James Stallkamp

CS 325 HW 2

1)

a)

b)

c)

using the master method

Case 3 then if the regularity condition holds for f(n)

Therefore according to the master theorem case #3,

d)

Since

We have

2

1. The input is first sorted so that if the list is of length two then the list can be immediately returned.
2. No. when N 4 the list would be divided into unequal subparts and then the sort wouldn’t work.

3

PseudoCode:

Function quaternarySearch(A, key, start, end)

If(start > end)

Return false

Piece1=start+(end-start)/4

Piece2=start+ 2 \*(end-start)/4

Piece3=start+ 3 \*(end-start)/4

If(A[piece1] == key)

Return(true)

Else if(A[piece2] == key)

Return(true)

Else if(A[piece3] == key)

Return(true)

Else If(key < A[piece1])

Return quaternarySearch(A, key, start, piece1 -1)

Else if(A[piece2] <key & A[piece3] >key)

Return quaternarySearch(A, key, piece1+1,piece2 -1)

Else if(A[piece3] <key)

Return quaternarySearch(A, key, piece2+1,piece3 -1)

Else

Return quaternarySearch(A, key, piece3+1,end)

Recurrence

Using the master method. A =1 b=4

Comparing F(n) so case 2 meaning T(n) =

Compare

Both Binary and quaternary search are

4

Min\_Max(a)

If |A| =1 then return min=max=A[0]

Divide A into two equal subsets A1 and A2

Return(min,max)

Recurrence:

Solution:

5

Solution:

The array is split into two equal pieces and then the function is called twice recursively to find the majority elements of both sub pieces. This is repeated until the sublists are one element long and then they are counted back up.

getMajorityElement(A[0….n])

n = |A|

if n = 1, return a[0]

k = n/2

leftSub = getMajorityElement(a[0…k])

rightSub = getMajorityElement(a[k…n])

if leftSub = rightSub

return leftSub

lCount = getFrequency(a[0…n], leftSub)

rCount = getFrequency(a[0…n], rightSub)

if lCount > k+1

return leftSub

else if rCount > k+1

return rightSub

else

return NO MAJORITY ELEMENT

recurrence is

Solution is