B06505004 莊博翰 HW2

****

* Explain briefly how you implemented the randomized queue and deque.
* Which data structure did you choose (array, linked list, etc.)
* and why?

對於 Deque 我採用了 linklist 的資料結構，為了確保 constant time 的 insert 和 pop 同時要兼顧資料的順序，implement 上我令一個 node 是 root 不存資料前後分別接到資料的頭跟尾。這樣實做較為簡易

而 randomized queue 我使用 array ，因為不需考慮順序，pop 時隨機取一個並把最後一個補入空位即可，不會有需要依序 pop 第一個的問題。
這樣使用的 memory 會比 linklist 小(linklist 每個 node 要存比較多東西)。至於 array 容量不足時就會將容量乘 2 如同 dynamic array，remove 到最大容量的 1/4 時會將容量減半。

****

* How much memory (in bytes) do your data types use to store n items
* in the worst case? Use the 64-bit memory cost model from Section
* 1.4 of the textbook and use tilde notation to simplify your answer.
* Briefly justify your answers and show your work.
*
* Do not include the memory for the items themselves (as this
* memory is allocated by the client and depends on the item type)
* or for any iterators, but do include the memory for the references
* to the items (in the underlying array or linked list).

****

Randomized Queue: ~ __20n+20___ bytes
計算
Randomized Queue 的本體是 array of object(reference)
size=容量*(reference size) = n*8 bytes
然後加上 object overhead 16bytes+4bytes(int length)
然而未必會用滿。
worst case:
4(要到 1/4 最大容量才會縮小) * 8bytes(reference)*n+20bytes=32n+20 bytes
best case:
1(用滿)* 8n +20= 8n+ 20 bytes
average : 20n+20 bytes

Deque: ~ _48n_+68___ bytes

計算: 每個 node 有 3 個 reference: next ,data ,previous : 3*8bytes

+16bytes (overhead)

+8 byte(inner class overhead) node 是 Deque 的 inner class

=48 bytes

總共有 n+1 個 node(包含 root) : 48*n +48 +

16(deque object overhead)+4(int size)bytes

=48n_+68

****

* Known bugs / limitations.

總是無可避免的有機會遇到 Randomized Queue 的 worst case