ECE254 Lab 4 Report

Problem Statement:

This report will analyze “Worst Fit”, and “Best Fit” memory allocation methods. “Best Fit” will appropriately split a memory block that is equal to or larger than the requested size and closest to the requested size. “Worst Fit” will appropriately split a memory block that is equal to or larger than the requested size and is the largest available memory block.

When creating the two algorithms, there is one call to malloc for each therefore the memory that is initialized should contain the control data and actual data. Utilizing the dynamic allocations by best fit and worst fit, the respective algorithms manage the fixed memory space.

Data Structures and Algorithms:

Data Structures:

Linked list:

Using linked list to manage the data blocks. Each block works as the node in the linked list. Allocating the new memory block by adding the new node into the linked list.

The structure of the node is as follows:

```
struct n
{
    void* mem;              // pointer to memory
    size_t free_mem;        // amount of free memory
    struct n* next;         // next node
    struct n* prev;         // prev node
    int allocated;          // allocation status
};
typedef struct n node;
```

Algorithms:

Best fit:

Finding the smallest size block, which is equal or greater than the new data block, to insert the new data. Splitting the node when the node is large by creating two nodes, one is allocated block, and another is free memory space. Recombining the two free memory blocks when doing the deallocation. Checking if the next node is free memory. If it does, recombining the current node with its next node. Then checking the previous node by the same method.

Worst fit:

Finding the biggest size block, which is equal or less than the new data block, to insert the new data. Splitting the node when the node is large by creating two nodes, one is allocated block, and another is free memory space. Recombining the two free memory blocks when doing the deallocation. Checking if the next node is free memory. If it does, recombining the current node with its next node. Then checking the previous node by the same method.

Testing Scenario Description:

Testing working functionality of code:

1. Randomly allocate for both Best Fit and Worst Fit until there is no more space left to allocate.
2. Deallocate all odd nodes for both Best Fit and Worst Fit and print the status of each node.
3. Deallocate all even nodes for both Best Fit and Worst Fit and print the status of each node. After this step there should only be one node that is not allocated with all the free memory.
4. Can repeat the above procedure in a for loop to test validity of the code.

```
**************************************
WORST Fit after DeAlloc loop, should Dealloc all even Nodes:
**************************************
Mode: WORST FIT
Node 1
        Memory: 984
        Memory ptr: 0x7f95d5802a68
        Allocated: 0
        Node Address: 0x7f95d5802400
        Next Ptr: NULL
        Prev Ptr: NULL
**************************************
**************************************
Best Fit after DeAlloc loop, should Dealloc all even Nodes:
**************************************
Mode: BEST FIT
Node 1
        Memory: 984
        Memory ptr: 0x7f95d5801668
        Allocated: 0
        Node Address: 0x7f95d5801000
        Next Ptr: NULL
        Prev Ptr: NULL
**************************************
```

Testing External Fragmentation:

1. Allocate blocks with random size of 1 to 13.
2. Then, deallocate blocks with alternating index (switches between all odd numbers and all even).
3. Repeat the above steps for 100 times.
4. Print the external fragmentation with size is less than 5.

Experimental Data:

Best Fit:

6	4	3	3	3	4	3	2	2	2
2	1	1	1	1	1	1	1	1	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Worst Fit:

4	3	3	3	3	3	1	2	2	2
2	2	2	2	2	2	2	1	1	2
2	2	1	1	0	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	0	1	0	1	0	1	1	1
1	1	0	1	1	1	1	1	1	1
0	1	0	1	0	1	0	1	0	1
1	1	1	1	1	1	0	1	1	1
0	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	0

Average fragmentation of best fit over 100 times: 0.41584158 fragmentations
Average fragmentation of worst fit over 100 times: 1.12871287 fragmentations

Analysis of Experimental Data:

From the above data, the average fragmentation of best fit is 0.41584158 and the average fragmentation of worst fit is 1.12871287. So, we see that the best fit has lower average fragmentations over running 100 times. Having a larger external fragmentation, you are potentially wasting more memory by having small chunks of unusable memory.

Conclusion:

From the analysis above, it can be seen that the Best Fit has lower external fragmentation on average when comparted to Worst Fit. By having a lower external fragmentation average, Best Fit

wastes less memory during allocation and deallocation cycles. This is preferable as less resources are being wasted. Best Fit is preferable over Worst Fit due to lower external fragmentation.