Big-O (Cont'd)

Find an asymptotic notation for
$3n+2<=4n$

Where the eq. is less than or equal, so we used a big-O

Since the max power of n is 1, then the solution is : $3n+2=O(n)$

$3n+2<=4n$

For all values $n>=2$

Big-O (Cont'd)

From the following truth table, we note that the notation is verified for n>=2

n	L. H. S	R. H. S.	3n+2<=4n	Verification
1	5	4	5<=4	False
2	8	8	8<=8	True
3	11	12	11<=12	True
4	14	16	14<=16	True

Big-O (Cont'd)

Find asymptotic notation for $10n^2$+4n+2<=$11n^2$

$10n^2$+4n+2=O(n^2)

$10n^2$+4n+2<=$11n^2$

For all values n>=5

<table>
<tr><th>n</th><th>L. H. S.</th><th>R. H. S.</th><th>.</th><th>R. H. S.</th><th>$10n^2$-</th><th>Verification</th></tr>
<tr><td>1</td><td>16</td><td>11</td><td colspan="3">16<=11</td><td>False</td></tr>
<tr><td>2</td><td>50</td><td>44</td><td colspan="3">50<=44</td><td>False</td></tr>
<tr><td>3</td><td>104</td><td>99</td><td colspan="3">99<=104</td><td>False</td></tr>
<tr><td>4</td><td>178</td><td>176</td><td colspan="3">176<=178</td><td>False</td></tr>
<tr><td>5</td><td>272</td><td>275</td><td colspan="3">275<=272</td><td>True</td></tr>
<tr><td>6</td><td>286</td><td>396</td><td colspan="3">396<=286</td><td>True</td></tr>
<tr><td>7</td><td>786</td><td>847</td><td colspan="3">847<=786</td><td>True</td></tr>
</table>

Minimum formula –Omega (Ω)

It is means that the complexities of space and time as possible to be greater than or equal to the minimum limit, and can not be lower than it. The general form as: F(n)= Ω (C g(n))

iff there is two positive constants as (C, n_0)

such that: C >0 , n_0 >= 1 ,

so F(n)>=Cg(n) for all values n>= n_0.

If f(n)=$a_m n^m$+$a_{m-1} n^{m-1}$+………+$a_1 n$ $+ a_0$

Is a polynomial of order m, then: F(n)= Ω (n^m)

Example (1)

If $3n+2$ represents the complexity for a program, find it in Ω?

Sol: $3n+2= \Omega(3n)$ $\longrightarrow$ $3n+2>=3n$

for all values $n>=1$

Note: 3 is a Coefficient of n remain the same,

i. e. $C=3$,

and the formula is represents as $3n+2= \Omega(1)$

Example (1) (Cont'd)

Let : 3n+2>=3n, Find asymptotic notation ?

Sol: Since the polynomial as the form >=,then we use big- Ω, so 3n+2= Ω(3n)

Because 3n+2>=3n for all values n>=1

n	L. H. S.	R. H. S.	3n+2>=3n	Verification
1	5	3	5>=3	true
2	8	6	8>=6	true
3	11	9	11>=9	true

Example (2)

If the asymptotic notation as $10n^2$+4n+2= $\Omega(n^2)$ find the polynomial for this form (where C=11)?

Since the minimum formula is used, so the constant C must be less than or equal the limit bound, so

$$10n^2+4n+2 \geq 10n^2$$

For all values n>=1.

The maximum – minimum Formula (Theta Ө)

The general form is F(n)= Ө (g (n)) iff there are three positive constants (n_0, C_1, C_2) such that:

C_1g(n) ≤ F(n) ≤ $C_2\ g(n)$ for all values n>= n_0

If f(n)=$a_m n^m$+$a_{m-1} n^{m-1}$+………+$a_1 n \quad + a_0$

Is a polynomial of order m, then: F(n)= Ө(n^m)

Example(1)

Prove the correctness of the equation:

$3n+2= \Theta(n)$

Since $3n \le 3n+2 \le 4n$ for all values $n>=2$

Note that the determination of the constant (2) is experimental which verified:

$C_1 g(n) \le F(n) \le C_2\, g(n)$

Example (2)

Let : 3n ≤ 3n+2 ≥ 4n, Find asymptotic notation ?

Sol: Since the polynomial contains upper and lower values ,then we use big- ϴ, F(n)= ϴ (g (n))

Since the constants (C_1, C_2)greater than 0, so it verifies the 1 st condition to determine the value n we use the following table: we note that n>=2

n	L. H. S.	M. S.	R. H. S.	Verification
1	3	5	4	false
2	6	8	8	true
3	9	11	12	true
4	12	14	16	true

Example (3)

Prove the correctness of the following equation:

$10n^2$+4n+2= $\Theta(n^2)$, Since the max coefficient is 10 so, we can put the form as:

$10n^2 \leq 10n^2$+4n+2 $\leq 11n^2$ Since the constants (C_1, C_2)greater than 0, so it verifies the 1 st condition to determine the value n we use the following table: we note that n>=5

n	L. H. S.	M. S.	R. H. S.	Verification
1	10	16	11	false
2	40	50	44	false
3	90	104	99	false
4	160	178	176	false
5	250	272	275	true
6	360	386	396	true

Sorting

- **Review of Sorting:** Sorting is among the most basic problems in algorithm design. We are given a sequence of items, each associated with a given *key value*. The problem is to permute the items so that they are in increasing (or decreasing) order by key. Sorting is important because it is often the first step in more complex algorithms.
- Sorting algorithms are usually divided into two classes, *internal sorting algorithms*, which assume that data is stored in an array in main memory, and *external sorting algorithm*, which assume that data is stored on disk or some other device that is best accessed sequentially.
- We will only consider internal sorting.

Insertion sort

- **Input:** A sequence of *n* numbers $\langle a_1, a_2, \ldots,$ $a_n \rangle$.
- **Output:** A permutation (reordering) $\langle a'_1 \ , a'_2, \ \ldots\ldots, a'_n \ \rangle$. of the input sequence such that $a'_1 \leq a'_2 \leq \ldots\ldots\ldots \leq a'_n$
- The numbers that we wish to sort are also known as the ***keys***.

Insertion sort (cont'd)

Insertion sort

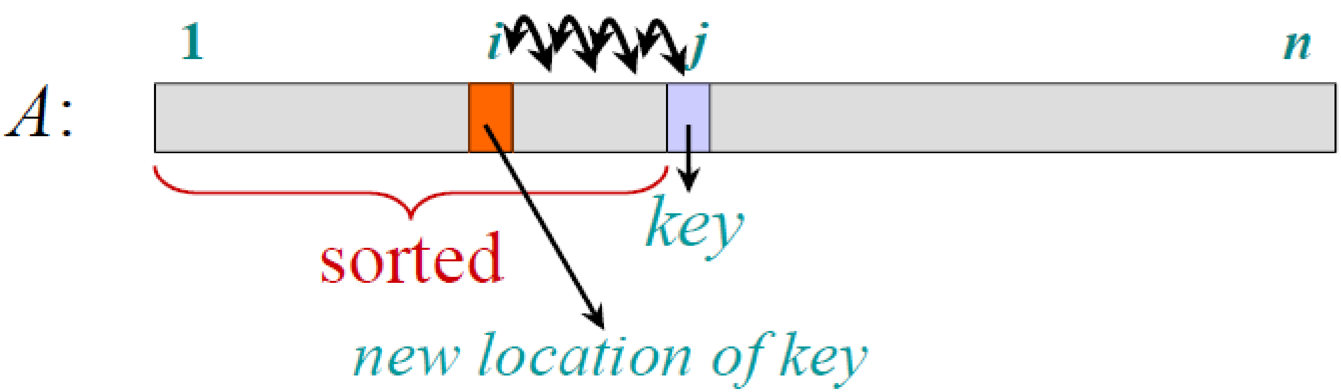
INSERTION-SORT (A, n) $\triangleright A[1 \,.\,.\, n]$

for $j \leftarrow 2$ **to** n

insert key $A[j]$ **into the (already sorted) sub-array** $A[1 \,..\, j\text{-}1]$.

by pairwise key-swaps down to its right position

Illustration of iteration j



Insertion sort (cont'd)

Example of insertion sort

8	2	4	9	3	6	
2	8	4	9	3	6	
2	4	8	9	3	6	
2	4	8	9	3	6	
2	3	4	8	9	6	
2	3	4	6	8	9	*done*

Insertion Sort (Cont’d)

Our pseudo code for insertion sort is presented as a procedure called INSERTIONSORT, which takes as a parameter an array $A[1\,.\,.\,n]$ containing a sequence of length n that is to be sorted. (In the code, the number n of elements in A is denoted by *length*[A].) The input array A contains the sorted output sequence when INSERTION-SORT is finished.

Insertion Sort (Cont'd)

```
INSERTION-SORT(A)
for j = 2 to length[A]
    do key ← A[j]
        //insert A[j] to sorted sequence A[1..j-1]
        i ← j-1
        while i >0 and A[i]>key
            do A[i+1] ← A[i]  //move A[i] one position right
               i ← i-1
        A[i+1] ← key
```

Insertion sort Program

```
#include<iostream.h>
void main()
{
		int i,j,index;
		int a[8];
		for(i=0;i<8;i++)

				cin>>a[i];
		for(j=1;j<8;j++)
		{
				index=a[j];//key
		i=j-1;
		while (i>-1&&(a[i]>index))
		{
				a[i+1]=a[i];
				i=i-1;
		}
		a[i+1]=index;
		}
		for(i=0;i<8;i++)

				cout<<a[i]<<endl;
}
```

Analysis of Insertion Sort

INSERTION-SORT(A)	*cost*	*times*
1. **for** j = 2 to length[A]	c_1	n
2. **do** $key \leftarrow$ A[j]	c_2	n-1
3. //insert A[j] to sorted sequence A[1..j-1]	0	n-1
4. $i \leftarrow j$-1	c_4	n-1
5. **while** i >0 and A[i]>key	c_5	$\sum_{j=2}^{n} t_j$
6. **do** A[i+1] $\leftarrow$ A[i]	c_6	$\sum_{j=2}^{n} (t_j - 1)$
7. $i \leftarrow i$-1	c_7	$\sum_{j=2}^{n} (t_j - 1)$
8. A[i+1] $\leftarrow$ key	c_8	$n - 1$

(t_j is the number of times the while loop test in line 5 is executed for that value of j)

The total time cost $T(n)$ = sum of *cost* $\times$ *times* in each line

$$= c_1 n + c_2(n-1) + c_4(n-1) + c_5\sum_{j=2}^{n} t_j + c_6\sum_{j=2}^{n} (t_j-1) + c_7\sum_{j=2}^{n} (t_j-1) + c_8(n-1)$$

Analysis of Insertion Sort (cont.)

- Best case cost: already ordered numbers
  - t_j=1, and line 6 and 7 will be executed 0 times
  - $T(n) = c_1n + c_2(n\text{-}1) + c_4(n\text{-}1) + c_5(n\text{-}1) + c_8(n\text{-}1)$

    $=(c_1 + c_2 + c_4 + c_5 + c_8)n – (c_2 + c_4 + c_5 + c_8) = cn + c'$
- Worst case cost: reverse ordered numbers
  - $t_j=j$,
  - so $\sum_{j=2}^{n} t_j = \sum_{j=2}^{n} j = n(n+1)/2\text{-}1$, and $\sum_{j=2}^{n}(t_j – 1) = \sum_{j=2}^{n}(j – 1) = n(n\text{-}1)/2$, and
  - $T(n) = c_1n + c_2(n\text{-}1) + c_4(n\text{-}1) + c_5(n(n+1)/2 \text{ -}1) + + c_6(n(n\text{-}1)/2 \text{ -}1) + c_7(n(n\text{-}1)/2)+ c_8(n\text{-}1) =((c_5 + c_6 + c_7)/2)n_2 +(c_1 + c_2 + c_4 +c_5/2\text{-}c_6/2\text{-}c_7/2+c_8)n\text{-}(c_2 + c_4 + c_5 + c_8) =an^2+bn+c$
- Average case cost: random numbers
  - in average, $t_j = j/2$. $T(n)$ will still be in the order of n^2, same as the worst case.

Insertion sort Program

```
#include<iostream.h>
#include<string.h>
void main() {
        int i,j;
        char index[50], a[8][50];
        for(i=0;i<8;i++)
                cin>>a[i];
        for(j=1;j<8;j++)     {
                strcpy(index,a[j]);//key
        i=j-1;
                while (i>-1&&(strcmp(a[i],index)>0))  {
                strcpy(a[i+1],a[i]);
                i=i-1;      }

        strcpy(a[i+1],index);            }
        for(i=0;i<8;i++)
        cout<<a[i]<<endl; }
```