|  |  |
| --- | --- |
| *TAD Stack*  pop | |
| *Grafico*  L  LL  peek  push  4  1  2  3  5 | |
| *Invariantes:*   * *Solo se puede agregar al final de la pila* * *Solo se puede eliminar el ultimo element de la pila* | |
| *Operaciones* | *Push L x Book -> I* |
| *Pop L -> Book* |
| *Peek L -> Book* |
| *Size L -> L* |

|  |  |  |
| --- | --- | --- |
| *TAD Hash Table* | | |
| |  | | --- | |  | | 1 | | 5 | | K1 | | 5 | |  | |  |   *grafico*  7  6  5  1  2  3  4  E1  K1  K3  K2  L  LL  E3  E2 | | |
| *Invariantes:*   * *Complejidad temporal de operaciones de consulta, insercion y borrado debe ser O(1)* | | |
| *Operaciones* | *Insert KxE -> L* | *Function k -> i* |
|  | *Search K -> E* | *codeHash i -> k* |
|  | *Delete K –> L* |  |
|  | *reHashing L -> L* |  |

|  |  |
| --- | --- |
| *TAD Queue* | |
| L  poll  offer  4  3  1  2 | |
| *Invariantes:*   * *Primer elemento en entrar pimero en salir* * *Solo ingresan elementos con la ultima posicion* | |
| *Operaciones* | *Offer L x Book -> L* |
| *poll L -> Book* |
| *Peek L -> Book* |
| *Empty L -> L* |
| *Size L -> L* |

|  |  |
| --- | --- |
| *TAD Priority Queue (Min Heap)* | |
| *Resultado de imagen para arbol binario completo*  L | |
| *Invariantes:*   * *El element padre debe ser menor que sus hijos* * *El arbol debe ser completo* | |
| *Operaciones* | *Insert L x Book -> L* |
| *Delete L x Book -> L* |
| *MinElement L -> Book* |
| *Empty L -> L* |
| *Size L -> L* |