TIME COMPLEXITY:

TYPE1:

For(i=0:i<n:i++) ----n+1

{

Statement; ----n+1

}

Polynomial therm f(n)=n+1

Which is O(n) big or order of n

For nested loops:

For(i=0:i<n:i++) n+1 times

{

For(j=0:j<n:j++)---n\*(n+1)

{

Statements n\*n—n square

}

Time complexity=O(n square)

H.W

What is stack and heep and where we will use it?

1. #impliment a 2D array and rotate the matrix 90 degrees

The languages which support primitive data types are C and JAVA .PYTHON supports non primitive data types.

Rotate matrix clockwise 90 degrees

1. 2 3 1

3 4 4 2

Clue: transpose matrix and then reverse every row

P=0

For(i=1;p<=n;i++)

{

P=p+i;

}

Time complexity O(sqrt(n))

Sum of n numbers =n(n+1)/2

n =sqrt(n)

for two loops:

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

{

For(j=0;j< i ;j++) O(n)

{

Statement:

}

}

Total time complexity o(n square)

Log(n base 2):

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

{

Statements

}

Analysis:

i =1 1 time

i =2 2 times (1\*2)

i =3 4 1\*2\*2=4

so when stopes i>=n

i=2 power k

1. power k>=n

k=log n base 2

time complexity O(log n base 2)

LOG(n):

For(i=n;i>=1;i=i/2)

{

Statements

}

N

n/2

n/2 power 2

.

.

n/2 power k

assume i<1 it stops right?

n/2 power k<1

n/2 power k=1

n=2 power k

k=nlog base 10 so O(log n base 2)

for(i=0;i<n;i++) -----O(n)

for(i=0;i<n;i+2) ----O(n)

for(i=n;i>1;i--)------O(n)

for(i=1;i<n;i=i\*2)----O(log n base 2)

for(i=1;i>1;i=i\*3)-----O(log n base 3)

for(i=n;i>1;i=i/2)----O(log n base 2)

NAME OF THE TIME COMPLEXITY:

1)constant time complexity: O(1)

2)Linear time complexity :O(n)

3) Logarithamic time complexity : O(logn)

4)Quadratic time complexity :O(n square)

5)Exponential time complexity: O(2^n)

* Main is inbuilt in python

SPACE COMPLEXITY

* Space complexity of array of size n,require O(n) space
* 2 D array size n\*n:O(n square)space
* Linear search -O(1)
* Merge sort O(n)
* Depth first search DFS -O(n)
* Breadth first search BFS – O(n)
* Dynamic programming -O(n^2) or O(n\*m)

Examples of O(1)

Constant space complexity O(1)

Def: same amount of space irrespective of the input size n it is called constant s c

Ex 1: sum of array elements

2: linear search

Because space is not depending on values

Space complexity of quick sort is O(log n)

When the space complexity of an algorithm grows proportionally to the input size and a logarithmic factor O(nlogn)-----Merge sort one more loop is added

Polynomial complexity:O(n square):

Space complexity grows proportionally to the square of the input size