Divide array to 2 parts and compare mon & not both to get max & min of whole thing  MaxMin (arr arr size)  if arr size = 1:  return elem as both mox armin  else; f arr size = 2  Compare to determine min & max  return max & min  else  recur for max & in in of left half  recur for righ half  one and max comparison  one finds min comparison  one finds min (omparison  one finds min (omparison  one finds min (omparison  one finds min (omparison  return max (in in)  B)  T(n) = T(floor(1/2)) + T(ceil (1/2)) + 2  T(2) = 1 T(1) = 0  T(n) = 2T(1/2) + 1  T(n) = 37/2 - 2  * uses n steps to get max/min cause it  traverses the array linearly		
Divide array to 2 parts and compare max & not both to get max & min of whole thing  MaxMin (arr arr size)  if arr size = 1:  return elem as both max & min  else; if arr size = 2  compare to determine min & max  return max & min of left half  recor for righ helf  one Chas max comparison  one finds min co	Donblom H	
Max Min (ass arr size)  if arr size = 1:  return elem as both max a min  else if assize = 2  (ompare to determine min & max  return max a min  else  recor for max in min of left half  recor for righ half  One that max comparison  one finds min comparison  return max a min  B)  T(n) = T(floor(1/2)) + T(ceil(1/2)) + 2  ()  T(2) = 1 T(1) = 0  T(n) = 2T(1/2) + 2  ** uses n steps to get max/min cause it  braverses the array linearly	1) Cociety (FI)	
Max Min (ass arr size)  if arr size = 1:  return elem as both max a min  else if assize = 2  (ompare to determine min & max  return max a min  else  recor for max in min of left half  recor for righ half  One that max comparison  one finds min comparison  return max a min  B)  T(n) = T(floor(1/2)) + T(ceil(1/2)) + 2  ()  T(2) = 1 T(1) = 0  T(n) = 2T(1/2) + 2  ** uses n steps to get max/min cause it  braverses the array linearly	A) Duil a see a la se	1 .
Max Min (ass arr size)  if arr size = 1:  return elem as both max a min  else if assize = 2  (ompare to determine min & max  return max a min  else  recor for max in min of left half  recor for righ half  One that max comparison  one finds min comparison  return max a min  B)  T(n) = T(floor(1/2)) + T(ceil(1/2)) + 2  ()  T(2) = 1 T(1) = 0  T(n) = 2T(1/2) + 2  ** uses n steps to get max/min cause it  braverses the array linearly	Civilae allay to 2 parts and compare mon	E, mh
Max Min (ass, associate)  if ast-size = 1:  return elem as both max a min  else; f ast size = 2  (ompare to determine min & max  return max a min  else  recor for max a min of left half  recor for righ half  One finds max comparison  one finds max comparison  return max a min  B)  T(n) = T(floor(1/2)) + T(ceil(1/2)) + 2  ()  T(2) = 1 T(1) = 0  T(n) = 2T(1/2) + 1  T(n) = 3/2 - 2  * uses n steps to get max/min cause it  traverses the array linearly	of Doth to get max a, min of whole thing	
return elem as both max a min  else; f art size = ?  (ompare to determine min & max  return max a min  else  recur for max & min of left half  recur for righ half  One Olds max comparison  one Linds min comparison  one Linds min comparison  return max a min  B)  T(n) = T(floor (//2)) + T(ceil (//2)) + ?  ()  T(2) = 1 T(1) = 0  T(n) = 2 T(1/2) + ?  **Wises of steps to get max/min cause it traverses the array linearly		
return elem as both max a min  else; f art size = ?  (ompare to determine min & max  return max a min  else  recur for max & min of left half  recur for righ half  One Olds max comparison  one Linds min comparison  one Linds min comparison  return max a min  B)  T(n) = T(floor (//2)) + T(ceil (//2)) + ?  ()  T(2) = 1 T(1) = 0  T(n) = 2 T(1/2) + ?  **Wises of steps to get max/min cause it traverses the array linearly	Max Min (ass, arr_size)	
return elem as both max a min  else; f art size = ?  (ompare to determine min & max  return max a min of left half  recor for max a min of left half  recor for righ half  one and max comparison  one and max a min comparison  return max a min  ()  ()  ()  ()  ()  ()  ()  ()  ()  (	it arr-size =1:	
else; f ar size = ?  (ompare to determine min & max  return max & min  else  return for max & min of left half  recor for right half  one and max comparison  one ainds min comparison  return max amin  B)  T(n) = T(floor(//2)) + T(ceil (//2)) + ?  ()  T(2) = 1 T(1) = 0  T(n) = 2 T(1/2) + 1  T(n) = 3 //2 - 2  * uses n steps to get max/min cause it  traverses the array linearly	return elem as both max a min	
Compare to determine min & max return max & min else  recor for max & min else  recor for migh helf  one that max comparison one finds min comparison return max & min  (2) = 1 T(1) = 0  T(2) = 1 T(1) = 0  T(n) = 2T(1/2) + 1  T(n) = 31/2 - 2  * uses n steps to get max/min cause it traverses the array linearly	2 2 2 2 2	
Tetur Max 9min  else  recor for max 2 min of left half  recor for righ half  One Ands max comparison  one finds min comparison  return max 2 min  B)  T(n) = T(floor(/2)) + T(ceil (/2)) + 2  ()  T(2) = 1 T(1) = 0  T(n) = 2T(/2) + 2  T(n) = 3/2 - 2  * uses n steps to get max/min cause it  traverses the array linearly	Compare to determine min & max	
Prew for max in min of left half recor for max in min of left half one Ords max comparison one Linds min comparison return max (min  B)  T(n) = T(floor(1/2)) + T(ceil(1/2)) + 2  ()  T(2) = 1 T(1) = 0  T(n) = 2T(1/2) + 1  T(n) = 31/2 - 2  * uses n steps to get max/min cause it transises the array linearly	Teting back	
recur for max in in of left half recur for righ half  Done This max comparison  one finds min comparison  return max in in  B)  T(n) = T(floor (1/2)) + T(ceil (1/2)) + 2  (1/2) = 1 T(1) = 0  T(n) = 2T(1/2) + 1  T(n) = 31/2 - 2  * uses n steps to get max/min cause it  transfers the array linearly	6/50	
The foreign helf  One finds max comparison  one finds min comparison  return max comparison  (eturn max comparison  (in) = T(floor(1/2)) + T(cell (1/2)) + 2  (in) = T(floor(1/2)) + T(cel	Court Committee Colored	
return max comparison  (eturn max comparison  Th) = T(floor(1/2)) + T(ceil(1/2)) + 2  () T(2) = 1 T(1) = 0  T(n) = 2T(1/2) + 2  T(n) = 31/2 - 2  * uses n steps to get max/min cause it  traverses the array linearly	Celler for Nich hall	
The finds on comparison  (eturn max cynin  B)  T(n) = T(floor(1/2)) + T(ceil(1/2)) + 2  ()  T(2) = 1 T(1) = 0  T(n) = 2 + (1/2) + 2  T(n) = 3 / 2 - 2  * uses n steps to get max/min cause it  transfers the array linearly	DOD OF THE MELT	-
T(n) = T(f(n)r(n/2)) + T(ce(n/2)) + 2 $T(2) = 1 T(n) = 0$ $T(n) = 2T(n/2) + 2$ $T(n) = 3/2 - 2$ $* uses n steps to get max/min cause it traverses the array linearly$	one that hax confanson	
T(n) = T(f(n)r(n/2)) + T(ce(n/2)) + 2 $T(2) = 1 T(n) = 0$ $T(n) = 2T(n/2) + 2$ $T(n) = 3/2 - 2$ $* uses n steps to get max/min cause it traverses the array linearly$	ore Linds him comparison	
T(n) = T(floor(7z)) + T(ceil(7z)) + 2 $C) = T(1) = 0$ $T(n) = 2T(7z) + 2$ $T(n) = 37z - 2$ $* uses n steps to get max/min cause it traverses the array linearly$	TETOIT MAX CANIN	
T(n) = $T(floor(7z)) + T(ceil(7z)) + 2$ C) $T(z) = 1$ $T(1) = 0$ T(n) = 2T(7z) + 2 T(n) = 37z - 2 * uses $n$ steps to get max/min cause it traverses the array linearly		
T(2)=1 T(1)=0  T(n)=2T(1/2)+2  T(n)=31/2-7  * uses n steps to get max/min cause it traverses the array linearly		
T(2)=1T(1)=0  T(n)=2T(1/2)+1  T(n)=31/2-2  * uses n steps to get max/min cause it  traverses the array linearly	1ch = 1(+100((/2))+1(cei(/2))+1	
T(2)=1T(1)=0  T(n)=2T(1/2)+1  T(n)=3/2-7  * uses n steps to get max/min (ause it traverses the array linearly		
T(n)=2T(1/2)+2  T(n)=31/2-2  * uses n steps to get max/min cause it traverses the array linearly		
* uses n steps to get max/min cause it traverses the array linearly	1 (2   2   1 (1) 20	
* uses n steps to get max/min cause it traverses the array linearly		
* uses n steps to get max/min cause it traverses the array linearly	$\frac{1}{(n)} = \frac{(1/2)}{4}$	
* uses n steps to get max/min cause it traverses the array linearly		
	* uses a steps to get max/min louse it	
	traverses the array linearly	

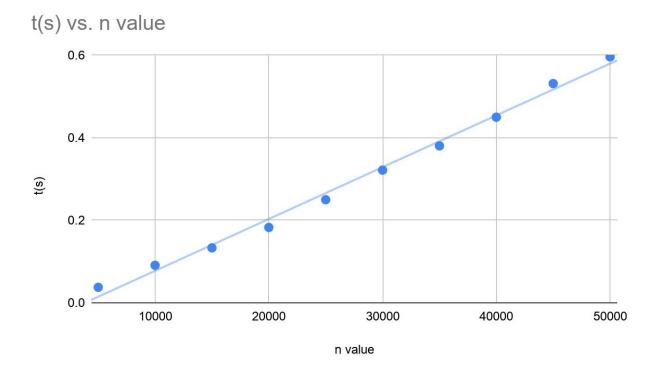
Problem #2 Mergesort3 (ARDIN-1 150 UC1 return A merges (Lo, LI, L3) return merge (LO, morge (11, L3 ê-0 B 1=3T(n3)+0 \* Salution of recurrence in Part B アレア ア ア ア ア ア ア ア ア ア ア

## Problem #4:

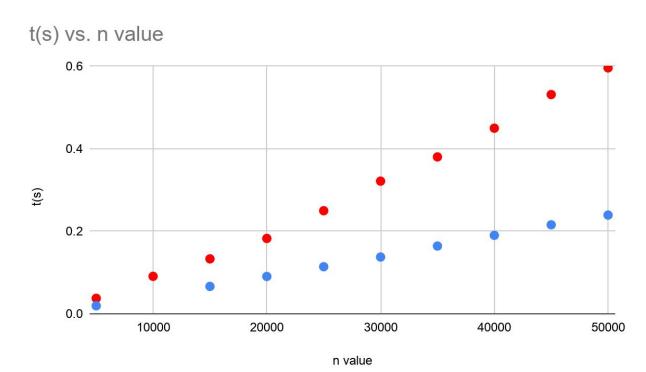
B.

N value	t(s)
5000	0.03757095337
10000	0.09090304375
15000	0.1331989765
20000	0.1827030182
25000	0.2499029636
30000	0.3215858936
35000	0.3803420067
40000	0.4499289989
45000	0.5314919949
50000	0.5961170197

C.



Judging from my data and the graph, The equation it mostly follows would be a polynomial curve just like the mergesort from assignment 1. Possible nlog(n).



This graph shows red being the mergesort3 and blue as the mergesort from assignment 1. When comparing the two I was really surprised because I figured mergesort3 would be faster. I averaged five separate time collections for each sorting technique and it remained the same. When comparing the two I see that they both seem to follow a similar curve. The only thing that I could see making the difference is the base of the nlog(n) for each curve. Mergesort3 has a log base of three compared to mergesorts log base 2.