|  |  |  |  |
| --- | --- | --- | --- |
|  | ListPQ | DictionaryPQ | HeapPQ |
| Dequeue | O(nlogn)\* | O(logn) | O(logn) |
| Peek | O(nlogn)\* | O(logn) | O(1) |
| Enqueue | O(1) | O(logn) | O(logn) |
| GetEnumerator | O(n) | O(nlogn) | O(n) |
| Remove(T,I) | O(n) | O(logn) | O(n) |
| Remoev(T) | O(n) | O(n) | O(n) |

\*nach erstmaligem Aufruf ohne weiteres Einfügen O(1)

Experimentelle Resultate:

Enqueue von 1.000.000 Elementen: DictQueue braucht ewig (2.5 Sekunden), ListQueue und HeapQueue sind ca gleich schnell (~0.3 Sekunden)

Sequentielles Dequeue von 1.000.000 Elementen: DictQueue und HeapQueue brauchen lang (1-3 Sekunden), ListQueue geht schneller (~0.7 Sekunden)

Enqueue und Dequeue vermischt: ListQueue ist komplett unbrauchbar (TIMEOUT). DictQueue braucht lange (1.6s), HeapQueue ist am schnellsten! (0.4s)

Entfernen von vielen Elementen nach T braucht bei allen super lang (TIMEOUT). Beim Löschen von 1000 Elementen, braucht die DictQueue am längsten (TIMEOUT), die ListQueue 5 Sekunden und die HeapQueue 6 Sekunden. Also definitiv keine gute Operation! Sollte man vermeiden!

|  |  |  |  |
| --- | --- | --- | --- |
|  | ListPQ | DictPQ | HeapPQ |
| Enqueue | OK (0.3s) | LANG (2.5s) | OK (0.3s) |
| Dequeue (Seq.) | OK (0.7s) | LANG (1-3s) | LANG (1-3s) |
| Enq / Deq gemischt | TIMEOUT | LANG (1.6s) | OK (0.4s) |
| RemoveByT (1000) | LANG (5s) | TIMEOUT | LANG(6s) |