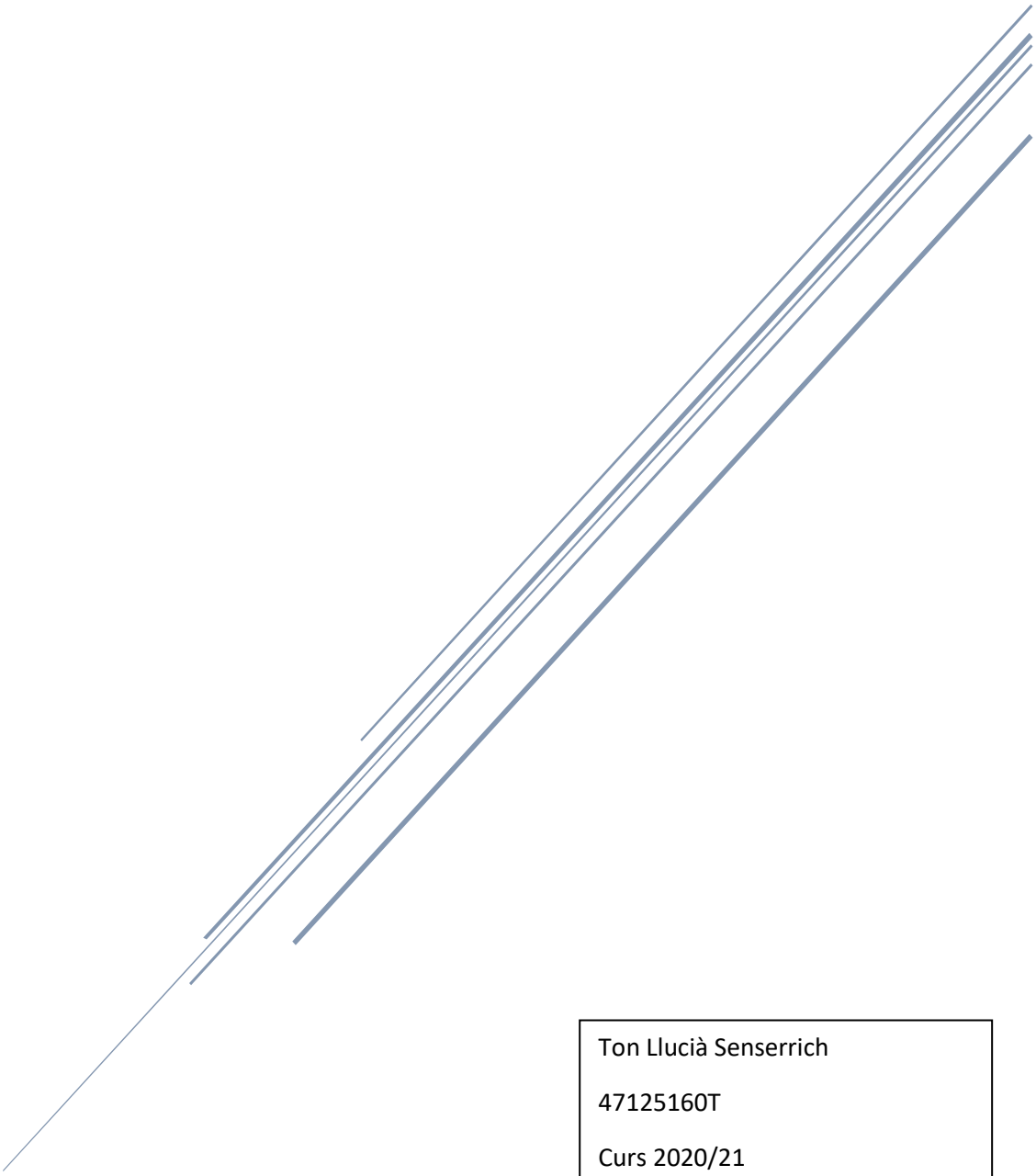
PRACRICA 1

Estructura de dades



Ton Llucià Senserrich

47125160T

Curs 2020/21

GEI UdL EPS

1 BubbleSort

Aquest algoritme consta de dues instruccions for, la primera el que farà serà separar la part ordenada de la no ordenada de l'array, i l'altre ens servirà per anar recorrent l'array i ordenant.

El funcionament consisteix a agafar 2 nombres consecutius de l'array, començant pel d'índex 0 i 1, i els anomenarem A i B respectivament, compararem si A es més gran que B i si es així els intercanviarem i en cas contrari A passarà a tenir el valor de B i B el del següent nombre i així fins al final delimitat pel primer for, a cada volta la part del array no ordenada disminueix la seva dimensió una posició fins a aconseguir l'array completament ordenada.

Aquest algoritme té un cost de O(n^2), ja que disposa de dos bucles for aniuats.

2 SelectionSort

Aquest algorisme també consta de dues instruccions fork que ens serviran per poder cada element amb els altres elements de l'array.

El funcionament consisteix a agafar el primer element de l'array i comparar-lo amb el següent, i si el 2n és més gran que el primer els intercanviem, i repetim això en tot l'array, però excloent els primers de l'array, ja que ja estaran ordenats

Aquest algoritme té un cost de O(n^2), ja que disposa de dos bucles for aniuats.

3 QuickSort

Aquest algorisme està implementat mitjançant la recursivitat en la funció quickSort, però tal com s'indica en l'enunciat s'ha implementat la funció auxiliar partition de forma iterativa, per tant, s'explicarà en 2 parts, la funció quicksort i la funció partition

1-la funció quickSort, la seva funció és ordenar l'array, per tant el que fa es trobar la posició i el valor de pivot mitjançant la funció auxiliar choosePivotPosition, llavors aparta el pivot per tal de no tenir-lo en compte i després crida a la funció partition i un cop obtingut el nou pivot retornat per la funció, es posa un altre cop el pivot que hem apartat en el seu lloc i es crida un altre cop a ella mateixa 2 cops per tal d'ordenar la part de la dreta i la de l'esquerra

2-la funció partition el que fa és recórrer l'array des de l'índex d'inici l, per tal de trobar algun nombre més gran que pivot, i l'índex de final r, per tal de trobar un nombre més petit que pivot, i si els troba els intercanvia i finalment retorna el nou pivot.

Aquest algorisme té una complexitat total de O(n^2), ja que a part de tenir un while te les crides recursives de la funció quickSort, tot i que si només volguéssim tenir en compte la part iterativa continguda en la funció partition tindria un cost de O(n).

4 -Benchmarks

Benchmark	(N)	Mode	Cnt	Score	Error	Units
MyBenchmark.testBubbleSort	100	avgt	5	0,014	± 0,001	ms/op
MyBenchmark.testBubbleSort	200	avgt	5	0,051	± 0,006	ms/op
MyBenchmark.testBubbleSort	400	avgt	5	0,164	± 0,024	ms/op
MyBenchmark.testBubbleSort	800	avgt	5	0,520	± 0,062	ms/op
MyBenchmark.testInsertionSort	100	avgt	5	0,005	± 0,001	ms/op
MyBenchmark.testInsertionSort	200	avgt	5	0,016	± 0,001	ms/op
MyBenchmark.testInsertionSort	400	avgt	5	0,065	± 0,014	ms/op
MyBenchmark.testInsertionSort	800	avgt	5	0,238	± 0,005	ms/op
MyBenchmark.testQuickSort	100	avgt	5	0,004	± 0,001	ms/op
MyBenchmark.testQuickSort	200	avgt	5	0,009	± 0,001	ms/op
MyBenchmark.testQuickSort	400	avgt	5	0,019	± 0,001	ms/op
MyBenchmark.testQuickSort	800	avgt	5	0,055	± 0,029	ms/op
MyBenchmark.testSelectionSort	100	avgt	5	0,017	± 0,002	ms/op
MyBenchmark.testSelectionSort	200	avgt	5	0,058	± 0,006	ms/op
MyBenchmark.testSelectionSort	400	avgt	5	0,200	± 0,027	ms/op
MyBenchmark.testSelectionSort	800	avgt	5	0,643	± 0,018	ms/op

En executar els benchmarks observem que els quick sort és el més ràpid amb diferència, I si ho volguéssim ordenar, podríem concloure que per ordre de velocitat seria: quickSort, insertSort, bubbleSort, selectionSort.

5 Conclusions

El que mes m'ha costat d'aquesta practica ha estat el desenvolupament de l'algorisme quickSort ja que en un principi no vaig adonar-me que unicament la funció partition havia de ser iterativa i per tant jo el vaig intentar desenvolupar completamente iteratiu, en adonarmen i fer una mica de recerca vaig aconseguir un algorisme que crec que es bastant eficiente.