Messungen Team03:

Aufbau: siehe Skizze

Resultate: Zahlen.dat (von der Hompage von Prof. Dr. Klauck) – 20003 Elemente

Die Laufzeit wurde in Nanosekunden (ns) gemessen und in Millisekunden (ms) umgerechnet:

*Resultat 1:*

* Zahlen.dat (von der Hompage von Prof. Dr. Klauck) – 20003 Elemente

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Quicksort OHNE Insortionsort** | **Links** | **Rechts** | **Random** | **Median** |
| Team 03 | 5352 | 5297 | 5281 | 5491 |
|  | 4940 | 4864 | 4950 | 4930 |
|  | 4933 | 4872 | 4947 | 5113 |
|  | 4957 | 4882 | 4880 | 4945 |
|  | 4967 | 4883 | 4919 | 4943 |
| Durchschnitt | **5029,8** | **4959,6** | **4995,4** | **5084,4** |
|  |  |  |  |  |
| **Quicksort mit Insortionsort (ab < 12 Elemente)** | **Links** | **Rechts** | **Random** | **Median** |
|  | 6302 | 6209 | 6305 | 6352 |
|  | 4932 | 4864 | 4892 | 4943 |
|  | 6308 | 6205 | 6311 | 6318 |
|  | 4971 | 4879 | 4972 | 4977 |
|  | 6292 | 6221 | 6152 | 6314 |
| Durchschnitt | **5761** | **5675,6** | **5726,4** | **5780,8** |
|  |  |  |  |  |
| **Quicksort mit Insortionsort (ab 50 Elemente)** | **Links** | **Rechts** | **Random** | **Median** |
|  | 6381 | 6357 | 6355 | 6355 |
|  | 6314 | 6237 | 6285 | 6402 |
|  | 6301 | 6240 | 6366 | 6335 |
|  | 4944 | 4863 | 4955 | 5007 |
|  | 4960 | 4892 | 5024 | 4998 |
| Durchschnitt | **5780** | **5717,8** | **5797** | **5819,4** |
|  |  |  |  |  |
| **Quicksort mit Insortionsort (ab 100 Elemente)** | **Links** | **Rechts** | **Random** | **Median** |
|  | 14963 | 14977 | 14969 | 15044 |
|  | 14999 | 14973 | 14976 | 15066 |
|  | 15317 | 15279 | 15459 | 15360 |
|  | 16311 | 16258 | 16315 | 16347 |
|  | 14933 | 14900 | 14968 | 14994 |
| Durchschnitt | **15304,6** | **15277,4** | **15337,4** | **15362,2** |

*Auswertung 1:*

Man sieht bei den Messungen, dass der Quicksort Algorithmus OHNE Insertionsort bei unserer Implementierung am schnellsten ist.

Wird Insertionsort mitverwendet, steigt die Laufzeit.

*Resultat 2:*

* Worse-case Szenario mit 10.000 Elementen

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Links | Rechts | Random | median |
| Team 09 | 137137 | 136707 | 1122 | 136918 |
| Team 03 | 931 | 536 | 569 | 865 |

*Auswertung 2:*

Es wurde einmal das ADTArray des Teams 09 und unser ADTArray mit unserer Implementation des Quicksort-Algorithmus verwendet.

Es zeigt sich, dass es zu einer massiven Laufzeit Erhöhung kam, als wir das ADTArray mit einem fremden ADTArray austauschten.

*Resultat 3:*

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Quicksort | **500** | **1000** | **2000** | **4000** | **8000** | **16000** | **32000** | **64000** | **128000** |
| **Laufzeit** | 61 | 100 | 216 | 489 | 1230 | 3802 | 13708 | 50866 | 199286 |
| **Lesezugriffe** | 10080 | 33516 | 82372 | 188914 | 432059 | 990200 | 2369269 | 6209014 | 17910310 |
| **Schreibzugriffe** | 1679870 | 1685064 | 1696414 | 1720498 | 1770888 | 1873606 | 2084838 | 2506220 | 3354848 |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Insertion-Sort | **100** | **500** | **700** | **1100** | **1300** | **1700** | **1900** | **2300** | **3000** | **3500** |
| **Laufzeit (ms)** | 84 | 954 | 2046 | 5028 | 8140 | 14213 | 18970 | 29631 | 56589 | 81443 |
| **Lesezugriffe** | 10098 | 250498 | 490698 | 1211098 | 1691298 | 2891698 | 3611898 | 5292298 | 9002998 | 12253498 |
| **Schreibzugriffe** | 5049 | 125249 | 245349 | 605549 | 845649 | 1445849 | 1805949 | 2646149 | 4501499 | 6126749 |

*Auswertung 3:*

Es zeigt sich beim Insortionsort Algorithmus, dass es zu einer exponentiellen Steigung komm, sodass nach 3500 Elementen die Messung beendet wurde.

Im Vergleich dazu steigt der Quicksort Algorithmus deutlich langsamer an.

Bei beiden Algorithmen wurde der worse-case Fall betrachtet.