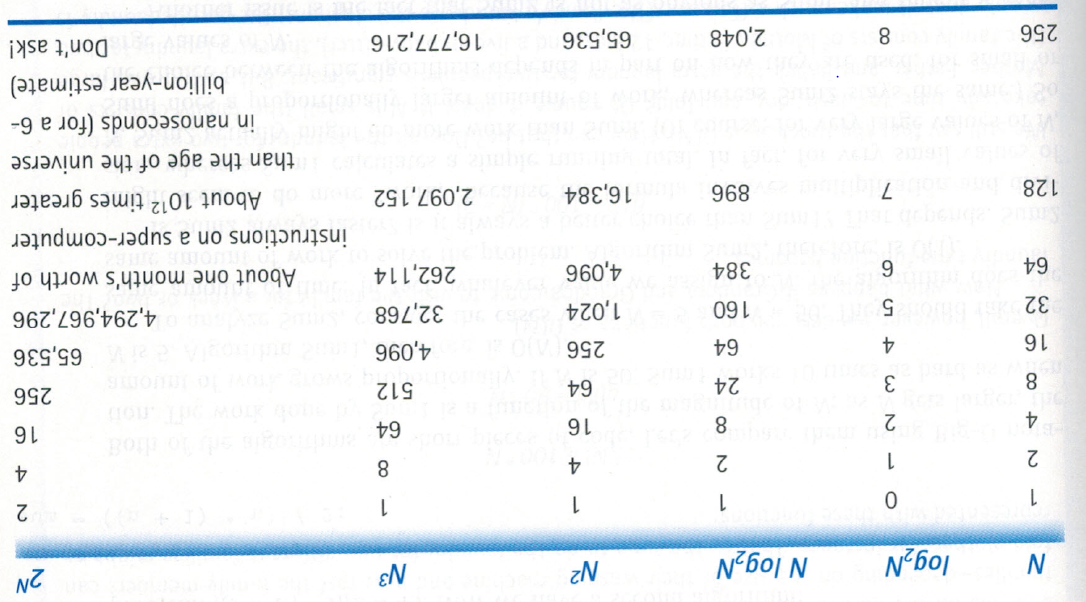
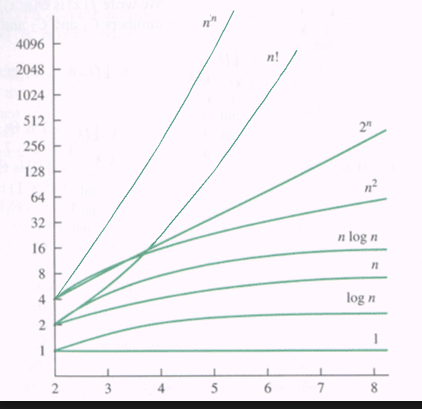
**Big O** notation is used in Computer Science to describe the performance or complexity of an algorithm. **Big O** specifically describes the worst-case scenario, and can be used to describe the execution time required or the space used (e.g. in memory or on disk) by an algorithm





|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 5 | 10 | 11 | 12 | 19 | 15 |

bool search (int item, int a[]){

bool found=false;

for (int i=0; i<n; i++)

if a[i]==item){

found=true;

i=n;

}

return found;

}

**O(1)**

bool IsFirstElementNull(String[] strings)

{

if(strings[0] == null)

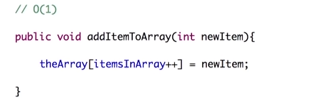
{

return true;

}

return false;

}



**O(N)**

bool ContainsValue(String[] strings, String value)

{

for(int i = 0; i < strings.Length; i++)

{

if(strings[i] == value)

{

return true;

}

}

return false;

}

### O(N2)

bool ContainsDuplicates(String[] strings)

{

for(int i = 0; i < strings.Length; i++)

{

for(int j = 0; j < strings.Length; j++)

{

if(i == j) // Don't compare with self

{

continue;

}

if(strings[i] == strings[j])

{

return true;

}

}

}

return false;

}

**O(N3)**

for(int i = 0; i < strings.Length; i++){

for(int j = 0; j < strings.Length; j++){

for(int j = 0; j < strings.Length; j++){

//Some stuff

}

}

}

**O(log N)**

210 = 1024

210 = x

log21024 = 10

log21024 = x

for(i=1; i <= n; i=2\*i)

{

//some statements...

}