1. Selection works by setting the loops beginning position to be the min, then check all the values, which if any of them is smaller then set that one to be the smallest. After the checking is done swap the smallest with the beginning of the loops position. Do the above steps again but a position further. Keep checking until all the values have been accounted in their respective position. An example would be a set of numbers [3, 4, 1, 5, 2] it would set three as the min, then check which value is smaller than three. Since one is smaller, we set that one as the min. Since there is nothing smaller than 1 in the list, we swap the two so the list would look like 1 4 3 5 2. The next step would be 1 2 3 5 4.
2. You pass the two of them into a merge function, where the smallest of the two would be compared then the smallest of the two would be stored into temp array position 1., Depending which is smaller, that ones position is incremented as well as the temporary one. This will continue until one of the two have been entered completely. We then fill the remainder of the spots with the other one.
3. The performance of the selection sort in the list size of 1024 is quite poor as the worst case is 1,048,576 moves. That is quite significant. The merge sort is n times log n which in this case is 3,083 (correct to closet decimal). This amount of moves is a lot quicker than the selection by a significant amount.
4. calAvgMaleAge

while (record is not at end of file)

if person == male

total\_age = total\_age + person age

number = number + 1

end while loop

final = total\_age / number

1. bubble\_sort(A,N)

for I = 0 I < (N -1)

for j = 0 j < (N-1)

if A[j] < A[j+1]

temp = A[j]

A[j] = A[j+1]

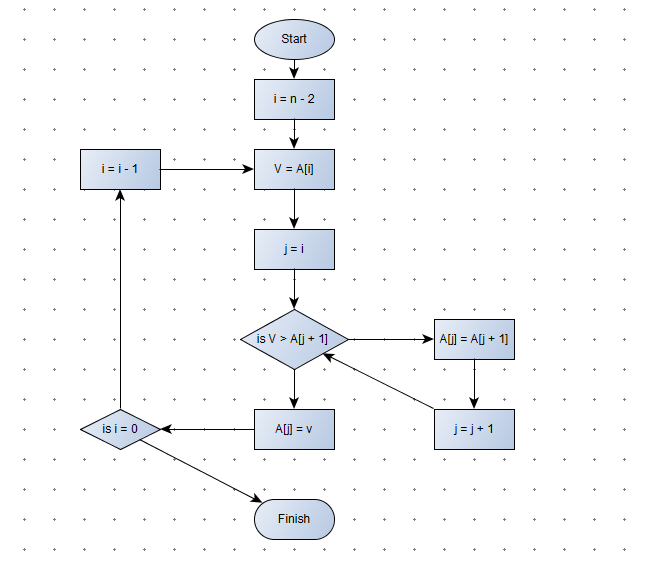
A[j+1] = temp

End if

End inner for

End outer for

1. Insertion sort



1. This algorithm finds all the prime numbers up to a particular limit by starting at two, then marking off all the multiples of two. We then go to the next unmarked number and repeat the process for all remaining unmarked numbers. Go to the next unmarked number and repeat. Do this until we have a list of numbers which can only be divided by themselves or one.
2. Mergesort [7, 11, 4, 9, 6, 15, 2]

Mergesort [7, 11, 4, 9]

Mergesort[7, 11]

Mergesort[7]

Mergesort[11]

Merge[7,11]

Mergesort[4, 9]

Mergesort[4]

Mergesort[9]

Merge[4, 9]

Merge[7, 11, 4, 9]

Mergesort[6,15,2]

Mergesort[6]

Merge[6]

Mergesort[15, 2]

Mergesort[15]

Mergesort[2]

Merge[15, 2]

Merge [6, 2, 15]

Merge[4, 7, 9, 11, 2, 6, 15]

[2, 4, 6, 7, 9, 11, 15]

1. The complexity of Merge sort is N log N
2. The space state search is a method which searches for all the possible routes that are connected to each other. It starts at a initial state and tries to find a goal node by going through a series of stages. This can be used to solve problems or computing games problems. This done through two of many ways, known as depth-first search and breath-first search. The depth-first automatically goes left and keeps checking the current node and its children until it finds a goal node. If there is none then it backtracks to look for more children. The breath-first search checks all the nodes on the current level then goes to the children on the same level. Unlike the depth-first it does not backtrack, due to the fact it checks all on the current level. It finds a goal node it automatically stops, like the depth-first search .It searches from left to right on the same level.
3. A. for I = 0, I < N DO

Min = A[i]

For j = I, j < N DO

If Min < A[j]

Min = A[j]

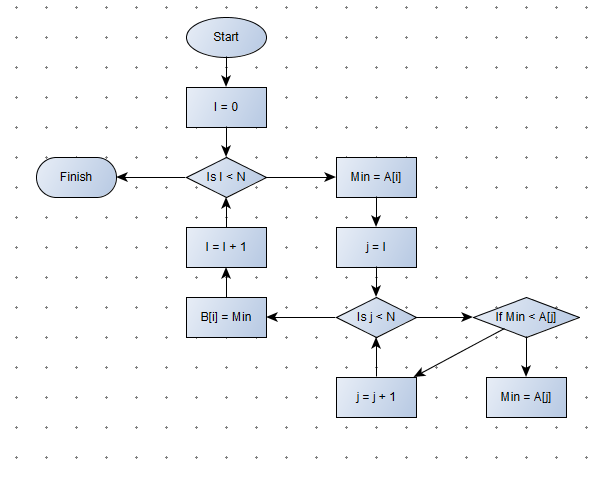
END IF

END FOR

Swap A[i] with Min

END For

B.



1. Look at question one for explanation and look at question 11 for pseudo code.

3 1 4 1 5 9 6 2

1 3 4 1 5 9 6 2

1 1 4 3 5 9 6 2

1 1 2 3 5 9 6 4

1 1 2 3 4 9 6 5

1 1 2 3 4 5 6 9

1. Lets have a list comprised of the following:

1 9 2 8 3 7 4 6 5

Insertion sort will push smaller values into position by looking at the previous value and swapping the two if the smaller value is in a higher position than the bigger number.

* Nine goes up

1 9 2 8 3 7 4 6 5

<- two goes down

1 2 9 8 3 7 4 6 5

We see the two has swapped with nine. It stays there because two is not smaller than one

Let’s go a few steps ahead

1 2 3 7 8 9 4 6 5

1 2 3 7 8 9 **4** 6 5 We are currently looking at four.

We check if the number to the left is bigger, and keep swapping until the number is not bigger

<- <- <-

1 2 3 7 8 9 **4** 6 5 4 is smaller than 9

<- <- <-

1 2 3 7 8 **4** 9 6 5 4 is smaller than 8

<- <- <-

1 2 3 7 **4** 8 9 6 5 4 is smaller than 7

Stop

1 2 3 **4** 7 8 9 6 5 4 is not smaller than 3

The worst case is the entire list is not sorted, making the complexity O(N^2)

The best case is if the entire list is sorted. We do not have to do anything and the complexity is O(N)

The selection sort differs in the way that even if the list was sorted, the runtime would be O(N^2). The insertion sort therefore is either more efficient or just the same at worst, making it a more suitable algorithm.

1. A linear search is searching one by one through each variable. This is not very efficient. However, if the list is not sorted already, then we have no choice but to use that algorithm. The worst case for a linear search would be if the element we are looking for is the last element in the list.

Lets say we are looking through a set of 1000 numbers. It is randomised in a random order from 0 to 999. The number we are looking for is 6. It is possible it would be the 999th number.

1. A. An iterative algorithm is an algorithm that uses a loop to run it. For example, to calculate the average of five numbers inputted we would use a loop to take the numbers in, and have a counter for counting the numbers inputted. After the five numbers have been inputted, divided the total of the five numbers by the counter itself.

A Recursive algorithm is an algorithm that calls itself to do an action. For this to end it must have a base case. Eg. The tower of Hanoi. For this game to work in the most efficient manner, it must be able to do several moves repeatly. This would be ideal if we could call the same function, with a little modification. That is what recursion does.

B. Gcd(72,30) => 72 = 30(2) +12

Gcd(30,12) => 30 = 12(2) + 6

Gcd(12,6) => 12 = 6(2) + 0

C.

Start PROGRAMME

GCD( A, B )

IF B Equals 0 DO

Return A

Else Do

Return GCD( B, (A mod B ))

D.

GCD(72,30)

GCD(30,12)

GCD(12,6)

GCD(6,0)

1. The eight puzzle describes a puzzle with block space of nine, eight randomised. You must move a piece into an unoccupied space to try to solve it. You can only move one piece at a time and you cannot swap any pieces with other ones. The eight piece puzzle is completed once you get to the goal stage.

EG.

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

4 5 6 -> 4 6 -> 4 2 6 -> 4 2 6 -> 2 6

7 8 7 5 8 7 5 8 7 5 8 7 5 8

1. Arrays, linked lists and stuctures/Vectors

Arrays or vectors as you can add or take of through the end of the array, as it is a lot easier to do.

1. A. Start Programme

IF Array is Full Do

Print “Array Full”

Else DO

Place\_available equal Size/2

For I Equal Size, I Greater than Place\_available Do

Array\_spot [I postion] equals Array\_spot [(I postion - 1)]

END FOR

Array\_spot [Place\_available] = value

END PROGRAMME

B.

Start Programme

Count equal number of linked lists

CURRENT Node equals HEAD Node

FOR TIME\_FOR\_COUNT = 0, TIME\_FOR\_COUNT less than (Count / 2) DO

CURRENT equals CURRENT->Next

END FOR LOOP

NEW equals to new\_list address

New\_list.Next = CURRENT->Next

CURRENT->Next = NEW

END PROGRAMME

1. The queue use the breath-first search

QUEUE\_IMPLEMENTATION

Start PUSH()

CURRENT = HEAD

WHILE CURRENT->Next does not equal NULL DO

CURRENT equals CURRENT->Next

END WHILE

CURRENT->Next equals new member

CURRENT equals CURRENT->Next

Input new details

END PUSH()

START POP()

HEAD = HEAD->NEXT

END POP()