
            memory[i] = sc.nextInt();
        }

        System.out.println();

        System.out.println("Initial available memory
blocks: ");

        for(int i=0; i<numB; i++){

            System.out.println(memory[i] + " ");

        }

        //allocating memory
        for(int i=0; i<numP; i++){

            boolean allocated = false;
            int allocatedWhere = -1;

            for(int j=0; j<numB; j++){

                if(memory[j] >= arr[i]){

                    memory[j] -= arr[i];
                    allocated = true;
                    allocatedWhere = j;
                    break;
                }
            }

            if(allocated){

                System.out.println("Memory allocation
for Process P" + (i+1) + " is successful in memory
block: " + (allocatedWhere + 1));

            }
            else{

                System.out.println("Memory allocation
for Process P" + (i + 1) + " is unsuccessful ");

            }
        }

        System.out.println();

        System.out.println("Fragmented memory
blocks: ");

        for(int i=0; i<numB; i++){

            System.out.println(memory[i] + " ");

        }
    }
}
```

```

dypcoe-student@admin1-MS-7D48: ~/Practical 2
dypcoe-student@admin1-MS-7D48:~/Practical 2$ javac FirstFit.java
dypcoe-student@admin1-MS-7D48:~/Practical 2$ java FirstFit
Enter number of processes:
4
Enter memory requirement of process 1:
50
Enter memory requirement of process 2:
150
Enter memory requirement of process 3:
100
Enter memory requirement of process 4:
200
Enter number of memory blocks:
4
Enter size of memory block 1:
200
Enter size of memory block 2:
300
Enter size of memory block 3:
150
Enter size of memory block 4:
250

Initial available memory blocks:
200
300
150
250
Memory allocation for Process P1 is successful in memory block: 1
Memory allocation for Process P2 is successful in memory block: 1
Memory allocation for Process P3 is successful in memory block: 2
Memory allocation for Process P4 is successful in memory block: 2

Fragmented memory blocks:
0
0
150
250
dypcoe-student@admin1-MS-7D48:~/Practical 2$ 

```

## 2. Best Fit -

```
import java.util.*;
```

```

public class BestFit{
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);

        System.out.println("Enter number of
processes: ");

        int numP = sc.nextInt();
        int arr[] = new int[numP];
        for(int i=0; i<numP; i++){
            System.out.println("Enter memory
requirement of process " + (i+1)+" ": " );
            arr[i] = sc.nextInt();
        }

        //sorting

```

```
Arrays.sort(arr);
```

```

        System.out.println("Enter number of memory
blocks: ");
        int numB = sc.nextInt();
        int memory[] = new int[numB];
        for(int i=0; i<numB; i++){
            System.out.println("Enter size of memory
block " + (i+1)+" ": " );
            memory[i] = sc.nextInt();
        }

        System.out.println();

        System.out.println("Initial available memory
blocks: ");

        for(int i=0; i<numB; i++){
            System.out.println(memory[i] + " ");

```

```

}

//allocating memory
for(int i=0; i<numP; i++){
boolean allocated = false;
int allocatedWhere = -1;
    for(int j=0; j<numB; j++){
        if(memory[j] >= arr[i]){
            memory[j] -= arr[i];
            allocated = true;
            allocatedWhere = j;
            break;
        }
    }
    if(allocated){

```

```

        System.out.println("Memory allocation
for Process P" + (i+1) + " is successful in memory
block: " + (allocatedWhere + 1));
    }
    else{
        System.out.println("Memory allocation
for Process P" + (i + 1 ) + " is unsuccessful ");
    }
}

System.out.println();

System.out.println("Fragmented memory
blocks: ");
    for(int i=0; i<numB; i++){
        System.out.println(memory[i] + " ");
    }
}
}
}

```

```
dypcoe-student@admin1-MS-7D48: ~/Practical 2
dypcoe-student@admin1-MS-7D48:~/Practical 2$ java BestFit
Enter number of processes:
3
Enter memory requirement of process 1:
50
Enter memory requirement of process 2:
100
Enter memory requirement of process 3:
150
Enter number of memory blocks:
3
Enter size of memory block 1:
100
Enter size of memory block 2:
150
Enter size of memory block 3:
200

Initial available memory blocks:
100
150
200
Memory allocation for Process P1 is successful in memory block: 1
Memory allocation for Process P2 is successful in memory block: 2
Memory allocation for Process P3 is successful in memory block: 3

Fragmented memory blocks:
50
50
50
dypcoe-student@admin1-MS-7D48:~/Practical 2$
```

### 3. Next Fit -

```
import java.util.Scanner;

public class NextFit {
    static void nextFit(int blockSize[], int m, int processSize[], int n) {
        int[] allocation = new int[n];
        for (int i = 0; i < n; i++) {
            allocation[i] = -1;
        }
        int j = 0;
        for (int i = 0; i < n; i++) {
            int count = 0;
            while (count < m) {
                if (blockSize[j] >= processSize[i]) {
                    allocation[i] = j;
                }
                else {
                    j = (j + 1) % m;
                    count++;
                }
            }
            System.out.println("Process No.\tProcess Size\tBlock No.");
            for (int i = 0; i < n; i++) {
                System.out.print(" " + (i + 1) + "\t\t" + processSize[i] + "\t\t");
                if (allocation[i] != -1) {
                    System.out.println(allocation[i] + 1);
                }
            }
        }
    }
}
```

```
dypcoe-student@admin1-MS-7D48: ~/Practical 2
dypcoe-student@admin1-MS-7D48:~/Practical 2$ javac NextFit.java
dypcoe-student@admin1-MS-7D48:~/Practical 2$ java NextFit
Enter number of memory blocks: 3
Enter size of each memory block:
50
150
200
Enter number of processes: 3
Enter size of each process:
25
200
150
Process No.      Process Size      Block No.
    1             25             1
    2            200             3
    3            150             2
dypcoe-student@admin1-MS-7D48:~/Practical 2$
```

#### 4. Worst Fit -

```
import java.util.Scanner;
```

```
public class WorstFit{
```

```
    public static int maxBlock(int[] memory, int
sizeReq){
```

```
        int maxIndex = -1;
```

```
        int maxSize = -1;
```

```
        for(int i=0; i<memory.length; i++){
```

```
            if(memory[i] > maxSize &&
memory[i] >= sizeReq){
```

```
                maxIndex = i;
```

```
                maxSize = memory[i];
```

```
            }
```

```
        }
```

```
        return maxIndex;
```

```
    }
```

```
    public static void main(String[] args){
```

```
        Scanner sc = new Scanner(System.in);
```

```
        System.out.println("Enter number of
processes: ");
```

```
        int numP = sc.nextInt();
```

```
        int arr[] = new int[numP];
```

```
        for(int i=0; i<numP; i++){
```

```
            System.out.println("Enter memory
requirement of process " + (i+1)+" : ");
```

```
            arr[i] = sc.nextInt();
```

```
        }
```

```
        System.out.println("Enter number of memory
blocks: ");
```

```
        int numB = sc.nextInt();
```

```
        int memory[] = new int[numB];
```

```
        for(int i=0; i<numB; i++){
```

```
            System.out.println("Enter size of memory
block " + (i+1)+" : ");
```

```
            memory[i] = sc.nextInt();
```

```
        }
```

```

        }

        System.out.println();
        System.out.println("Initial available memory
        blocks: ");
        for(int i=0; i<numB; i++){
            System.out.println(memory[i] + " ");
        }

        //allocating memory
        for(int i=0; i<numP; i++){
            int index = maxBlock(memory, arr[i]);
            if(index != -1){
                memory[index] -= arr[i];
                System.out.println("Memory
                allocation for Process P" + (i+1) + " is successful
                in memory block: " + (index+1));
            }
        }
    }

    else{
        System.out.println("Memory
        allocation for Process P" + (i + 1) + " is
        unsuccessful ");
    }
}

System.out.println();
System.out.println("Fragmented memory
blocks: ");
for(int i=0; i<numB; i++){
    System.out.println(memory[i] + " ");
}
}
}

```

The screenshot shows a terminal window titled "dypcoe-student@admin1-MS-7D48: ~/Practical 2". The user runs the command `java WorstFit`. The program prompts for the number of processes (3), memory requirements for each process (50, 100, 150), the number of memory blocks (3), and the sizes of the memory blocks (100, 150, 200). It then displays the initial available memory blocks (100, 150, 200) and the successful memory allocation for each process: Process P1 in block 3, Process P2 in block 2, and Process P3 in block 3. Finally, it shows the fragmented memory blocks (100, 50, 0).

```

dypcoe-student@admin1-MS-7D48:~/Practical 2$ java WorstFit
Enter number of processes:
3
Enter memory requirement of process 1:
50
Enter memory requirement of process 2:
100
Enter memory requirement of process 3:
150
Enter number of memory blocks:
3
Enter size of memory block 1:
100
Enter size of memory block 2:
150
Enter size of memory block 3:
200

Initial available memory blocks:
100
150
200
Memory allocation for Process P1 is successful in memory block: 3
Memory allocation for Process P2 is successful in memory block: 2
Memory allocation for Process P3 is successful in memory block: 3

Fragmented memory blocks:
100
50
0
dypcoe-student@admin1-MS-7D48:~/Practical 2$

```