TP4 – Quicksort & Mergesort

1)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| QUICKSORT-CUTOFF | | | | | | | | | | | | | | | |
|  |  |  | TIME | | | | | | | | | |  |  |  |
| N | M | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 17 | 20 | 23 | 25 |
| 1000 | 34 | 34 | 34 | 35 | 36 | 35 | 34 | 37 | 34 | 33 | 34 | 34 | 33 | 35 | 34 |
| 10000 | 1016 | 1042 | 967 | 1017 | 959 | 1010 | 982 | 1024 | 1000 | 1001 | 1002 | 998 | 982 | 981 | 1018 |
| 100000 | 97531 | 103158 | 126362 | 96473 | 101013 | 109671 | 109609 | 104772 | 110033 | 99821 | 121425 | 103189 | 113063 | 107469 | 113054 |
| 1000000 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

El Quicksort con cutoff funciona de la siguiente manera. Ordena como un quicksort, es decir, toma un elemento de la lista a ordenar como pivote, en este caso, el ultimo elemento de la lista, y va comparando los demás elementos con ese pivote. Manda de un lado del pivote los que son mayores a él y del otro lado los que son menores, hasta el momento en que el pivote es menor o igual al primer elemento de la lista. Sin embargo, el cutoff se diferencia porque corta antes el ordenamiento por quicksort, y a partir de ese momento ordena lo que queda de la lista utilizando un insertion sorter. Cuanto antes corta con el ordenamiento por quicksort esta dado por el M que utiliza.

Para realizar estas mediciones se hicieron cinco pruebas para cada M distinto, se sumaron los resultados de tiempo que daban cada una de las pruebas, y luego se realizó un promedio. Estan marcados en amarillo los menores tiempos.

En este grafico se puede ver como hasta M = 17 el tiempo se mantiene mas o menos constante, para luego crecer bruscamente a partir de ese valor de M.

|  |  |  |  |
| --- | --- | --- | --- |
| Quicksort-not recursive | | | |
| N | time | swaps | greater |
| 1000 | 35 | 1455 | 53574 |
| 10000 | 954 | 14687 | 4037293 |
| 100000 | 106706 | 146643 | 438229894 |
| 1000000 |  |  |  |

2)

|  |  |  |  |
| --- | --- | --- | --- |
| Quicksort | | | |
| N | time | swaps | greater |
| 1000 | 36 | 1468 | 52458 |
| 10000 | 975 | 14633 | 4342024 |
| 100000 | 107059 | 145835 | 433920759 |
| 1000000 |  |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Quicksort-CutOff | | | | | | | |
| N | time | swaps(quicksort) | swaps(insertion) | swaps(sum) | greater(quicksort) | greater(insertion) | greater(sum) |
| 1000 | 32 | 1166 | 157 | 1323 | 48068 | 1156 | 49224 |
| 10000 | 964 | 12800 | 216 | 13016 | 4148552 | 10215 | 4158767 |
| 100000 | 116524 | 137751 | 186 | 137937 | 469434769 | 100185 | 469534954 |
| 1000000 |  |  |  |  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Quicksort- three way partitioning | | | |
| N | time | swaps | greater |
| 1000 | 19 | 1969 | 0 |
| 10000 | 31 | 25858 | 0 |
| 100000 | 115 | 250184 | 0 |
| 1000000 |  |  |  |

En el grafico se puede ver la comparacion del tiempo que tardan en ordenar los diferentes tipos de quicksort.

Podemos observar que el metodo por three way partitioning es muchisimas veces mas rapido que los otros metodos. No se observa en el grafico por la magnitud de la escala, pero si comparamos el tiempo que tarda en ordenar 100000 numeros, por Quicksort normal tarda 107059 milisegundos, mientras que por three way partitioning tarda en promedio 151 milisegundos, es decir 700 veces menos, aproximadamente.

3) Mergesort- BottomUp-TopDown

|  |  |  |  |
| --- | --- | --- | --- |
| MergeSort - BottomUp | | | |
| N | time | greater | swaps |
| 100 | 10 | 688 | 0 |
| 500 | 25 | 4492 | 0 |
| 1000 | 30 | 9984 | 0 |
| 2500 | 41 | 29080 | 0 |
| 5000 | 57 | 63160 | 0 |
| MergeSort - TopDown | | | |
| N | time | greater | swaps |
| 100 | 13 | 672 | 0 |
| 500 | 26 | 4488 | 0 |
| 1000 | 39 | 9976 | 0 |
| 2500 | 43 | 28404 | 0 |
| 5000 | 54 | 61808 | 0 |

En el grafico se puede observar que el método BottomUp es mas rápido, pero esta tendencia va cambiando a medida que N va creciendo. Por otro lado, vemos que el método Top Down es mas lento en un principio, para luego, a medida que N crece, volverse mas rápido que el método BottomUp para N mucho mas grande.

En el grafico se puede observar que la cantidad de comparaciones que realizan los dos métodos es aproximadamente la misma.