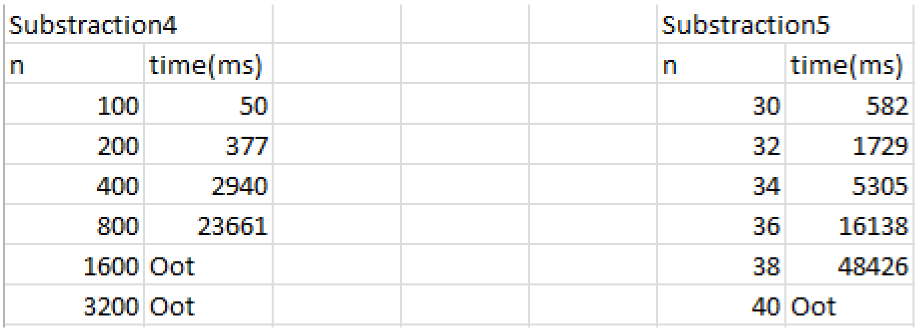
Algorithmics	Student information	Date	Number of session
	UO: 295368		3
	Surname: Álvarez Hernández		
	Name: Miguel		

Activity 1. Divide and conquer by subtraction

In subtraction 1 we have a complexity of O(n) as a == 1 so it is O(n^(k+1)) and k is equal to 0, while in subtraction 2 we have a complexity O(n^3) as a == 1 and k == 2 and we have the same formula as in subtraction 1 as they both are divide and conquer by substraction. They both stop giving times after n = 8192 because of an exception java.lang.StackOverflowError which is caused because the counter get too big so we get stack overflow.

To calculate how many years will take to execute Substraction3 with n = 80 we have to take a look at the complexity that is O(2^n) as a > 1 so we have to apply O(a^(n/b)), and b is 1. Now We use the formula t2 = a^(n2-n1)*t1 where k = n2/n1:

-> t = (2^80)/(2^30) * 39527= 4,45 x 10^19 ms = 1,4 x 10^9 years



Substraction4					Substraction5	
n	time(ms)				n	time(ms)
100	50				30	582
200	377				32	1729
400	2940				34	5305
800	23661				36	16138
1600	Oot				38	48426
3200	Oot				40	Oot

To calculate how many years will take to execute Substraction5 (O(3n/2)) with n = 80 we follow the same procedure as in Subtraction 3:

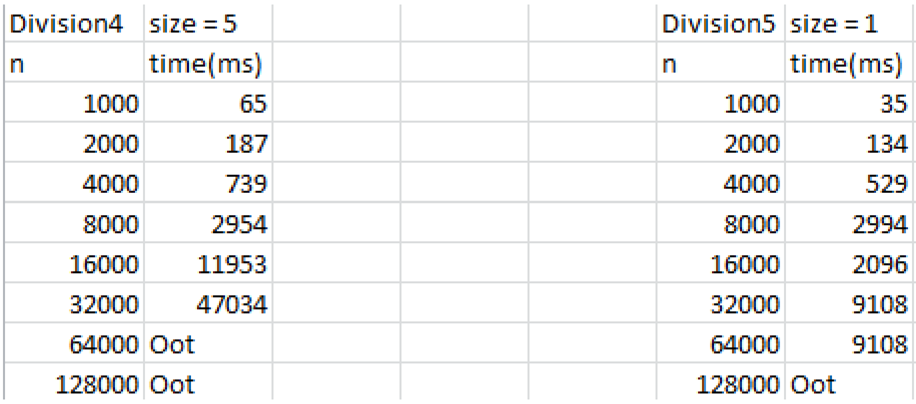
-> t = (3^(80/2))/(2^(30/2)) * 582= 4,93 x 10^14 ms = 1,56 x 10^4 years

Activity 2. Divide and conquer by division

We can see that the fastest algorithm is Division1, which has complexity is O(n).

Division2 has a complexity O(nlogn) this is because in this case a is 2, b is 2 and k is 1 so a=b^k and we use the formula O(n^k logn). Division3 has the same complexity as Division1 but in this case k = 0 so a > b^k so we get the complexity in this way: O(n^(log$_b$a) and also another difference is that Division 1 divides the size problem by 3 and Division3 by 2 so with the same size problems and same repetitions, Division1 will always be faster.

Now I get the times of Division4 and Division4:



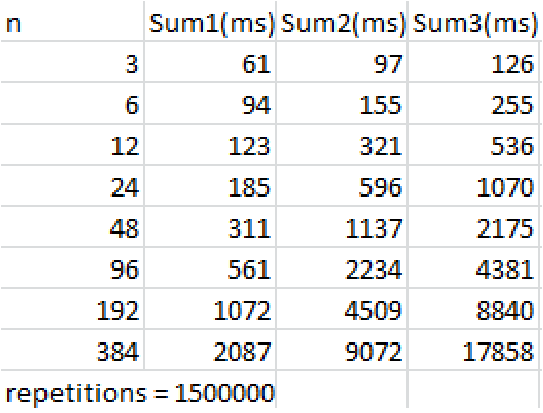
Division4	size = 5				Division5	size = 1
n	time(ms)				n	time(ms)
1000	65				1000	35
2000	187				2000	134
4000	739				4000	529
8000	2954				8000	2994
16000	11953				16000	2096
32000	47034				32000	9108
64000	Oot				64000	9108
128000	Oot				128000	Oot

We can see that if both programs had the same size and despite having the same complexity, Division 4 is faster than Division5. This is probably because Division5 have many more recursive calls than Division4..

Activity 3. Basic Examples

Sum1 has a linear complexity O(n), sum2 has the same complexity as Sum 1 but using a recursive call, and sum3 has also the same complexity as a > b^k and a = 2, b = 2 and k = 0 so O(n^(log$_b$a)) is also O(n)

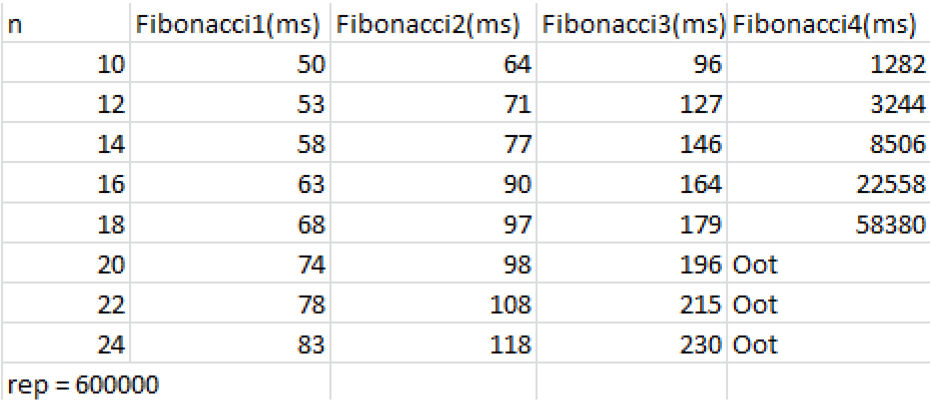
From this point the times are measured with my personal computer



n	Sum1(ms)	Sum2(ms)	Sum3(ms)
3	61	97	126
6	94	155	255
12	123	321	536
24	185	596	1070
48	311	1137	2175
96	561	2234	4381
192	1072	4509	8840
384	2087	9072	17858
repetitions = 1500000			

Sum1 is clearly the best as it does not use recursion

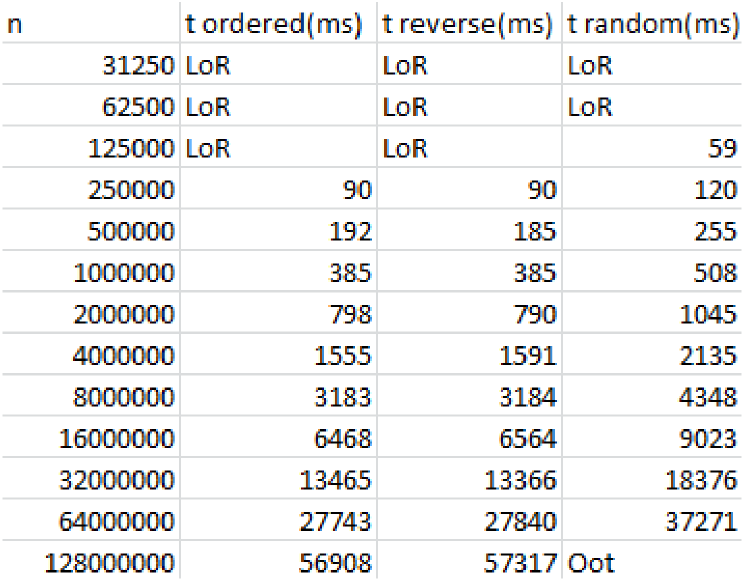
With the Fibonacci problems happens practically the same, Fibonacci1 and 2 do not use recursion and have both O(n) complexity, Fibonacci3 has a recursive call so O(n) and in Fibonacci 4 we get the complexity knowing that a = 2, b = 1 or 2 and k = 0 in both cases of b a > b^k and T(n)=T(n-1)+T(n-2)+O(1), that is an exponential solution O(1.6^n)



n	Fibonacci1(ms)	Fibonacci2(ms)	Fibonacci3(ms)	Fibonacci4(ms)
10	50	64	96	1282
12	53	71	127	3244
14	58	77	146	8506
16	63	90	164	22558
18	68	97	179	58380
20	74	98	196	Oot
22	78	108	215	Oot
24	83	118	230	Oot
rep = 600000				

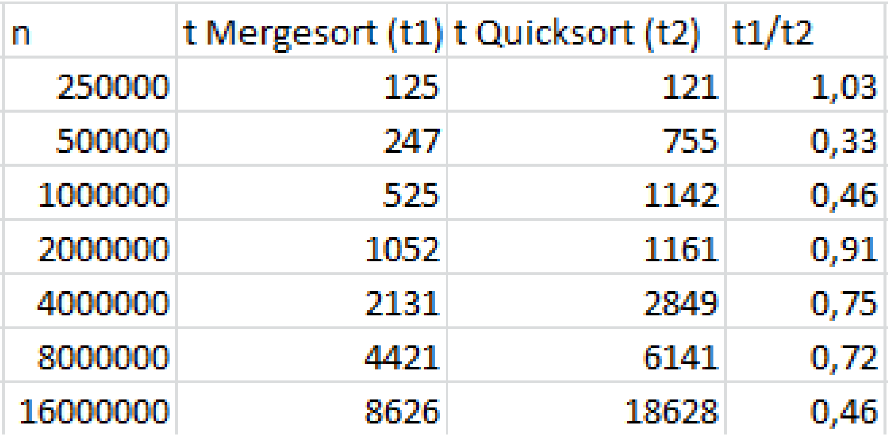
Clearly, Fibonacci1 and Fibonacci2 are the fastest ones. Fibonacci4 has the worst times as it has the worst complexity O(1.6^n) .

Activity 4. Mergesort



n	t ordered(ms)	t reverse(ms)	t random(ms)
31250	LoR	LoR	LoR
62500	LoR	LoR	LoR
125000	LoR	LoR	59
250000	90	90	120
500000	192	185	255
1000000	385	385	508
2000000	798	790	1045
4000000	1555	1591	2135
8000000	3183	3184	4348
16000000	6468	6564	9023
32000000	13465	13366	18376
64000000	27743	27840	37271
128000000	56908	57317	Oot

	Student information	Date	Number of session
Algorithmics	UO: 295368		3
	Surname: Álvarez Hernández		
	Name: Miguel		



n	t Mergesort (t1)	t Quicksort (t2)	t1/t2
250000	125	121	1,03
500000	247	755	0,33
1000000	525	1142	0,46
2000000	1052	1161	0,91
4000000	2131	2849	0,75
8000000	4421	6141	0,72
16000000	8626	18628	0,46

With small problem size, quicksort seems to be a little bit faster, but as the problem size increases, Mergesort starts getting faster times than Quicksort.