Selection Sort

Implementation:

```
void selection_sort(int ara[],int n)
{
   for(int i=0;i<(n-1);i++)
   {
      for(int j=i+1;j<n;j++)
      {
         if(ara[i]>ara[j])
            swap(ara[i],ara[j]);
      }
   }
   return;
}
```

Analysis:

		cost	time
1.	void selection_sort(int ara[],int n)		
2.	{		
3.	for(int i=0;i<(n-1);i++)	c1	n-1
4.	{		
5.	for(int j=i+1;j<n;j++)	c2	$\frac{n(n+1)}{2}$
6.	{		
7.	if(ara[i]>ara[j])	c3	$\frac{n(n+1)}{2}$
	a. swap(ara[i],ara[j]);	c4	$\frac{n(n+1)}{2}$
8.	}		
9.	}		
10.	return;		
11.	}		

So time function will be

$f(t) = c1*(n-1) + c2* \frac{n(n+1)}{2} + c3* \frac{n(n+1)}{2} + c4* \frac{n(n+1)}{2}$

Best Case occurs when the array is already sorted

1	2	3	4	5

For a already sorted array swap in inner loop will not happen, value of c4 will be 0
so time function will be

f(t) = $\left(\frac{c2+c3}{2}\right) * n^2 +$ $\left(\frac{c2+c3+2c1}{2}\right) * n$ - c1

This function is looking like $f(t) = an^2 + bn + c$, which is a quadric function

And worst case will be occurred when the array is reversely sorted

5	4	3	2	1

For a reversly sorted array time function will be,

f(t) = $\left(\frac{c4+c2+c3}{2}\right) * n^2 +$ $\left(\frac{c4+c2+c3+2c1}{2}\right) * n$ - c1

This function is looking like $f(t) = an^2 + bn + c$, which is also a quadric function

Time complexity

Best Case:

For best case , $f(t) = an^2 + bn + c$

so time complexity $O(n^2)$

Worst case:

For worst case , $f(t) = an^2 + bn + c$

so time complexity $\Omega(n^2)$