1. (A) 寫一個分治演算法來解決一個給定的數值集合的求秩(rank finding)問題，並進行複雜度分析。  
   Algorithm 1D\_rank(S,p,r)   
   Input: n個一維的點集合S，n>=1  
   Output: S集合中所有的rank  
   if(r=p)   
    return S  
   end if  
   mid = (p+r)/2  
   1D\_rank(S,p,mid)  
   1D\_rank(S,mid+1,r)  
   merge(S,p,mid,r)

return S  
  
merge(S,p,mid,r)  
{

n1<-mid-p+1  
 n2<-r-mid  
 for i=1 to i=n1  
 L[i] = S[p+i-1]  
 end for  
 for j=1 to j=n2  
 R[j] = S[mid+j]

end for  
 L[n1+1] = inf  
 R[n2+1] = inf

i<-1 , j<-1  
 for k=p to k=r  
 if( L[i]<=R[j])  
 A[k]= L[i]  
 i++  
 else   
 A[k] = R[j]  
 j++  
 end if  
 end for  
 return A }  
  
count\_rank(S,n)  
for i= 0 to i=n-1  
 S.rank = i  
end for  
 retrun S  
時間複雜度: (1)1D\_rank + merge:   
 T(n) = 2T(n/2)+n  
 =2kT(n/2k)+kn  
 n=2k, k=logn ,T(1) = 1  
 ->T(n) = nlogn  
 (2) count\_rank:  
 共作n次  
 total: nlogn+n = O(nlogn)

1. (B) 寫一個分治演算法來找出一群在X軸上的點中的最近點對(closest pair of points on X-axis)，並進行複雜度分析。  
   Algorithm closesetpoint(S,l,r)  
   Input: n個一維的點集合S，n>=2  
   Output: S集合中的最近點對  
   sort(S,S+n)// S中集合由小排到大  
   closesetpoint(S,l,r)  
   {  
    mid = (l+r)/2  
    if r-l==1  
    return abs(S[r]-S[l])  
    end if  
    dmin = min(closesetpoint(S,l,mid) , closestpoint(S,mid,r))  
    return dmin  
   )  
   時間複雜度: closestpoint:T(n) = 2T(n/2)+c  
    =2kT(n/2k) +c’= 2kT(1)+c’ = O(n)  
    sort: O(nlogn)  
    total: c1nlogn + c2n = O(nlogn)
2. (C) 寫出一個分治演算法來解決最長單調遞增連續元素子序列(the longest monotonically increasing consecutive element subsequence) 問題，並分析演算法的時間複雜度，並進行複雜度分析。  
   Algorithm: longestsub(A,l,r)  
   Input: n個整數 array A  
   Output: 最長單調遞增連續元素子序列 array C  
   mid = (l+r)/2  
   if( l = r)  
    return A[l].size  
   LIS = RIS = 1  
   for i=mid to i=l+1  
    if(A[i]>A[i-1])  
    LIS++  
    else   
    break  
    end if  
   end for  
   for i=mid+1 to i=r-1  
    if(A[i+1]>A[i])  
    RIS++  
    else   
    break  
    end if  
   end for  
   if( A[mid] <=A[mid+1] )  
    MMIS = LIS+RIS  
   else  
   end if  
   MLIS = longestsub(A, l, mid)  
   MRIS = longestsbu(A, mid+1, r)  
   return max( MLIS, MRIS, MMIS)  
   時間複雜度: T(n) = 2T(n/2)+n  
    =2kT(n/2k)+kn (2k = n , k=logn)  
    = nT(1) + nlogn  
    =O(nlogn)
3. (D) 給定一個包含n個正或負整數的一維陣列A，以分治演算法在O(n log n)時間複雜度求出連續相鄰元素的最大總和。你必須進行複雜度分析以驗證你的結果。  
   Algorithm maxsubsum(A,l,r)  
   Input: n個整數 array A  
   Output: 連續相鄰元素的最大總和  
   mid = (l+r)/2  
   if(l = r)  
    return A[l]  
   LBS = RBS = 0  
   for i = mid to i = l  
    LBS +=A[i]  
    if(LBS > MLBS)  
    MLBS = LBS  
    end if  
   end for  
   for i = mid+1 to i = r  
    RBS +=A[i]  
    if(RBS > MRBS)  
    MRBS = RBS  
    end if  
   end for  
   MMS = max(MLBS,MRBS, MLBS+ MRBS)  
   MLS = maxsubsum(A, l, mid) // 左半最大sum  
   MRS = maxsubsum(A, mid+1, r) // 右半最大sum  
   return max(MLS, MRS, MMS)  
   時間複雜度: T(n) = 2T(n/2)+n  
    =2kT(n/2k)+kn (2k = n , k=logn)  
    = nT(1) + nlogn  
    =O(nlogn)  
   http://emn178.pixnet.net/blog/post/88907691-%E6%9C%80%E5%A4%A7%E5%AD%90%E5%BA%8F%E5%88%97(maximum-subarray)
4. (E) 在選取刪尋演算法中，若每個分割子集改為3個元素，則其最壞狀況時間複雜度是否依然為O(n)，為什麼?  
   step:5 刪掉的部分為 2xceil(n/2)/3n >= 1/3(此n為分割大小) ->每次會刪掉n/3的部分  
   T(n) = T(2n/3)+T(n/3)+cn  
    <=T(2n/3 + n/3)+cn  
    =T(n)+cn  
   T(n) = T(n)+ cn2 >O(n) (遞迴n次後)
5. (F) 在選取刪尋演算法中，將每個分割子集改為6個元素後重新分析其時間複雜度。  
   step1:分割成子集6個元素: O(n)  
   step2:直接排序: O(n)  
   step3:找出每個子集中，中位數集合之中位數p: O(n/6)  
   step4:以p分為大於、小於、等於三個集合: O(n)  
   step5:判斷第k小元素在哪個集合中(若在S1、S3跳回step1 ,在S2則輸出p):O(3n/4)  
   T(n) = T(3n/4)+T(n/6)+cn  
    <=T(3n/4 + n/6)+cn  
    =T(11n/12)+cn  
    =cn+ 11/12cn + T( (11/12)2n)  
    =[ 1-(11/12)p+1 / 1-(11/12) ]cn <=12cn = O(n)  
   \*(11/12)p+1n<=1  
   http://blog.csdn.net/xsc\_c/article/details/8922191