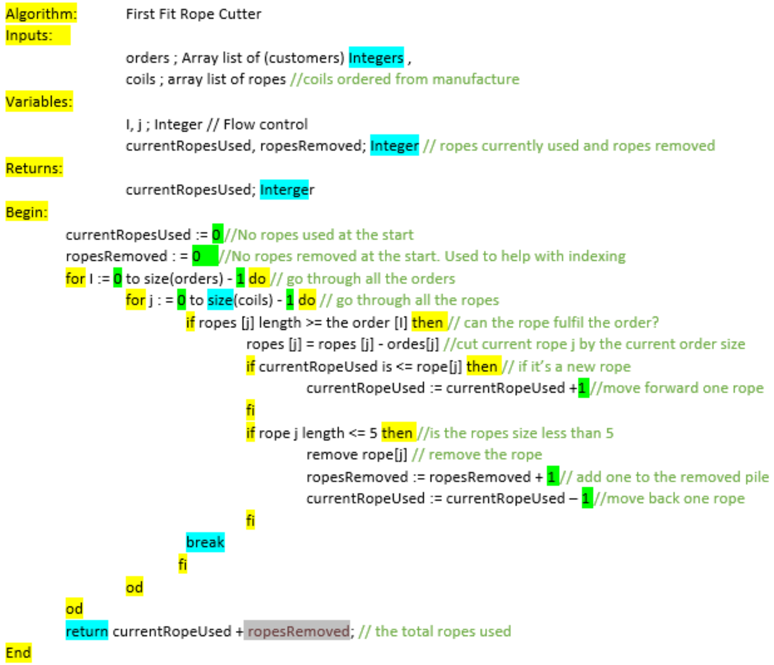
Callum Simpson B6030326

Callum Simpson Report on comparions

Pesduo code

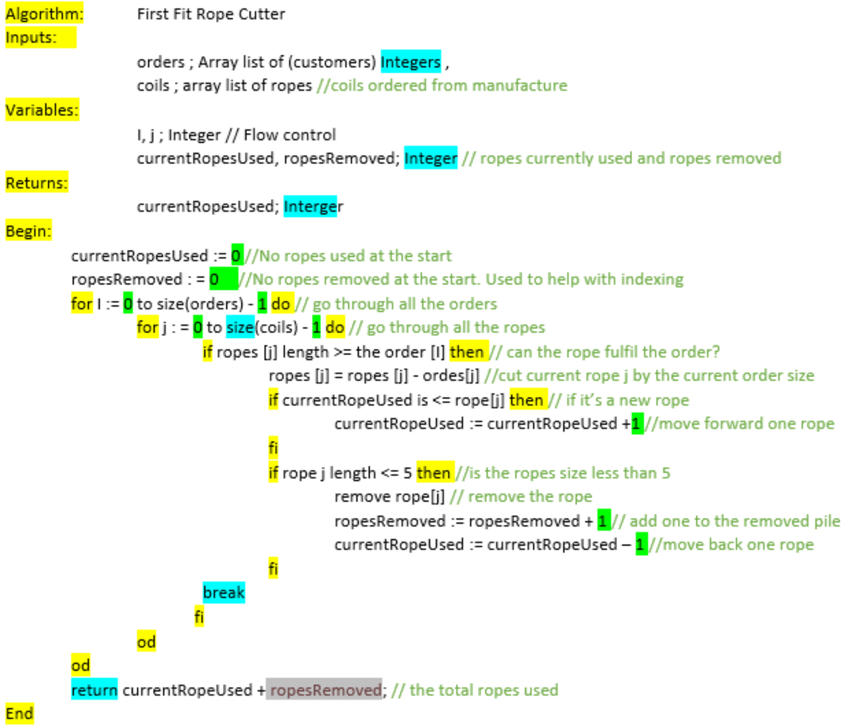
Here is the pesduo code I will be using to create my algorthim.

First fit



```
Algorithm:      First Fit Rope Cutter
Inputs:
                orders ; Array list of (customers) Integers ,
                coils ; array list of ropes //coils ordered from manufacture
Variables:
                I, j ; Integer // Flow control
                currentRopesUsed, ropesRemoved; Integer // ropes currently used and ropes removed
Returns:
                currentRopesUsed; Interger
Begin:
        currentRopesUsed := 0 //No ropes used at the start
        ropesRemoved : = 0   //No ropes removed at the start. Used to help with indexing
        for I := 0 to size(orders) - 1 do // go through all the orders
                for j : = 0 to size(coils) - 1 do // go through all the ropes
                        if ropes [j] length >= the order [I] then // can the rope fulfil the order?
                                ropes [j] = ropes [j] - ordes[j] //cut current rope j by the current order size
                                if currentRopeUsed is <= rope[j] then // if it's a new rope
                                        currentRopeUsed := currentRopeUsed +1 //move forward one rope
                                fi
                                if rope j length <= 5 then //is the ropes size less than 5
                                        remove rope[j] // remove the rope
                                        ropesRemoved := ropesRemoved + 1 // add one to the removed pile
                                        currentRopeUsed := currentRopeUsed – 1 //move back one rope
                                fi
                        break
                        fi
                od
        od
        return currentRopeUsed + ropesRemoved; // the total ropes used
End
```

Next fit



```
Algorithm:      First Fit Rope Cutter
Inputs:
                orders ; Array list of (customers) Integers ,
                coils ; array list of ropes //coils ordered from manufacture
Variables:
                I, j ; Integer // Flow control
                currentRopesUsed, ropesRemoved; Integer // ropes currently used and ropes removed
Returns:
                currentRopesUsed; Interger
Begin:
        currentRopesUsed := 0 //No ropes used at the start
        ropesRemoved : = 0   //No ropes removed at the start. Used to help with indexing
        for I := 0 to size(orders) - 1 do // go through all the orders
                for j : = 0 to size(coils) - 1 do // go through all the ropes
                        if ropes [j] length >= the order [I] then // can the rope fulfil the order?
                                ropes [j] = ropes [j] - ordes[j] //cut current rope j by the current order size
                                if currentRopeUsed is <= rope[j] then // if it's a new rope
                                        currentRopeUsed := currentRopeUsed +1 //move forward one rope
                                fi
                                if rope j length <= 5 then //is the ropes size less than 5
                                        remove rope[j] // remove the rope
                                        ropesRemoved := ropesRemoved + 1 // add one to the removed pile
                                        currentRopeUsed := currentRopeUsed – 1 //move back one rope
                                fi
                        break
                        fi
                od
        od
        return currentRopeUsed + ropesRemoved; // the total ropes used
End
```

Callum Simpson B6030326

About each method

First fit works by getting a order , checks the first rope in the list. If it can be cut, cut it. If after its cut its less than 5 remove the rope from the list. Else move onto the next rope. Its should loop until a suitable rope if found. After an order is complet go to the next order and start to process again with the first rope.

Next fit works in a similar way. Gets a order , checks the first rope in the list. If it can be cut, cut it. If after its cut its less than 5 remove the rope from the list. Its should loop until a suitable rope if found. After an order is complet go to the next order and start to process again however with the last rope used.

Other methods I have implermented

Report

A method to print off a list of ropes / orders

Duplicaties

Two methods to duplicate a list of ropes and a list of orders

Timers

Two methods to time how long it takes my algorthim

Correctness test

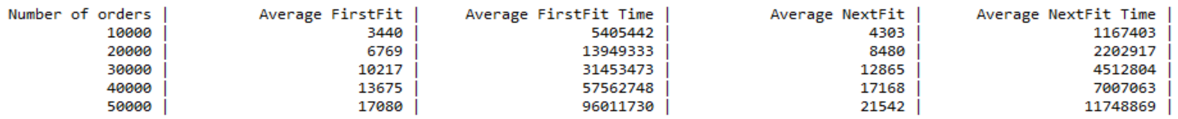
To test that my algorithm performs as expected I have first created a correctness test in a test class. To ensure that my algorithm is correct I first created a list of ropes and orders and then gave each list a set of specific values. From there I manually worked out the results by performing the algorithm out on paper. From this, I set up a check to ensure the result I was getting from my algorithm was the same as the one I worked out. Also in the correctness test, I tested to make sure generators where creating the right amount of ropes and orders. I also tested to make sure that my algorithm performed both first fits and next fist best/worst correctly. After these tests I am certain that my algorithm works as intended.

Performance test

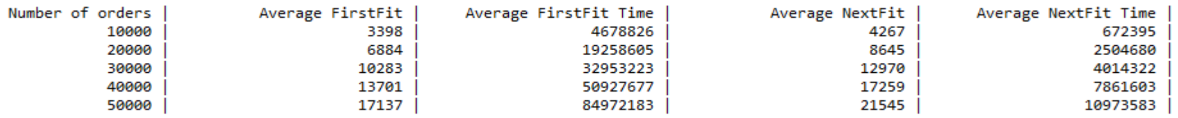
After I was sure that my algorithm worked correctly I went about setting up a way to measure performance. To do this I measured the amount of time it took to perform each algorithm in Nano seconds. This was done in nanoseconds to get an extra layer of preciseness. Earlier test where done in milliseconds, and though this was fine for larger orders (which took more time) the low order numbers came back with the same runtime quite often (2 seconds), this made comparison difficult.

To make sure I tested my algorithm enough. I have written, when called my methods performs 5 test. The first test starts with 10000 orders and incrementing by 10000 each time. For each test I perform 10 reps. Each rep involves me generating a new list of ropes\orders and performing both algorithms. After each algorithm has ran I add the total ropes used and time taken to a value which I later dived to get me an average. My test later prints off each average

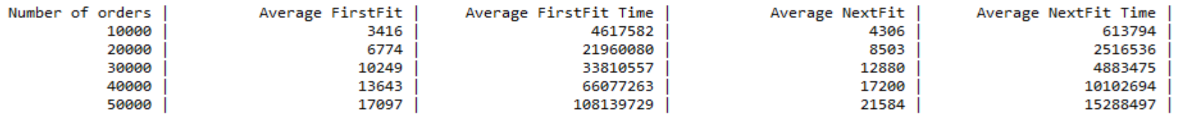
Here are 4 examples of results my test have produced.



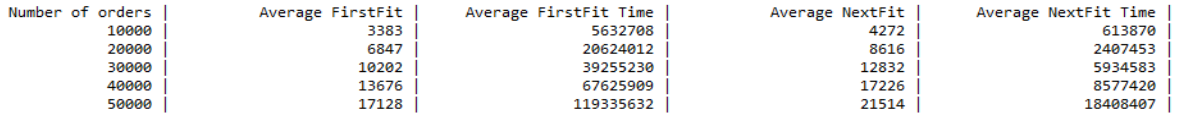
```
Number of orders |         Average FirstFit |    Average FirstFit Time |             Average NextFit |    Average NextFit Time |
           10000 |                     3440 |                  5405442 |                        4303 |                 1167403 |
           20000 |                     6769 |                 13949333 |                        8480 |                 2202917 |
           30000 |                    10217 |                 31453473 |                       12865 |                 4512804 |
           40000 |                    13675 |                 57562748 |                       17168 |                 7007063 |
           50000 |                    17080 |                 96011730 |                       21542 |                11748869 |
```



```
Number of orders |         Average FirstFit |    Average FirstFit Time |             Average NextFit |    Average NextFit Time |
           10000 |                     3398 |                  4678826 |                        4267 |                  672395 |
           20000 |                     6884 |                 19258605 |                        8645 |                 2504680 |
           30000 |                    10283 |                 32953223 |                       12970 |                 4014322 |
           40000 |                    13701 |                 50927677 |                       17259 |                 7861603 |
           50000 |                    17137 |                 84972183 |                       21545 |                10973583 |
```



```
Number of orders |         Average FirstFit |    Average FirstFit Time |             Average NextFit |    Average NextFit Time |
           10000 |                     3416 |                  4617582 |                        4306 |                  613794 |
           20000 |                     6774 |                 21960080 |                        8503 |                 2516536 |
           30000 |                    10249 |                 33810557 |                       12880 |                 4883475 |
           40000 |                    13643 |                 66077263 |                       17200 |                10102694 |
           50000 |                    17097 |                108139729 |                       21584 |                15288497 |
```



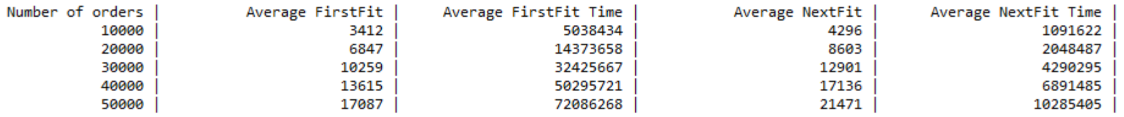
```
Number of orders |         Average FirstFit |    Average FirstFit Time |             Average NextFit |    Average NextFit Time |
           10000 |                     3383 |                  5632708 |                        4272 |                  613870 |
           20000 |                     6847 |                 20624012 |                        8616 |                 2407453 |
           30000 |                    10202 |                 39255230 |                       12832 |                 5934583 |
           40000 |                    13676 |                 67625909 |                       17226 |                 8577420 |
           50000 |                    17128 |                119335632 |                       21514 |                18408407 |
```

Though some values are large than the value in another table. However I can say that this is likely due to the fact that each rep has me generating a new random list of ropes and orders meaning its likely that in those scenarios a large set of ropes/orders are generated in such a way that it's the algorithms worst case. This is an issue that can occur when generating with random number and not something that I can really help. These issues could be caused by other issues in Eclipse that have nothing to do with my code.

These tests have given me reinsurance that my performance works as there wasn't any stand out errors and all my graphs produced a similar outcome (i.e. there was never a point where one of the order increments averages where larger than the next increment test after it).

Graphs

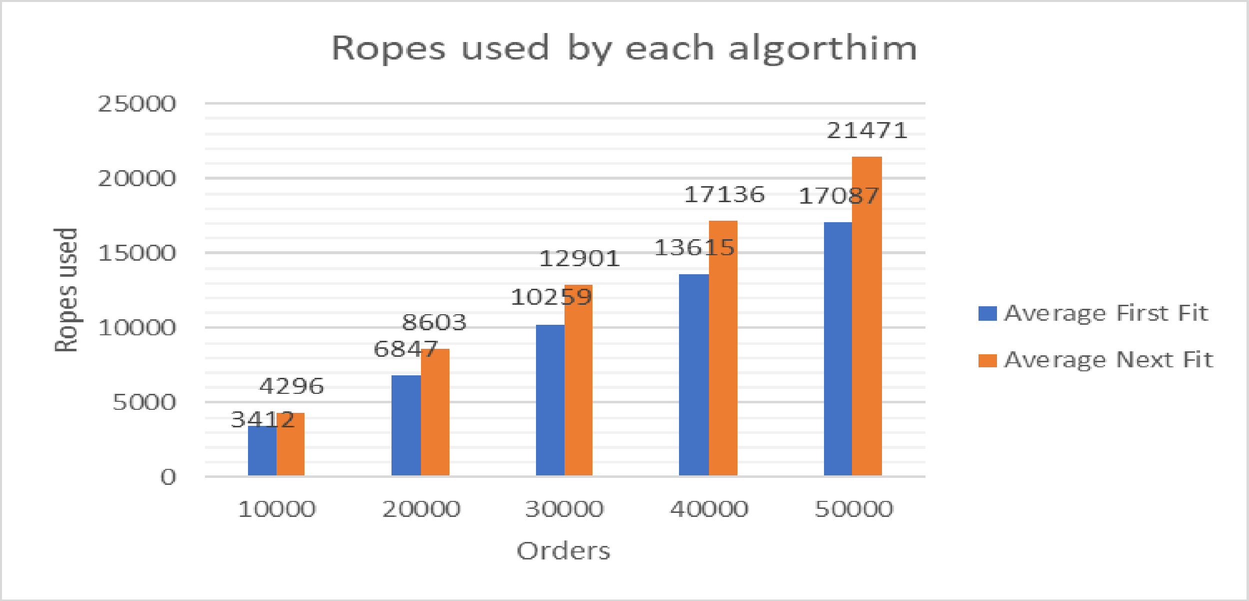
My graphs will be based on the following table. I understand that every run will produce a different set of averages however based on the previous test I believe all the observations I will make will work on every set of tests I have produced and will work on any future test I create.



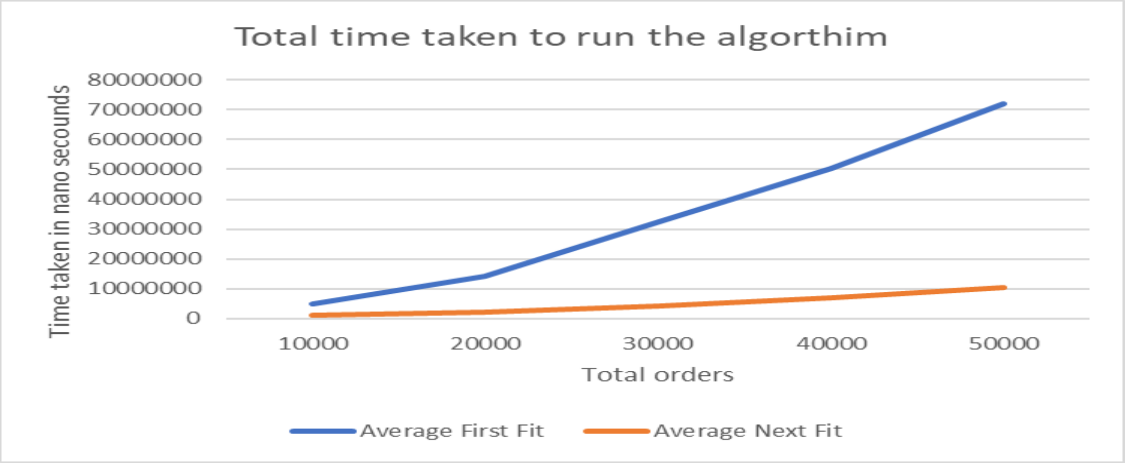
```
Number of orders |         Average FirstFit |    Average FirstFit Time |             Average NextFit |    Average NextFit Time |
           10000 |                     3412 |                  5038434 |                        4296 |                 1091622 |
           20000 |                     6847 |                 14373658 |                        8603 |                 2048487 |
           30000 |                    10259 |                 32425667 |                       12901 |                 4290295 |
           40000 |                    13615 |                 50295721 |                       17136 |                 6891485 |
           50000 |                    17087 |                 72086268 |                       21471 |                10285405 |
```

Average Rope used graphs

Callum Simpson B6030326



Average time used graphs



Observations

In figure one we can see that increasing the number of orders(N) will increase the total number of coils used by roughly the same amount for both algorithms so it can be said that both have a constant increase

When it comes to efficient use of ropes, the First fit will produce the best results. As you can see in graph one the First fit algorithm used less rope in every single test. By comparing the results from each algorithm, I have worked out that First fit is about 20% more efficient than Next Fit.

First fit normal uses ropes equal to 30% of Number of orders

Next fit normal uses ropes equal to 40% of Number of orders

Graph two shows us that First Fit is slower than the Next Fit algorithm.

In First Fit the algorithm the time taken to complete increase more rapidly depending on the size of N. For example, comparing the 10000 order results to the 50000 orders there is 14 times increase which is over an increase of 2N complexity.

Next fit completed a lot faster than the first Fit algorithm. We can see this in graph 2. First fits test with 20000 orders clearly broke the 10000000 nanosecond mark whereas it took until the test with 50000 orders for the next fit algorithm barely pass the 10000000 nanosecond mark.

Conclusion

In conclusion, if you want to be efficient with how many you use ropes use First fits as its around 20% more efficient. However, If you want to complete the orders as quickly as possible then Next fit would be the better option.