Algo 1:

void printMorePrimes(int N) {  
 vector<bool> nums(N+1,true); 1  
 for (int j = 2; j < N; j++) { N-2  
 for (int k = 2\*j; k <= N; k += j) { N/2  
 nums[k] = false;  
 }  
 }  
 for (int i = 2; i <= N; i++) { N

if (nums[i]) cout << "Prime: " << i << endl;  
 }  
}

Time Complexity:

The loops loop through N-2 times, N/2 times, and N-2 times respectively

I believe that the –2 on the first and last N’s is negligible so we can treat them as N which gives us N \* (N/2) + N making N^2/2 + N. The /2 is a constant so we will ignore it giving us N^2 + N and since the + N is negligible compared to N^2 we can ignore that as well giving us a result of O(N^2)

Space:

This algorithm has a vector of N+1 size. This could potentially take up a large amount of space if you wish to find a very large set of primes.

Algo 2:

void printPrimes(int N) {  
 for (int i = 2; i <= N; i++) { N-2  
 bool divisorFound{ false };  
 for (int j = 2; j < i; j++) { N-2  
 if (i % j == 0) divisorFound = true;  
 }  
 if (!divisorFound) cout << "Prime: " << i << endl;  
 }  
}

Time Complexity:

The loops both run N-2 times and since the –2 is negligible we can say they run through N times. This makes the time complexity of the algorithm O(N^2)

Space: this algorithm only stores a small amount of data being I and J so it doesn't impact memory much

Comparison of the 2:

While both algorithms have the same time complexity of O(N^2) the firs one will still run faster than the second. However, when it comes to space taken up the first algorithm is much worse as it uses a vector of size N+1 this means that you would be taking up a very large amount of space if you wanted to find a large prime number such as 35742549198872617291353508656626642567. With this in mind, I feel that it would be better to use the second algorithm if you were working with tighter memory constraints on an embedded system or something else of the sort. I also think it’s important to note that the second algorithm makes many more logical operations with its main if-statement being at the center of its nested for loops (runs N^2 times) while the first algorithm has the if-statement in a sperate for loop that runs N times. I am unsure, however, if this would significantly impact run time.