|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Вектор | Списки | Стек | Очередь | Бинарное дерево | Красно-чёрное дерево | Хэш-таблицы |
| Доступ | О(1) | O(n) | *top*(0-ой элемент)  LinkedList: O(1)  Vector: O(1) | *front*(0-ой элемент)  LinkedList: O(1)  DoubleLinkedList: O(1)  Vector: O(1) |  |  |  |
| Вставка в начало | O(n) | О(1) | *push*  LinkedList: O(1)  Vector: amort(О(1)) [при реализации на векторе добавляем в конец вектора] |  |  |  |  |
| Вставка в рандомное место | O(n) | О(1) [Если **знаем, куда** вставлять] |  |  | O(log n) | O(log n) | Amort(O(1)) - цепочки  O(n) - пробирование [худшие случаи] |
| Вставка в конец | amort(О(1)) | O(n) для 1-св  О(1) для 2-св |  | *enqueue*  LinkedList: O(n)  DoubleLinkedList: O(1)  Vector: amort(О(1)) |  |  |  |
| Удаление первого элемента | O(n) или amort(О(1)) (зависит от реализации) | О(1) | *pop*  LinkedList: O(1) Vector: amort(О(1)) [в векторе реализовывается как удаление последнего] | *dequeue*  LinkedList: O(1)  DoubleLinkedList: O(1)  Vector: amort(О(1))/ O(n) |  |  |  |
| Удаление рандомного элемента | O(n) | О(1) [Если **знаем, что** удаляем] |  |  | O(log n) | O(log n) | O(n) [худший случай] |
| Удаление последнего | amort(О(1)) | О(1) |  |  |  |  |  |
| Поиск элемента | O(n) | O(n) |  |  | O(log n) | O(log n) | O(n) [худший случай] |