function binary\_search(list, target):

left = 0

right = length(list) - 1

while left <= right:

mid = (left + right) // 2

if list[mid] == target:

return mid

elif list[mid] < target:

left = mid + 1

else:

right = mid - 1

return -1 **BEST CASE:1, WC: log n**

function QUICKSORT(ARRAY, START, END)

if START >= END then

return

end if

PIVOTINDEX = PARTITION(ARRAY, START, END)

QUICKSORT(ARRAY, START, PIVOTINDEX – 1)

QUICKSORT(ARRAY, PIVOTINDEX + 1, END)

end function **BEST TC = n log n  
 WORST TC = n^2**

function selectionSort(array):

for i = 0 to size - 1

minIndex = i

for j = i + 1 to size

if array[j] < array[minIndex]

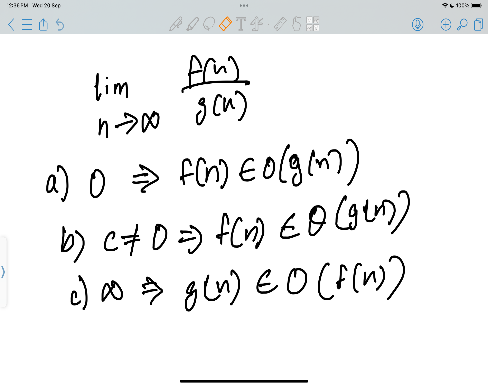
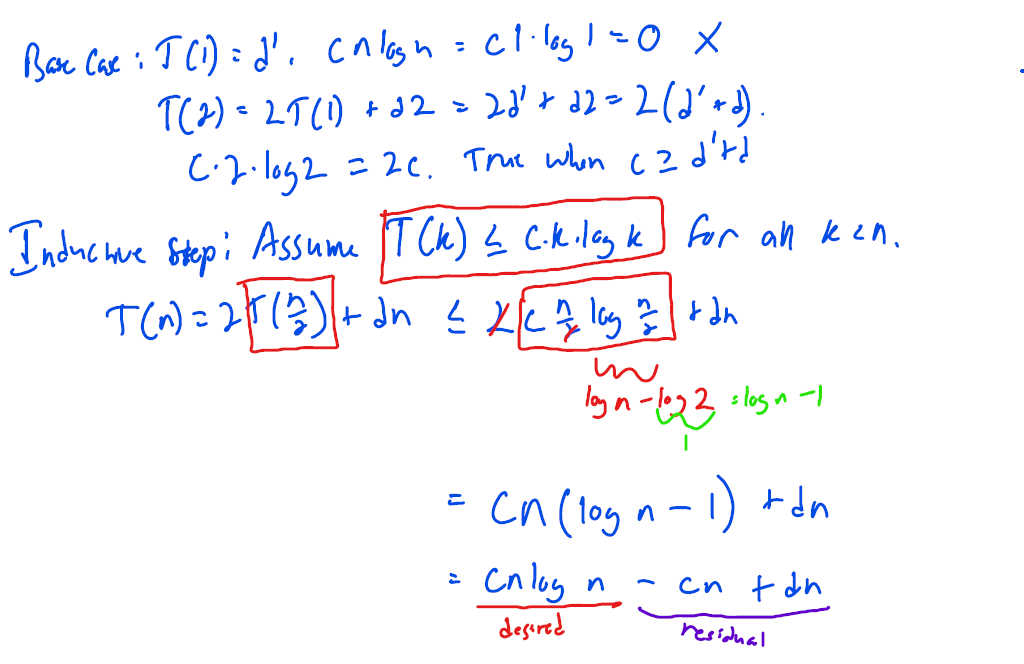
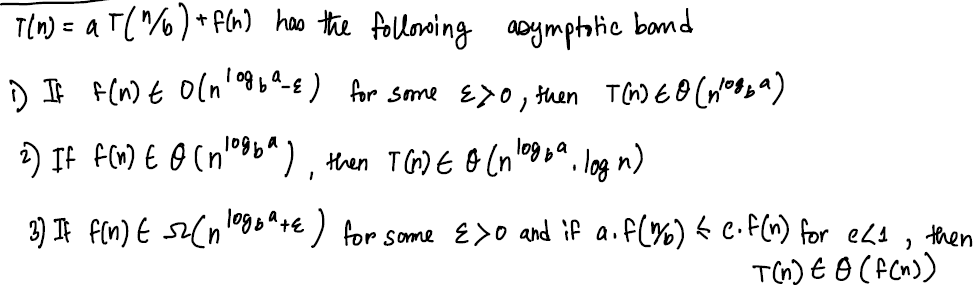
minIndex = j

swap array[i] with array[minIndex]

**BEST TC = n^2  
WORST TC = n^2**

A screenshot of a whiteboard with black text

Description automatically generatedA close-up of a math problem

Description automatically generatedA math equations on a white background

Description automatically generated

P[E] = (1/6)\*1+(5/6)\*0  
Ev[X=5]=(1/6\*5)

Inductive steps: Initialization; iteration ( before and after same ); termination

b^(log\_n b)=n

function PARTITION(ARRAY, START, END)

PIVOTVALUE = ARRAY[END]

PIVOTINDEX = START

loop INDEX from START to END

if ARRAY[INDEX] <= PIVOTVALUE

TEMP = ARRAY[INDEX]

ARRAY[INDEX] = ARRAY[PIVOTINDEX]

ARRAY[PIVOTINDEX] = TEMP

PIVOTINDEX = PIVOTINDEX + 1

end if

end loop

return PIVOTINDEX – 1

function MERGESORT(ARRAY, START, END)

if END - START + 1 == 1 then

return

end if

if END - START + 1 == 2 then

if ARRAY[START] > ARRAY[END] then

TEMP = ARRAY[START]

ARRAY[START] = ARRAY[END]

ARRAY[END] = TEMP

end if

return

end if

HALF = int((START + END) / 2)

MERGESORT(ARRAY, START, HALF)

MERGESORT(ARRAY, HALF + 1, END)

MERGE(ARRAY, START, HALF, END)

end function **TC = n log n**

procedure insertionSort( A : array of items )

int holePosition

int valueToInsert

for i = 1 to length(A) inclusive do:

valueToInsert = A[i]

holePosition = i

while holePosition > 0 and A[holePosition-1] > valueToInsert do:

A[holePosition] = A[holePosition-1]

holePosition = holePosition -1

end while **BEST TC = n ; WORST TC = n^2**

A[holePosition] = valueToInsert

end for

end procedure