1.

Conclusion,

if the array A is **sorted**then TC of while loop will be O(1) because a[j] < a[k] won't evaluate to true.

if Array **A is sorted in reverse order**, then while will run O(n) in total.

loop i and loop j are dependent, hence we will find total complexity of both loops.   
that is O(n^2) as loop j will run 1+ 2 + 3 + 4 +  ..... n-1  

**Time Complexity of this code is O(n^2)**

2.

inner for loop for index k is running n times

for loop with index j is running i+1 times

and outer for loop is running (logn + 1) times if we consider n as power of 2

that means for loop with index j is running for [ (1+1) + (2+1) + (4+1) + ....... + (2logn+1) ] times

so the sum will be = (Geometric series)[ 1 + 2 + 4 + ..... + 2logn ] + [ 1 + 1 + 1 + .... (logn+1) times ]

= 2logn+1 - 1 + logn + 1

= O(n + logn)

now multiplying with O(n) becoz innermost query runs for n times

= O(n2+nlogn) = O(n2)

3.

Ok,  For each i ,   j loop runs  N/i times.  
  
Example for i = 5 ,  
  
It will run 1 ,  6 , 11 , 16 , 21      means( N/5 )  
  
Hence overall :  N/1 + N/2 + N/3 + N/4 ..... N/N.  
  
=N(1+12+13+14+⋯+1n)=N(1+12+13+14+⋯+1n)  
  
=Θ(NlogN)=Θ(Nlog⁡N) (Sum to N term of Harmonic series is Θ(NlogN)

4.

Trick about this question is that variable 'i' is shared and variable j is global.  
  
Therefore, outer loop will only iterate once because on second iteration, i will be 2n due to inner for loop.  
  
Also, while loop will also run for just once. Again, during second iteration, j will be n and hence condition will evaluate to false.  
  
So this code runs for=>  
  
2n(inner for loop) + n(while loop)  
  
=> O(n)

5.

The 2nd loop runs i^2 – i times .

So 2nd and 3rd loop runs   
(i + 2i+3i+4i+…..+ (i^2 - i)

= i \* (1 +2+3+4+……+ (i-1) )

= I \* {(i-1){(i-1)+1} / 2} note : [ n(n+1) / 2 ]

= I \*{ i(i-1)/2 }

= i^2(i-1) / 2

And the outer loop runs 1 to n

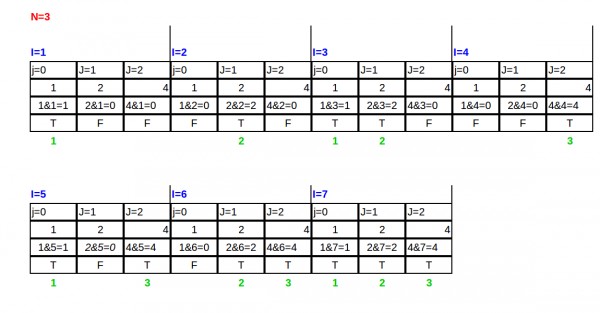
sum_(i=1)^n 1\/2 i^2 (i - 1) = 1\/24 (n - 1) n (n + 1) (3 n + 2)

So below equation gives the exact number of loops

t(n) = 1\/24 n (3 n^3 + 2 n^2 - 3 n - 2)

So big O is O(n^4)

6.



Here the complexity of the code is coming to be O(n2n)

7.

T(n)=Ω(n)T(n)=Ω(n), when nn is a prime number.  
  
T(n)=O(nn−−√)T(n)=O(nn), when n−−√n is a factorial.

The nested loop runs only in case of divisor of N, whenever i dividies N , the nested loop runs N times,  
  
As divisor of any number are not equivalent to N , they may be less then sqrt(N)  
  
And to find number of divisors of any number, it will take sqrt(N) complexity.  
  
So overall lets say N have sqrt(N) divisors  
  
Hence as per my knowledge , the overall complexity, of the above code is sqrt(N) \* N   
  
Not N^2.  
  
Best complexity is o(N)  in case of primes number, when N = prime , it have only two divisors in that case complexity is linear.

8.

N+n/2 +n/3+n/4+n/5+n/6+n/7+….+n/logn

= n( ½ + ¼ + ¼+ 1/5 + 1/6 +….1/logn) = n\* log log n

9.

For solving this, think how many iterations out of NN are allowed to pass through IF block . Only 11iteration can pass the IF block and that particular iteration will run KK loop NN times .

Hence,JJ loops NN times but only 11 iteration of it, is allowed to pass through and execute KK loop NN times ,hence, only that 11 iteration will be counted .

**Final time complexity =**N∗1∗N=O(N2)N∗1∗N=O(N2)**.**

10.

T(N)=Θ(N)

11.

inner most loop runs 5log15 n times in its one iteration

it runs like 1,15,152........15k

where 15k=n5 ==>k=5 log15n

when I=1 and j=1 then inner most loop runs **5 log15n** times

when I=2 and j=1,2 then inner most loop runs 2(**5 log15n)** times

when I=3 and j=1,2,3 then inner most loop runs 3(**5 log15n)** times

when I=n and j=1,2,3..n then inner most loop runs n(**5 log15n)** times

total time will be **5 log15n+2(5 log15n)+3(5 log15n)+4(5 log15n)....+n(5 log15n)**

**5 log15n(1+2+3+4........n)**

**5 log15n \* n\*(n+1)/2**

**so time complexity is O(n2 log15n)**

12.

We need to calculate upper bound of this series summation.S =  1k+2k+3k+−−−−−−+nk1k+2k+3k+−−−−−−+nk.

for k = 1 S = n∗(n+1)2=O(n2)n∗(n+1)2=O(n2)

for k = 2 S = n∗(n+1)(2∗n+1)6=O(n3)n∗(n+1)(2∗n+1)6=O(n3)

for k = 3 S = (n∗(n+1)2)2=O(n4)(n∗(n+1)2)2=O(n4)

--------------------------------------------------------

In this question ans should be  = O(nk+1)

13.

See first loop will run **O(n**) time

Middle loop **condition j<i\*i**so it will run**O(n2 ) time** (since i is running n time)

Last loop **condition *k<j***so i will run **O( n2 ) time** (j is running n2 time)

**So total time complexity=O(n5)**

14.

if n>=1   
then j will never change and always be true.  
  
so i will run every time j =n/2 time   
so it will be infinte loop program will run till stack overflow.

15.

0(n^3) because the internal loop runs n^3 times and now it does not satisfy the condition of 1st and 2nd loop so it comes out of the loop

16

Counting the number of times the inner most loop gets executed equals,

n-2 + n-3 + n-2 + ....1   
+n-3 + n-2 + .... + 1   
+   
...   
+ 3 + 2 + 1

= (n-2)(n-1)/2 + (n-3)(n-2)/2 + ... + 3\*4/2

= (n2 - 3n + 2 + n2 - 5n + 6 + .... )/2

= O(n3) as there are n terms and n2 is the dominating term in each of them.

17.

Answer = **option B**

Worst Case : T(n)=O(n−−√)T(n)=O(n)

Best Case : When nn is an even number body of forfor loop is executed only 1 time (due to "return 0" inside if) which is irrespective of nn. ∴T(n)=Ω(1)