Vivekanand Education Society's
Institute of Technology
Approved by AICTE & Affiliated to University of Mumbai

Artificial Intelligence and Data Science Department.

OS / Even Sem 2021-22 / Experiment 10.

YASH SARANG.
47 / D6AD.

EXPERIMENT - 10.

Dynamic Partitioning.

Aim: Write a program to implement dynamic partitioning placement algorithms i.e Best Fit, First-Fit, Worst-Fit, etc.

Theory:

Theory:

If we were to use dynamic partitioning to manage our memory, and we wanted to allocate memory to a new process, we would have to decide which algorithm we would use to select an empty block of memory to use. Each algorithm are limited to blocks that are bigger than new processes.

Some of these algorithms are:

1) Best fit - Chooses the block closest in size to the request

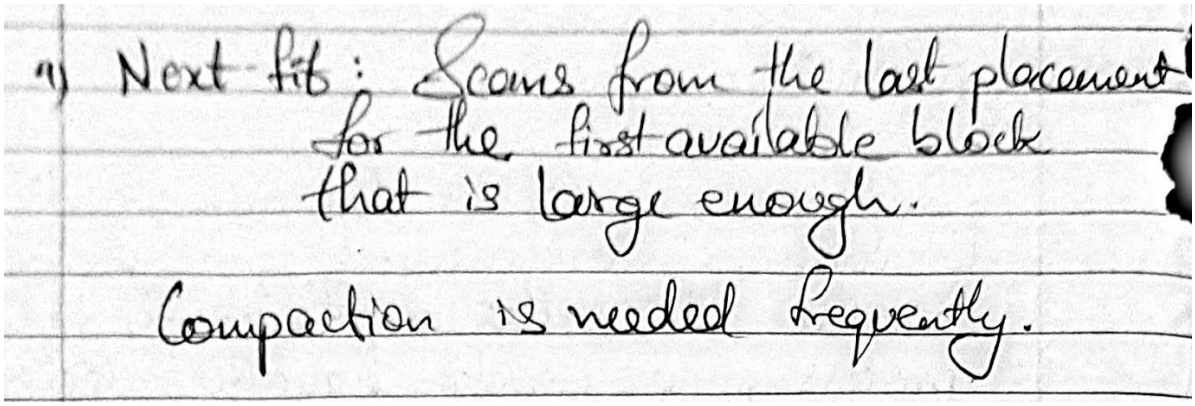
It is usually the worst performer. Compaction is needed more frequently

2) First fit - Scans from the beginning for the first available block that is large enough.

It is usually the best & fastest.

3) Worst fit - Chooses the process which is largest sufficient among the freely available partitions in memory.

If a large process comes at a later stage, memory will have no space to accomodate it.



Code:

1. First Fit

```cpp
#include<bits/stdc++.h>
using namespace std;

void firstFit(int blockSize[], int m,
              int processSize[], int n)
{
    int allocation[n];

    memset(allocation, -1, sizeof(allocation));

    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < m; j++)
        {
            if (blockSize[j] >= processSize[i])
            {
                // allocate block j to p[i] process
                allocation[i] = j;

                // Reduce available memory in this block.
                blockSize[j] -= processSize[i];

                break;
            }
        }
    }

    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 blockSize[] = {100, 500, 200, 300, 600};
    int processSize[] = {212, 417, 112, 426};
    int m = sizeof(blockSize) / sizeof(blockSize[0]);
    int n = sizeof(processSize) / sizeof(processSize[0]);

    firstFit(blockSize, m, processSize, n);

    return 0 ;
}
```

Output:

```
Process No.     Process Size    Block no.
 1              212             2
 2              417             5
 3              112             2
 4              426             Not Allocated


...Program finished with exit code 0
Press ENTER to exit console.
```

2. Best Fit

```cpp
#include<bits/stdc++.h>
using namespace std;

void bestFit(int blockSize[], int m, int processSize[], int n)
{
    int allocation[n];
    memset(allocation, -1, sizeof(allocation));
    for (int i=0; i<n; i++)
    {
        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)
        {
            allocation[i] = bestIdx;

            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 blockSize[] = {100, 500, 200, 300, 600};
    int processSize[] = {212, 417, 112, 426};
    int m = sizeof(blockSize)/sizeof(blockSize[0]);
    int n = sizeof(processSize)/sizeof(processSize[0]);
    bestFit(blockSize, m, processSize, n);
    return 0 ;
}
```

Output:

```
Process No.      Process Size     Block no.
   1             212              4
   2             417              2
   3             112              3
   4             426              5


...Program finished with exit code 0
Press ENTER to exit console.
```

3. Worst Fit

```cpp
#include<bits/stdc++.h>
using namespace std;
void worstFit(int blockSize[], int m, int processSize[],
                                                    int n)
{
    int allocation[n];
    memset(allocation, -1, sizeof(allocation));
    for (int i=0; i<n; i++)
    {
        int wstIdx = -1;
        for (int j=0; j<m; j++)
        {
            if (blockSize[j] >= processSize[i])
            {
                if (wstIdx == -1)
                    wstIdx = j;
                else if (blockSize[wstIdx] < blockSize[j])
                    wstIdx = j;
            }
        }
        if (wstIdx != -1)
        {
            allocation[i] = wstIdx;
            blockSize[wstIdx] -= 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 blockSize[] = {100, 500, 200, 300, 600};
    int processSize[] = {212, 417, 112, 426};
    int m = sizeof(blockSize)/sizeof(blockSize[0]);
    int n = sizeof(processSize)/sizeof(processSize[0]);
    worstFit(blockSize, m, processSize, n);
    return 0 ;
}
```

Output:

```
Process No.     Process Size    Block no.
   1            212             5
   2            417             2
   3            112             5
   4            426             Not Allocated


...Program finished with exit code 0
Press ENTER to exit console.
```

Conclusion:-