Project Report

Alex Geary 1188083

Statistics

traceSmall.txt

Heap size: 1K

FF	Ratio of the heap used:	0.991
	Number of free holes:	1
	Average size of the free holes:	9
	Number of allocated chunks:	28
BF	Ratio of the heap used:	0.991
	Number of free holes:	1
	Average size of the free holes:	9
	Number of allocated chunks:	28
WF	Ratio of the heap used:	0.991
	Number of free holes:	1
	Average size of the free holes:	9
	Number of allocated chunks:	28

Heap size: 16K

FF	Ratio of the heap used:	0.870562
	Number of free holes:	2
	Average size of the free holes:	1035.5
	Number of allocated chunks:	384
BF	Ratio of the heap used:	0.870562
	Number of free holes:	2
	Average size of the free holes:	1035.5
	Number of allocated chunks:	384
WF	Ratio of the heap used:	0.870562
	Number of free holes:	2
	Average size of the free holes:	1035.5
	Number of allocated chunks:	384

Heap size: 128K

FF	Ratio of the heap used:	0.108820
	Number of free holes:	2
	Average size of the free holes:	57035.5
	Number of allocated chunks:	384
BF	Ratio of the heap used:	0.108820
	Number of free holes:	2
	Average size of the free holes:	57035.5
	Number of allocated chunks:	384

WF	Ratio of the heap used:	0.108820
	Number of free holes:	2
	Average size of the free holes:	57035.5
	Number of allocated chunks:	384

traceMedium.txt

Heap size: 1K

FF	Ratio of the heap used:	0.991
	Number of free holes:	1
	Average size of the free holes:	9
	Number of allocated chunks:	28
BF	Ratio of the heap used:	0.991
	Number of free holes:	1
	Average size of the free holes:	9
	Number of allocated chunks:	28
WF	Ratio of the heap used:	0.991
	Number of free holes:	1
	Average size of the free holes:	9
	Number of allocated chunks:	28

Heap size: 16K

FF	Ratio of the heap used:	0.998250
	Number of free holes:	3
	Average size of the free holes:	9.333333
	Number of allocated chunks:	434
BF	Ratio of the heap used:	0.998250
	Number of free holes:	3
	Average size of the free holes:	9.333333
	Number of allocated chunks:	434
WF	Ratio of the heap used:	0.994687
	Number of free holes:	2
	Average size of the free holes:	42.5
	Number of allocated chunks:	433

Heap size: 128K

FF	Ratio of the heap used:	0.190461
	Number of free holes:	5
	Average size of the free holes:	20724.199219
	Number of allocated chunks:	618
BF	Ratio of the heap used:	0.190461
	Number of free holes:	4
	Average size of the free holes:	25905.25
	Number of allocated chunks:	618
WF	Ratio of the heap used:	0.190461
	Number of free holes:	3
	Average size of the free holes:	34540.332031
	Number of allocated chunks:	618

traceLarge.txt

Heap size: 1K

FF	Ratio of the heap used:	0.991
	Number of free holes:	1
	Average size of the free holes:	9
	Number of allocated chunks:	28
BF	Ratio of the heap used:	0.991
	Number of free holes:	1
	Average size of the free holes:	9
	Number of allocated chunks:	28
WF	Ratio of the heap used:	0.991
	Number of free holes:	1
	Average size of the free holes:	9
	Number of allocated chunks:	28

Heap size: 16K

FF	Ratio of the heap used:	0.998250
	Number of free holes:	3
	Average size of the free holes:	9.333333
	Number of allocated chunks:	434
BF	Ratio of the heap used:	0.998250
	Number of free holes:	3
	Average size of the free holes:	9.333333
	Number of allocated chunks:	434
WF	Ratio of the heap used:	0.994687
	Number of free holes:	2
	Average size of the free holes:	42.5
	Number of allocated chunks:	433

Heap size: 128K

FF	Ratio of the heap used:	0
	Number of free holes:	1
	Average size of the free holes:	128000
	Number of allocated chunks:	0
BF	Ratio of the heap used:	0
	Number of free holes:	1
	Average size of the free holes:	128000
	Number of allocated chunks:	0
WF	Ratio of the heap used:	0
	Number of free holes:	1
	Average size of the free holes:	128000
	Number of allocated chunks:	0

Discussion:

When the heap size was 1,000 bytes, the program finished due to a heap allocation failure, because there were no holes big enough for any more allocations on the heap. This was the case for all of the algorithms and with all three input files. There was also no difference between the results from the first, best or worst fit algorithms and this was the case no matter what the size of the input file was.

An observation which can be made from the results using just the small input file is that, there is no difference in performance between the algorithms no matter what the size of the heap is. This observation may not really mean that much though in terms of comparing the algorithms but is probably more something to do with the strings in the file.

When the heap size was 16,000 bytes, the program finished due to a heap allocation failure with all three algorithms when the medium and large input files were used. There was no difference between first, best or worst fit with the smallest input file and there was no difference between the results of the first and best fit algorithms with the medium input file. However the results from worst fit were different for the medium input file compared to the other algorithms. The difference was that there was one less hole for the worst fit algorithm and the average size of the holes was subsequently larger as well. The results for the large input file were exactly the same as the results for the medium input file.

From observing the results from using a heap size of 128,000 bytes, again it seems that using the small input file, there is no difference between the algorithms and this is also the case with the large input file. This is expected with the large input file where the whole file was read and all of the strings in the file were deallocated from the heap leaving one hole of maximum heap size. However there are differences between all three algorithms using the medium input file. There are only three holes with worst fit compared with the five holes for first fit and the four holes for best fit. The reason for the worst fit algorithm having less holes with a heap size of 16,000 bytes and 128,000 bytes is probably due to the first and best fit algorithms leaving smaller holes which couldn't fit strings in them. The worst fit algorithm would pick the biggest hole to put a string in to and it would often be big enough for more strings to fit in the rest of that hole, more often than that of the first and best fit algorithms. The reason why best fit had one less hole than that of the first fit algorithm with a heap size of 16,000 bytes is because best fit would've searched through all the holes available and it would have found one that was a perfect fit, thus, removing the hole on allocation. This could have happened with the first fit algorithm as well, but only if it was the first hole available of which there is less chance of that being the case.

In conclusion it seems that the first and best fit algorithms both appear to be quite similar in performance. Worst fit appears to be the best in terms of achieving a lesser number of holes than the other two algorithms and the larger the heap is, the greater the advantage for the worst fit algorithm will be. This is because the only way the other algorithms can really improve on the performance of worst fit is if allocations are often the perfect size to fit in to a hole and thus remove it altogether. If this isn't the case then worst fit will always output perform the others, in terms of leaving less holes, by simply allocating strings in the last hole which will be the biggest hole for a long time if the heap is really large. In so doing it will not be leaving small holes like the other two algorithms will likely be doing.