## Memory managment

## 

Computer memory can be defined as a collection of some data represented in the binary format.

Question

with the help of example explain need of memory managment in multiprogramming

answer

If the size of the main memory is larger than CPU can load more processes in the main memory at the same time and therefore will increase degree of Multi programming as well as CPU utilization.

Example

Let's consider,

Process Size = 4 MB

Main memory size = 4 MB

The process can only reside in the main memory at any time.

If the time for which the process does IO is P,

Then,

CPU utilization = (1-P)

let's say,

P = 70%

CPU utilization = 30 %

Now, increase the memory size, Let's say it is 8 MB.

Process Size = 4 MB

Two processes can reside in the main memory at the same time.

Let's say the time for which, one process does its IO is P,

Then

CPU utilization = (1-P^2)

let's say P = 70 %

CPU utilization = (1-0.49) =0.51 = 51 %

And

If memory size =8\* 1024MB

That’s tends (approach to zero)

Then cpu utilization=100%(approx.)

Therefore, we can state that the CPU utilization will be increased if the memory size gets increased.

Varous ways of memory allocation

1. Contiguous memory allocation
2. Non contigues memory allocation

 Fixed partitioning or Contiguous memory allocation.

1. Simplest technique
2. main memory is divided into partitions of equal or different sizes.
3. operating system always resides in the first partition while the other partitions can be used to store user process

in fixed partitioning

1. The partitions cannot overlap.
2. A process must be contiguously present in a partition for the execution.

Disavantages of fixed parititioning

**1. Internal Fragmentation**

If the size of the process is lesser then the total size of the partition then some size of the partition get wasted and remain unused. This is wastage of the memory and called internal fragmentation.

As shown in the image below, the 4 MB partition is used to load only 3 MB process and the remaining 1 MB got wasted.

**2. External Fragmentation**

The total unused space of various partitions cannot be used to load the processes even though there is space available but not in the contiguous form.

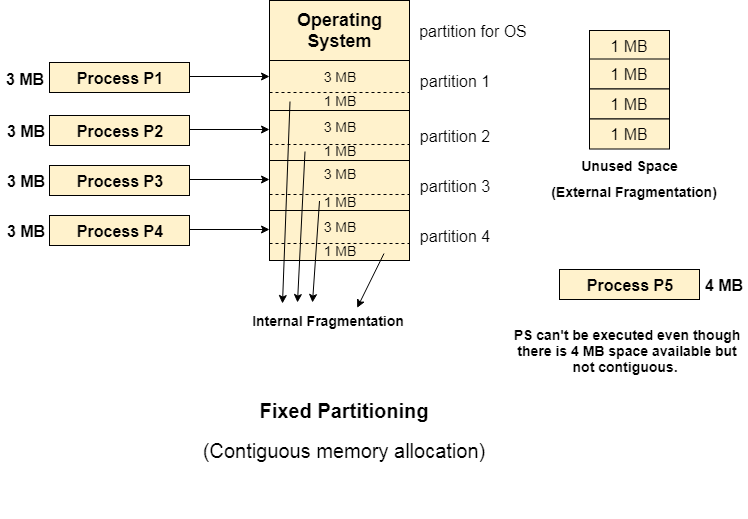
As shown in the image below, the remaining 1 MB space of each partition cannot be used as a unit to store a 4 MB process. Despite of the fact that the sufficient space is available to load the process, process will not be loaded.

**3. Limitation on the size of the process**

If the process size is larger than the size of maximum sized partition then that process cannot be loaded into the memory. Therefore, a limitation can be imposed on the process size that is it cannot be larger than the size of the largest partition.

**4. Degree of multiprogramming is less**

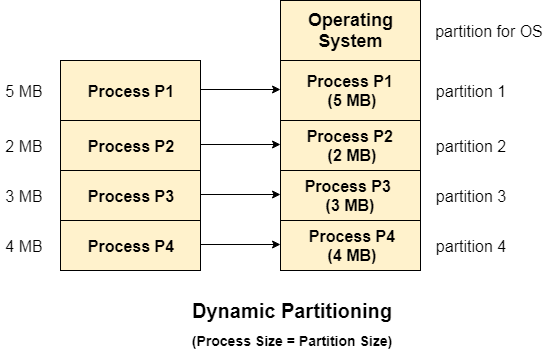
By Degree of multi programming, we simply mean the maximum number of processes that can be loaded into the memory at the same time. In fixed partitioning, the degree of multiprogramming is fixed and very less due to the fact that the size of the partition cannot be varied according to the size of processes.



# **Dynamic Partitioning**

In Dynamic partitioning first partition is reserved for the operating system. The remaining space is divided into parts. The size of each partition will be equal to the size of the process.

partition size varies according to the need of the process so that the internal fragmentation can be avoided.



## Advantages of Dynamic Partitioning over fixed partitioning

### **. No Internal Fragmentation**

Given the fact that the partitions in dynamic partitioning are created according to the need of the process, It is clear that there will not be any internal fragmentation because there will not be any unused remaining space in the partition.

### **2. No Limitation on the size of the process**

In Fixed partitioning, the process with the size greater than the size of the largest partition could not be executed due to the lack of sufficient contiguous memory. Here, In Dynamic partitioning, the process size can't be restricted since the partition size is decided according to the process size.

### **3. Degree of multiprogramming is dynamic**

Due to the absence of internal fragmentation, there will not be any unused space in the partition hence more processes can be loaded in the memory at the same time.

## Disadvantages of dynamic partitioning

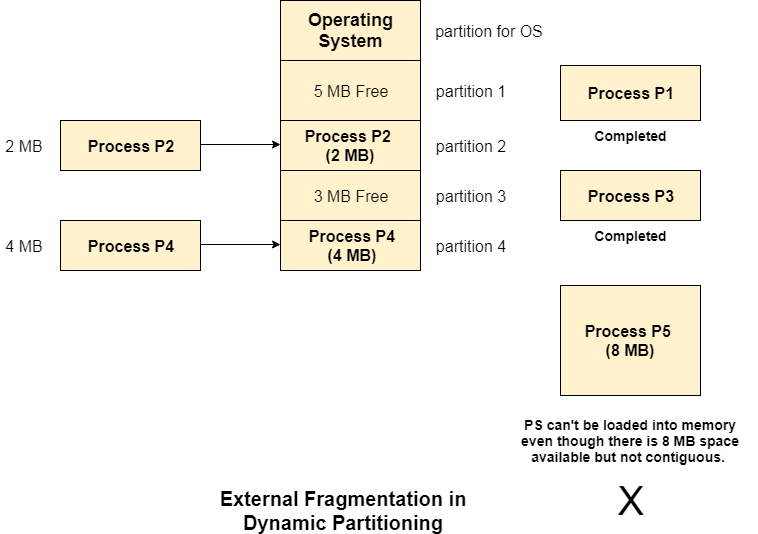
### **External Fragmentation**

Absence of internal fragmentation doesn't mean that there will not be external fragmentation.

Let's consider three processes P1 (1 MB) and P2 (3 MB) and P3 (1 MB) are being loaded in the respective partitions of the main memory.

After some time P1 and P3 got completed and their assigned space is freed. Now there are two unused partitions (1 MB and 1 MB) available in the main memory but they cannot be used to load a 2 MB process in the memory since they are not contiguously located.

The rule says that the process must be contiguously present in the main memory to get executed. We need to change this rule to avoid external fragmentation.



### **Complex Memory Allocation**

In Fixed partitioning, the list of partitions is made once and will never change but in dynamic partitioning, the allocation and deallocation is very complex since the partition size will be varied every time when it is assigned to a new process. OS has to keep track of all the partitions.

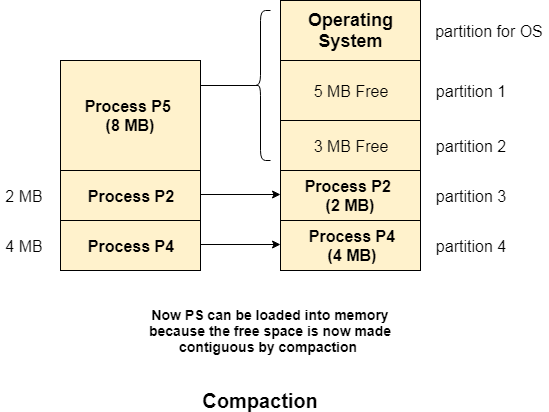
Due to the fact that the allocation and deallocation are done very frequently in dynamic memory allocation and the partition size will be changed at each time, it is going to be very difficult for OS to manage everything.

# **Compaction**

Use to solve the problem of external fragmentation

minimize the probability of external fragmentation

compation is the processes of free partitions are merged which can now be allocated according to the needs of new processes. Also called defragmention



### **Problem with Compaction**

efficiency of the system is decreased in the case of compaction due to the fact that all the free spaces will be transferred from several places to a single place

consider that OS needs 6 NS to copy 1 byte from one place to another.

1. 1 B transfer needs 6 NS
2. 256 MB transfer needs 256 X 2^20 X 6 X 10 ^ -9 secs

# **Bit Map for Dynamic Partitioning**

The Main concern for dynamic partitioning is keeping track of all the free and allocated partitions. However, the Operating system uses following data structures for this task.

1. Bit Map
2. Linked List

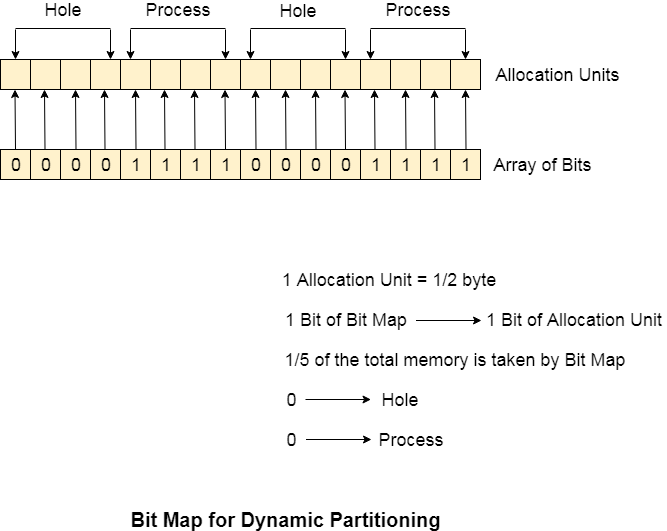
Bit Map is the least famous data structure to store the details. In this scheme, the main memory is divided into the collection of allocation units. One or more allocation units may be allocated to a process according to the need of that process. However, the size of the allocation unit is fixed that is defined by the Operating System and never changed. Although the partition size may vary but the allocation size is fixed.

The main task of the operating system is to keep track of whether the partition is free or filled. For this purpose, the operating system also manages another data structure that is called bitmap.

The process or the hole in Allocation units is represented by a flag bit of bitmap. In the image shown below, a flag bit is defined for every bit of allocation units. However, it is not the general case, it depends on the OS that, for how many bits of the allocation units, it wants to store the flag bit.

The flag bit is set to 1 if there is a contiguously present process at the adjacent bit in allocation unit otherwise it is set to 0.

A string of 0s in the bitmap shows that there is a hole in the relative Allocation unit while the string of 1s represents the process in the relative allocation unit.



## Disadvantages of using Bitmap

1. The OS has to assign some memory for bitmap as well since it stores the details about allocation units. That much amount of memory cannot be used to load any process therefore that decreases the degree of multiprogramming as well as throughput.

In the above image,

The allocation unit is of 4 bits that is 0.5 bits. Here, 1 bit of the bitmap is representing 1 bit of allocation unit.

1. Size of 1 allocation unit = 4 bits
2. Size of bitmap = 1/(4+1) = 1/5 of total main memory.

Therefore, in this bitmap configuration, 1/5 of total main memory is wasted.

1. To identify any hole in the memory, the OS need to search the string of 0s in the bitmap. This searching takes a huge amount of time which makes the system inefficient to some extend

# **Linked List for Dynamic Partitioning**

The better and the most popular approach to keep track the free or filled partitions is using Linked List.

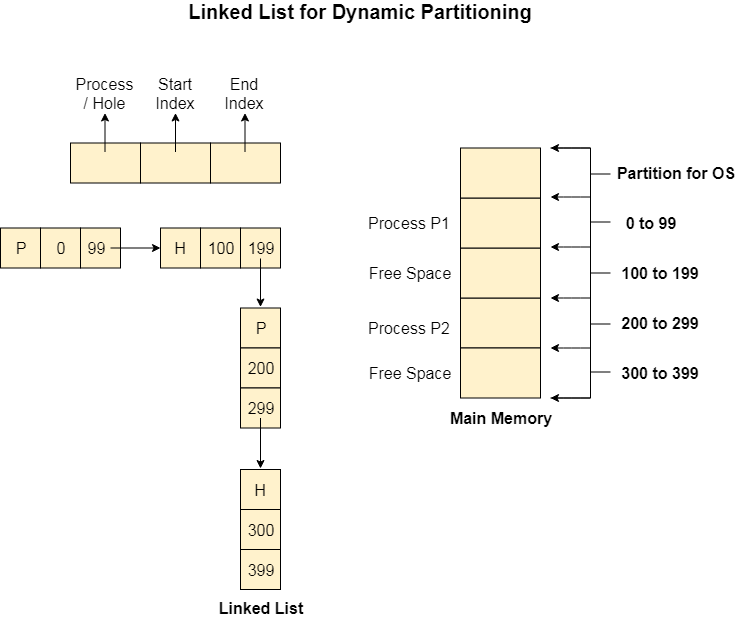
In this approach, the Operating system maintains a linked list where each node represents each partition. Every node has three fields.

1. First field of the node stores a flag bit which shows whether the partition is a hole or some process is inside.
2. Second field stores the starting index of the partition.
3. Third filed stores the end index of the partition.

If a partition is freed at some point of time then that partition will be merged with its adjacent free partition without doing any extra effort.

There are some points which need to be focused while using this approach.

1. The OS must be very clear about the location of the new node which is to be added in the linked list. However, adding the node according to the increasing order of starting index is suggestible.
2. Using a doubly linked list will make some positive effects on the performance due to the fact that a node in the doubly link list can also keep track of its previous node.



# **Partitioning Algorithms**

**1. First Fit Algorithm**

First Fit algorithm scans the linked list and whenever it finds the first big enough hole to store a process, it stops scanning and load the process into that hole. This procedure produces two partitions. Out of them, one partition will be a hole while the other partition will store the process.

First Fit algorithm maintains the linked list according to the increasing order of starting index. This is the simplest to implement among all the algorithms and produces bigger holes as compare to the other algorithms.

**2. Next Fit Algorithm**

Next Fit algorithm is similar to First Fit algorithm except the fact that, Next fit scans the linked list from the node where it previously allocated a hole.

Next fit doesn't scan the whole list, it starts scanning the list from the next node. The idea behind the next fit is the fact that the list has been scanned once therefore the probability of finding the hole is larger in the remaining part of the list.

Experiments over the algorithm have shown that the next fit is not better then the first fit. So it is not being used these days in most of the cases.

**3. Best Fit Algorithm**

The Best Fit algorithm tries to find out the smallest hole possible in the list that can accommodate the size requirement of the process.

Using Best Fit has some disadvantages.

1. 1. It is slower because it scans the entire list every time and tries to find out the smallest hole which can satisfy the requirement the process.
2. Due to the fact that the difference between the whole size and the process size is very small, the holes produced will be as small as it cannot be used to load any process and therefore it remains useless.  
   Despite of the fact that the name of the algorithm is best fit, It is not the best algorithm among all.

**4. Worst Fit Algorithm**

The worst fit algorithm scans the entire list every time and tries to find out the biggest hole in the list which can fulfill the requirement of the process.

Despite of the fact that this algorithm produces the larger holes to load the other processes, this is not the better approach due to the fact that it is slower because it searches the entire list every time again and again.

**5. Quick Fit Algorithm**

The quick fit algorithm suggestsmaintaining the different lists of frequently used sizes. Although, it is not practically suggestible because the procedure takes so much time to create the different lists and then expending the holes to load a process.

The first fit algorithm is **the best algorithm** among all because

1. It takes lesser time compare to the other algorithms.
2. It produces bigger holes that can be used to load other processes later on.
3. It is easiest to implement.

### **Q. Process requests are given as;**

**25 K , 50 K , 100 K , 75 K**



Determine the algorithm which can optimally satisfy this requirement.

1. First Fit algorithm
2. Best Fit Algorithm
3. Neither of the two
4. Both of them

In the question, there are five partitions in the memory. 3 partitions are having processes inside them and two partitions are holes.

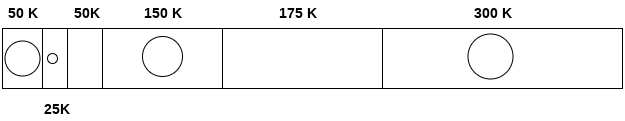
Our task is to check the algorithm which can satisfy the request optimally.

## Using First Fit algorithm

Let's see, how first fit algorithm works on this problem.

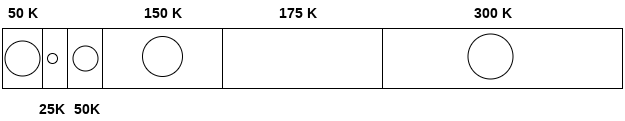
### **1. 25 K requirement**

The algorithm scans the list until it gets first hole which should be big enough to satisfy the request of 25 K. it gets the space in the second partition which is free hence it allocates 25 K out of 75 K to the process and the remaining 50 K is produced as hole.



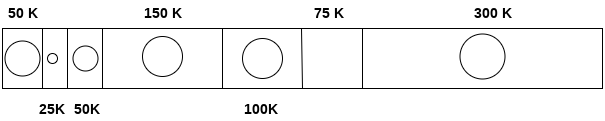
### **2. 50 K requirement**

The 50 K requirement can be fulfilled by allocating the third partition which is 50 K in size to the process. No free space is produced as free space.



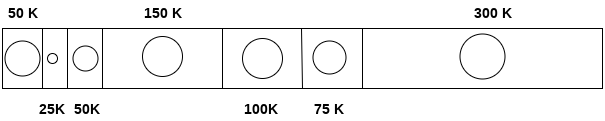
### **3. 100 K requirement**

100 K requirement can be fulfilled by using the fifth partition of 175 K size. Out of 175 K, 100 K will be allocated and remaining 75 K will be there as a hole.



### **4. 75 K requirement**

Since we are having a 75 K free partition hence we can allocate that much space to the process which is demanding just 75 K space.



Using first fit algorithm, we have fulfilled the entire request optimally and no useless space is remaining.

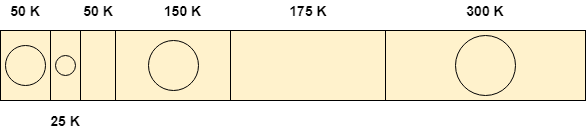
Let's see, How Best Fit algorithm performs for the problem.

## Using Best Fit Algorithm

### **1. 25 K requirement**

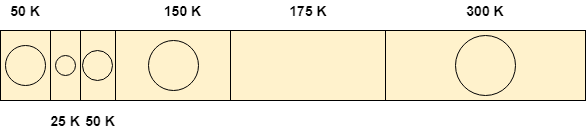
To allocate 25 K space using best fit approach, need to scan the whole list and then we find that a 75 K partition is free and the smallest among all, which can accommodate the need of the process.

Therefore 25 K out of those 75 K free partition is allocated to the process and the remaining 5o K is produced as a hole.



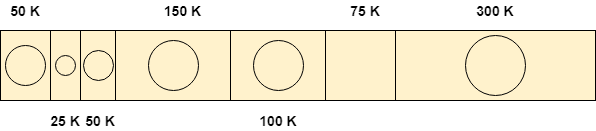
### **2. 50 K requirement**

To satisfy this need, we will again scan the whole list and then find the 50 K space is free which the exact match of the need is. Therefore, it will be allocated for the process.



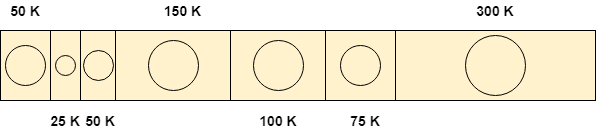
### **3. 100 K requirement**

100 K need is close enough to the 175 K space. The algorithm scans the whole list and then allocates 100 K out of 175 K from the 5th free partition.



### **4. 75 K requirement**

75 K requirement will get the space of 75 K from the 6th free partition but the algorithm will scan the whole list in the process of taking this decision.



By following both of the algorithms, we have noticed that both the algorithms perform similar to most of the extant in this case.

Both can satisfy the need of the processes but however, the best fit algorithm scans the list again and again which takes lot of time.

Therefore, if you ask me that which algorithm performs in more optimal way then it will be **First Fit algorithm** for sure.

Therefore, the answer in this case is A.