**Selection Sort**

In this algorithm, we “select” the smallest element from array and finalise/fix it’s position by putting the element in the 0th position in array. We repeat the same process for next smallest element from remaining elements & by putting it in 1st position in array & so on. Due to this the no of swappings will be considerably lesser than the bubble sort.

i/~~pm~~ ~~j~~ ~~j/pm~~ ~~j/pm~~ j/pm

0 1 2 3 4 temp (10)

// var i is the position where the element is finalised

// var pos\_min is the position of min element in

// remaining array

// var j is used to locate the min element in array

// Logic – Bring the min element to position i

i < noe – 1 (Generalised)

i < 4 (OK but by considering array size = 5)

for (i=0;i <= 3;i++)

{

pos\_min = i;

j < noe (Generalised)

j < 5

for (j=i+1;j<=4;j++)

{

// here if we find a smaller element,

// capture it’s position

if (nos[j] < nos[pos\_min])

pos\_min = j;

}

// swap the elements @ pos i & pos\_min

if (pos\_min != i) // refer to pass III

{

temp = nos[pos\_min];

nos[pos\_min] = nos[i];

nos[i] = temp;

}

}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 40 | 50 | 30 | 20 | 10 |

**I**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 10 | 50 | 30 | 20 | 40 |

i/pm j/pm j/pm j

0 1 2 3 4 temp (20)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 10 | 50 | 30 | 20 | 40 |

**II**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 10 | 20 | 30 | 50 | 40 |

i/pm j j j (loop terminate)

0 1 2 3 4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 10 | 20 | 30 | 50 | 40 |

**III**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 10 | 20 | 30 | 50 | 40 |

i

0 1 2 3 4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 10 | 20 | 30 | 40 | 50 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 10 | 20 | 30 | 50 | 40 |

**IV**