**Q1**

**Algorithm MultipleX(A, n,X)**

**Input array A of n integers and Integer X we want to check**

**Output the string that show which index is multiple of X**

**for i ← 0 to n − 1 do**

**if A[i] mod X =0 then**

**display “** **Index i with value A[i]”**

**end for**

**return**

**a) use for loop,if mod X=0,then its is multiple of X,then can System.out.println (“Index “+I+” with value “+A[i]+”**

**”)**

**b) >=n, the for loop is from i=0 to i=n-1,and there is no nested loop in the for loop,other sentence are just O(1)**

**c)**  **<= n，Θ(n) is n,then O(n) is>=,Ω(n)is <=**

**d) O(1),there is no recursion so we don’t need ADT like stack to hold other method**

**(2)**

**Algorithm MultipleX(A, n,X)**

**Input array A of n integers and Integer X we want to check**

**Output the string that show which index is multiple of X**

**T<-an empty array with length n+1// queue with length n can contain only n-1 element**

**f<-0,r<-0**

**for *i* ← 0 to *n-1* do**

**(*T*[*i*] ← *A*[*i*]**

**r<-r+1)**

**end for**

**while f !=r do**

**if T[f] mod X =0 then**

**display “Index f with value T[f]”**

**f<-f+1 //deq()**

**end if**

**end while**

**return**

1. **create a new array with n+1 length, see the new array as queue,f=r=0,then copy the old array into new array,with r++ n times. the rest thing is just deq until the queue is empty**
2. **>=n, the for loop is from i=0 to i=n-1,and there is no nested loop in the for loop**

**the while loop will repeat n times and there is no nested loop in the while loop**

1. **<= n，Θ(n) is n,then O(n) is>=,Ω(n)is <=**
2. **0(n),we use the new array as queue ADT to store the data in old array**

**2.**

*1.* ***for*** *i=0 to n-1* ***do n****2. Res[i]=0 n****3. end for****4.* ***for*** *i=0 to n-2* ***do n-1****5.* ***for*** *j=i+1 to n-1* ***do n-1+n-2+….1****6.* ***if*** *A[i]*≤*A[j]* ***then n-1+n-2+….1****7. Res [j]= Res [j]+1*

***8. else****9. Res[i]= Res[i]+1 step7+9=* ***n-1+n-2+….1******10. end if   
11. end for  
12. end for****13.* ***for*** *i=0 to n-1* ***do n****14. B[Res [i]]= A[i] n****15. end for****16.* ***Return*** *B 1*

1. **O(n^2), Ω(n^2 ) ,the biggest step is the for loop from 5 to 11, n-1+…1=n\*(n-1)/2=O(n^2)**
2. **From 1-3,wil create a array Res (0,0,0,0,0,0,0)  
    when i=0, for j=1 to 6 ,if A[0]<=A[j],res[j]++,else res[i]++**

**Res[i]++,res[j]++,res[i]++,res[i]++.res[i] ++,res[i]++, res[0]=5,res[2]=1**

**I=1,FOR J= 2 TO6, res[1]++,res[2]++,res[3]++,res[5]++,res[6]++. Res[1]=1,res[2]=2,res[3]=1,res[5]=1,res[6]=1**

**I=2. For j=3 to 6, res[2]++\*4,res[2]=6**

**I=3,for j=4 to6, res[3]++,res[5]++,res[6]++. Res[3]=2,res[5]=2,res[6]=2**

**I=4 ,for j= 5 to 6, res[5]++,res[6]++,res[5]=3,res[6]=3**

**I=5, ,for j=6, res[6]++, res[6]=4**

**For 4-12, res(5,1,6,2,0,3,4)**

**For 13-15,**

**B[RES[0]]=A[0] ->B[5]=88,B[1]=12,B[6]=94,B[2]=17,B[0]=2,B[3]=36.B[4]=69**

**B(2,12,17,36,69,88,94)**

**FOR 16, RETURN B(2,12,17,36,69,88,94)**

1. **It sort the array from small to big.**

**With the nested for loop , the element reverse[i] shows the rank in size of A[i] in whole array A. Why? It compares 2 element with each other once, which A[i] is bigger, reverse[i]++，why j=i+1 in second for loop? Because we only want to compare once, if it start with j=0, then we will repeat.**

**At the last for loop, we just input the element of A into B with the order with reverse[i]**

**//it is a** **selection-sort**

1. **Yes, now time complexity is n^2, we can just use heap sort, which time complexity is nlogn**
2. **No，both selection-sort and heapsort just need O(1) for exchange**

**3.   
1,. Θ(n)=(logn)^3 <g(n), fn is** O(g(n))

**2. Θ(n)=n^1.5 >g(n), fn is** Ω(g(n))

**3. Θ(n)=n > g(n) fn is** Ω(g(n))

**4. Θ(n)=n^0,5 >g(n) fn is** Ω(g(n))

**5. Θ(n)=2^n! >g(n) fn is** Ω(g(n))

**6. Θ(n)=2^10n <g(n) fn is** O(g(n))

**7. Θ(n)=(n^n)5 <g(n) fn is** O(g(n))