Mohammed Al Aadhami
ID# 1001417231

Raghad Safauldeen
ID# 1001417235

* Benchmarks: A discussion on how you benchmarked your library and what the results were.

We benchmarked our library by using two benchmark programs provided by the professor, bench1 and bench2.

* Which heap management strategy does the best job of reusing free blocks? Which one is the worst?

The best heap management strategy to use gave us different results. For bench1, both of best fit and worst fit give same reuse, which is higher than first fit. However, in bench2, the higher reuse of free blocks is the first fit. According to the statistics of the bench2, the first fit produces the highest number of reusing free blocks, followed by both best fit and worst fit with the same values.

* Which heap management strategy requires the least amount of heap space? Which one is the worst?

As for bench1, we noticed that the highest value of max heap is for best fit, followed by worst fit and then first fit (least heap). On the other hand, for bench2, we found that the highest max heap is for first fit, followed by both best fit and worst fit at the same values (the least heap values).

* Which heap management strategy allows for the most splits? most coalescing? least?

For bench1, the split values were the same for best fit, worst fit and first fit. Also, the coalescing values were at the same for the three algorithms of fits we have.
As for bench2, we found that the highest splits and coalescing are for first fit. Followed by equal values of splits and coalescing for worst fit and best fit.

* Which heap management strategy was the fastest? slowest?

The fastest strategy is for the first fit, because it searches very little, less than the other algorithms, so it would not take much time. Next fit is slower than first fit, because next fit searches the list from the place where it left off every time. Best fit is the slowest algorithm because it searches the whole list every time to pick the smallest hole that is larger than the requested amount of memory. Worst fit is also as slow as the best fit algorithm because it also searches the whole list before it decides to choose the largest hole in memory to use it.

* Which one requires the most mallocs? frees?

In each one of the two benches, bench1 and bench2, we found that the same number of mallocs and the same number of frees took place for the three fit algorithms in each bench test.

* Which one requests the most amount of space?

The worst fit takes the most amount of space, it is not size-efficient because it uses the biggest hole in the memory for each process or free memory allocation. Some argue that the worst fit is not bad because when it gives the largest hole, then the big holes could be split further which is still helpful.

* Which one requires the largest heap?

As for bench1, we noticed that the highest value of max heap is for best fit, but for bench2, we found that the highest max heap is for first fit. But usually the one that requires the largest heap is the worst fit.

Analysis

* Which heap management strategy suffers the most from fragmentation (what type)?

Basically, the type of fragmentation is the external fragmentation (the free memory is more than only one piece, so the free memory sometimes is useless, and programs cannot run using it.)
We can know the most fragmentation by looking at the most number of free blocks at the end, (num blocks). So, the least number of blocks in the free list means the least amount of fragmentation in the end.
We have the similar values for the three algorithms we have for bench1, also we have found that the free is similar in values for the three algorithms we have for the bench2.

* Which heap management strategy is the best?

If we care the most for speed, then we just use the fastest algorithm, which is the first fit, because it searches very little, less than the other algorithms. However, if we care the most for free space, then we use the algorithm that uses the least heap, which was first fit for bench1, and was equally the best fit and worst fit for bench2.
Also, the best algorithm could be the one that uses the least number of splits. (but we found that the three algorithms give the similar splits for bench1. We found the same numbers of splits for bench2 for best fit and worst fit.)

Summary

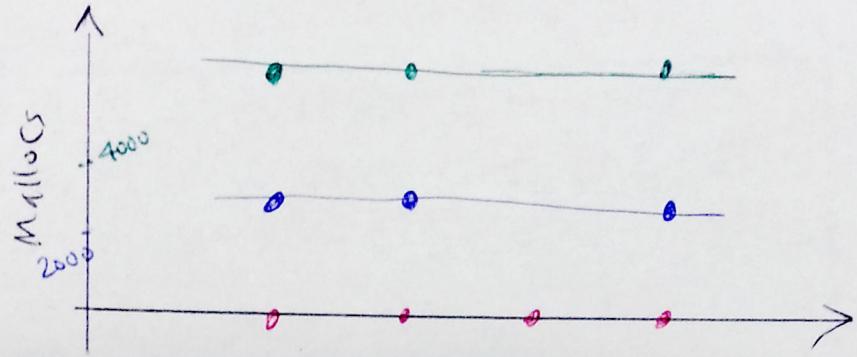
We have four algorithms to manage the heap, the best fit, first fit, next fit, and worst fit. In our program, we have three of these algorithms working, we had the next fit working but for some reason it was not working shortly before we finish the assignment.
Each algorithm has its advantages and disadvantages, and it depends on the uses whether we need a quick algorithm, or a memory efficient algorithm to fulfill the task we want to do. These four different kinds of algorithms could have different number of splits, coalesces, blocks, mallocs, frees, heap grows, and heap sizes.

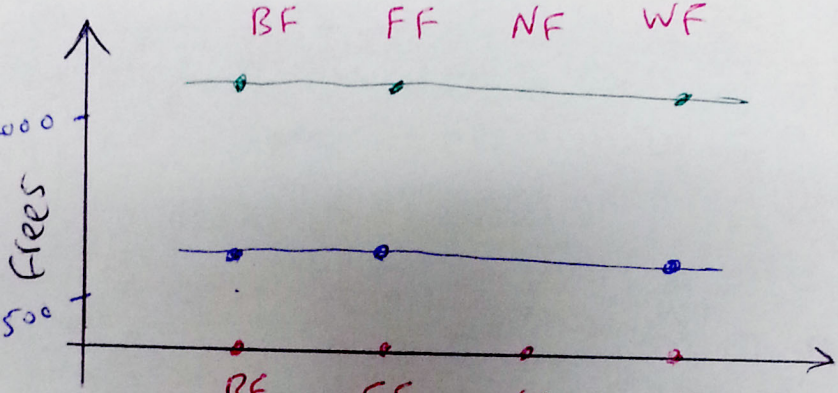
	Bench1 BF	Bench2 BF	Bench1 FF	Bench2 FF	Bench1 WF	Bench2 WF
mallocs	2049	4609	2049	4609	2049	4609
frees	513	4609	513	4609	513	4609
reuses	2	8	1	1003	2	8
grows	2047	4601	2048	3606	2047	4601
splits	1	2	1	1003	1	2
coaleses	0	2	0	2558	0	2
blocks	2048	4601	2049	2051	2048	4601
requested	2049	4609	2049	4609	2049	4609
max heap	4279588	14915072	4107896	14965248	4209148	14915072

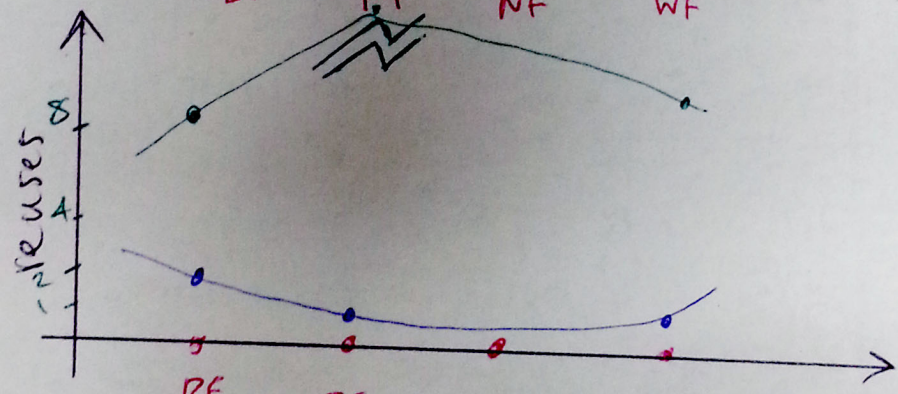
This table shows the different values we got for three types of fit algorithms, best fit, first fit, and worst fit, benchmarked in two benchmark programs, bench1 and bench2.

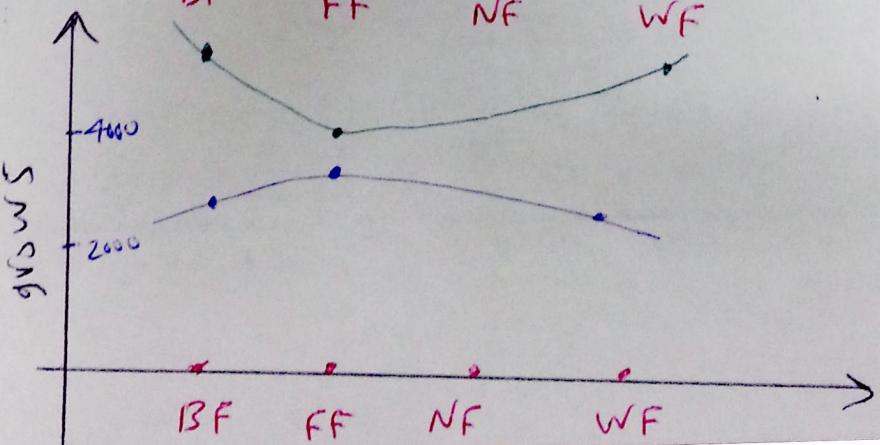
Bench2 in green

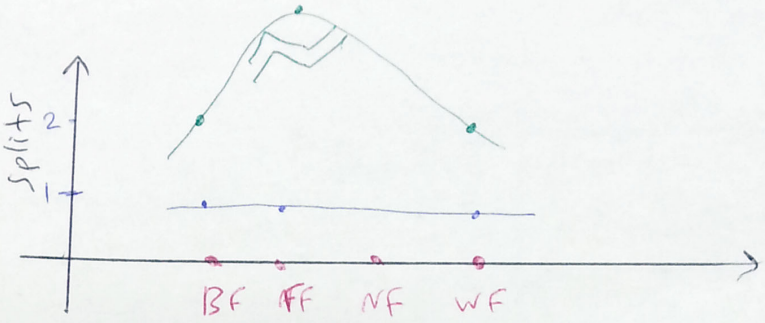
Bench 1 in blue

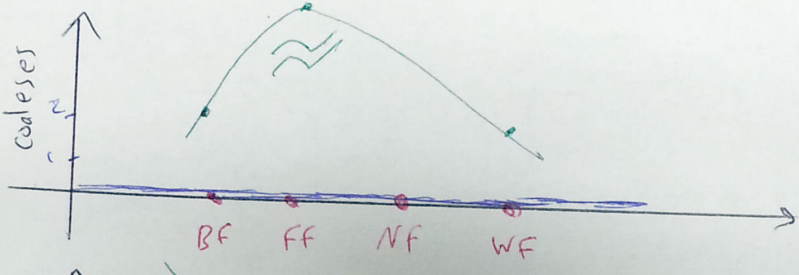


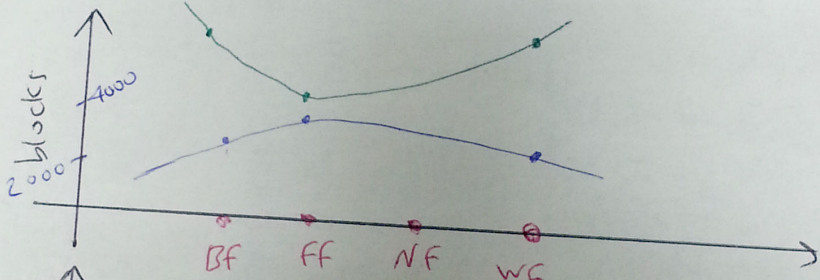


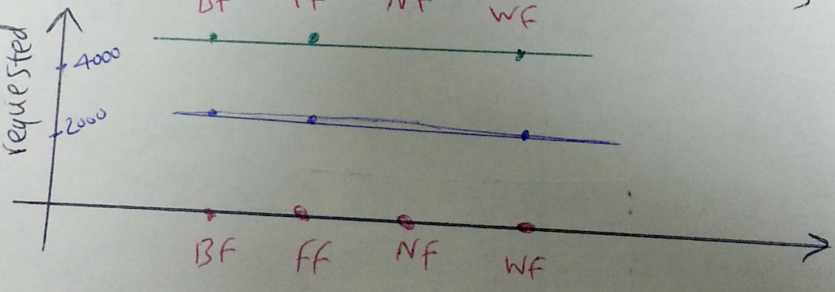


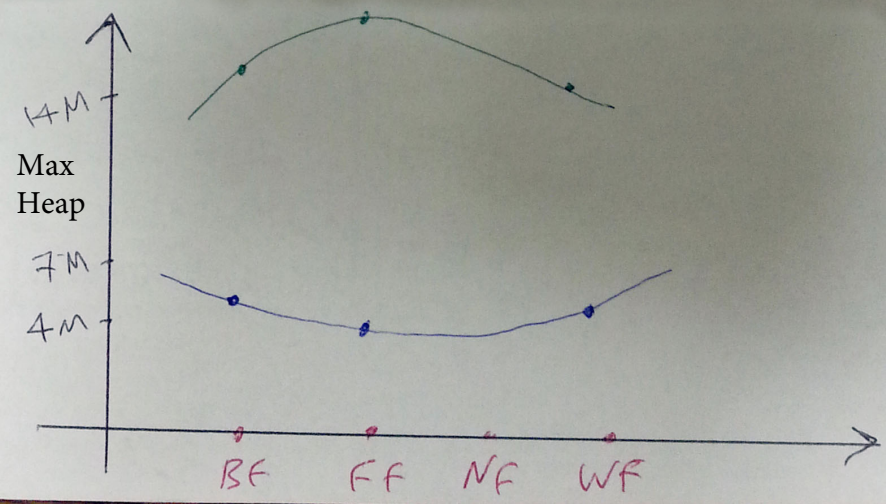



Splits
2
1
BF
FF
NF
WF


Coalesces
2
1
BF
FF
NF
WF


2 blocks
4000
2000
BF
FF
NF
WF


Requested
4000
2000
BF
FF
NF
WF


14M
Max
Heap
7M
4M
BF
FF
NF
WF