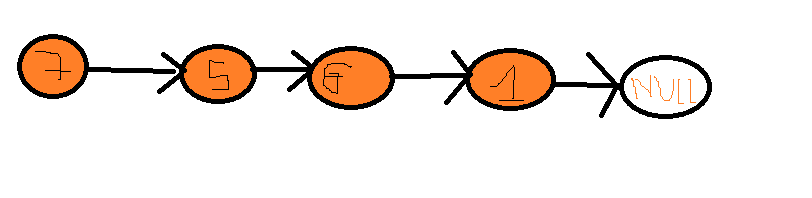
LINKED LIST

Là những cái node được liên kết với nhau

Đặc điểm của node là sẽ trong mỗi node sẽ luôn có 1 cái data và 1 cái pointer của cái node mà nó trỏ tới.

Sẽ có cái đặc điểm là cái node cuối cùng sẽ trỏ tới null



**Khi nào thì dùng linked list**

**Linked list** thường được sử dụng để tạo ra những loai ctdl mà ở trong đó nó sẽ yêu cầu có độ insert và delete nhất là nhiều vd như là : 1 cái list, 1 cái stack, 1 cái queue (bản chất của stack và queue sẽ thêm vào và bớt đi liên tục).

**Được lưu trữ như thế nào ?**

**Ưu điểm của linked list**

* **Thêm vào xóa rất đơn giản chỉ cần thay đổi mối liên kết giữa các node với nhau**
* **Không cần biết trước số lượng phần tử**
* **Các phần tử không cần nằm liên tiếp nhau trong bộ nhớ máy tính**

**Nhược điểm của linked list**

* **Thông thể truy xuất đến giá trị của node thông qua index được mà phải duyệt tuần tự**
* **Tốn bộ nhớ => vì phải chứa con trỏ => liên kết giữa các node với nhau**

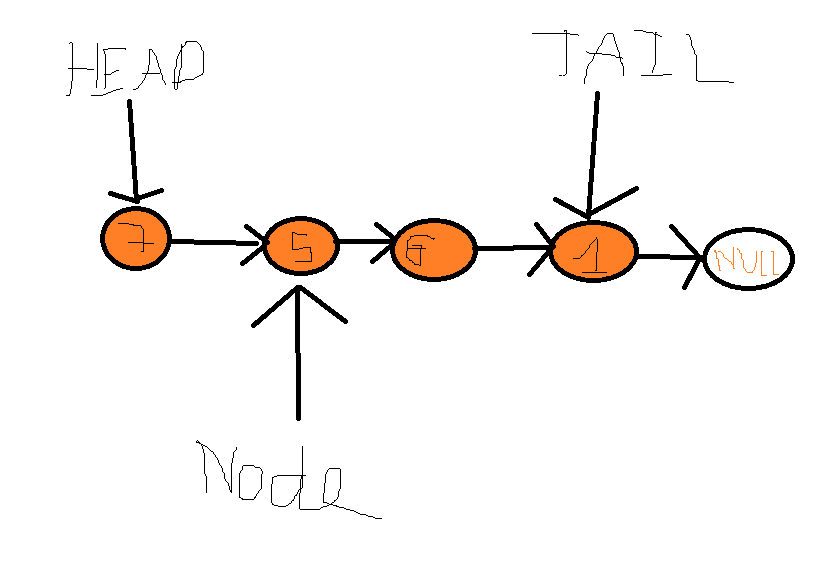
**Ưu điểm của mảng 1 chiều**

* **Truy xuất đến giá trị của mảng thông qua index**
* **Ít tốn bộ nhớ**

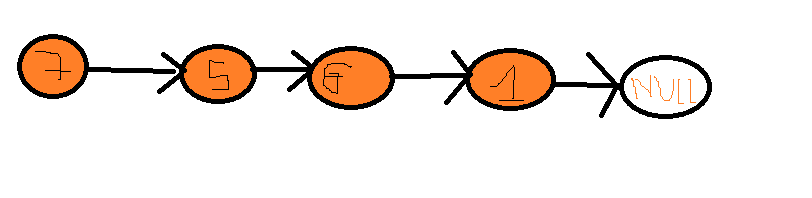
**Nhược điểm của mảng 1 chiều**

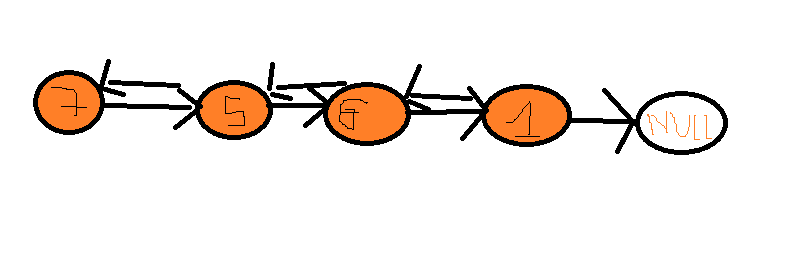
* **Việc thêm, xóa bản chất là ko phải xóa thật mà là thay đổi các vị trị phần tử cho nhau,còn đc gọi là thay đổi index với nhau**
* **Phải biết trước số lượng phần từ của mảng**
* **Các phần tử phải nằm liên tiếp nhau trong bộ nhớ mấy tính**

**Thành phần linked list**

****

**So sánh Singly và Doubly**

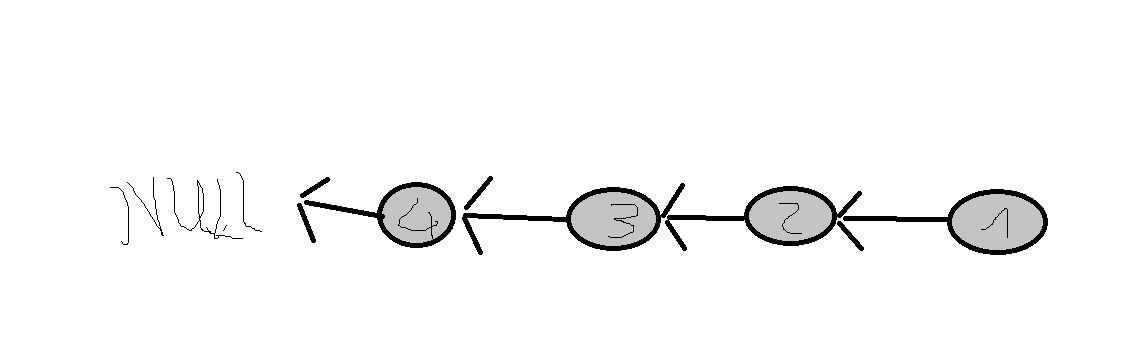


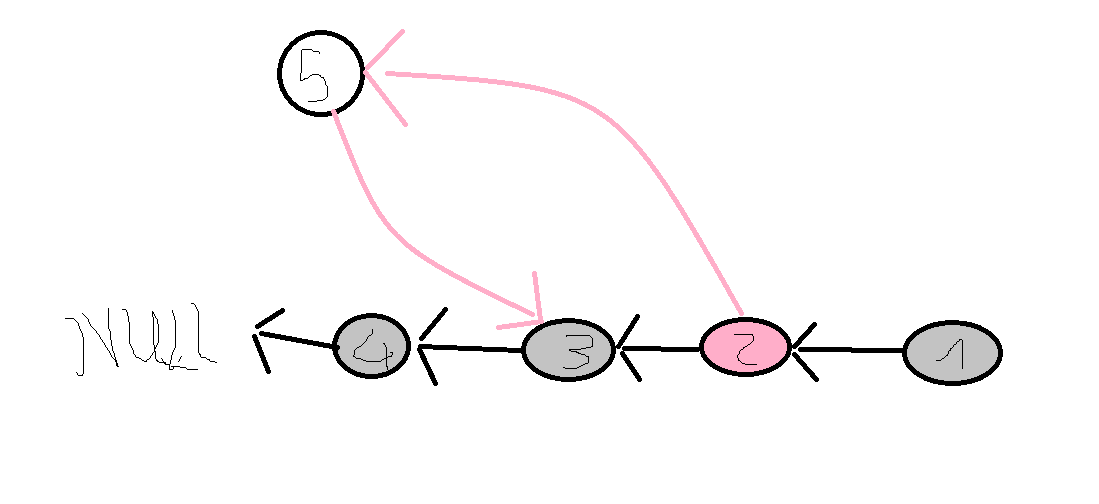
****

**Lợi và Hại**

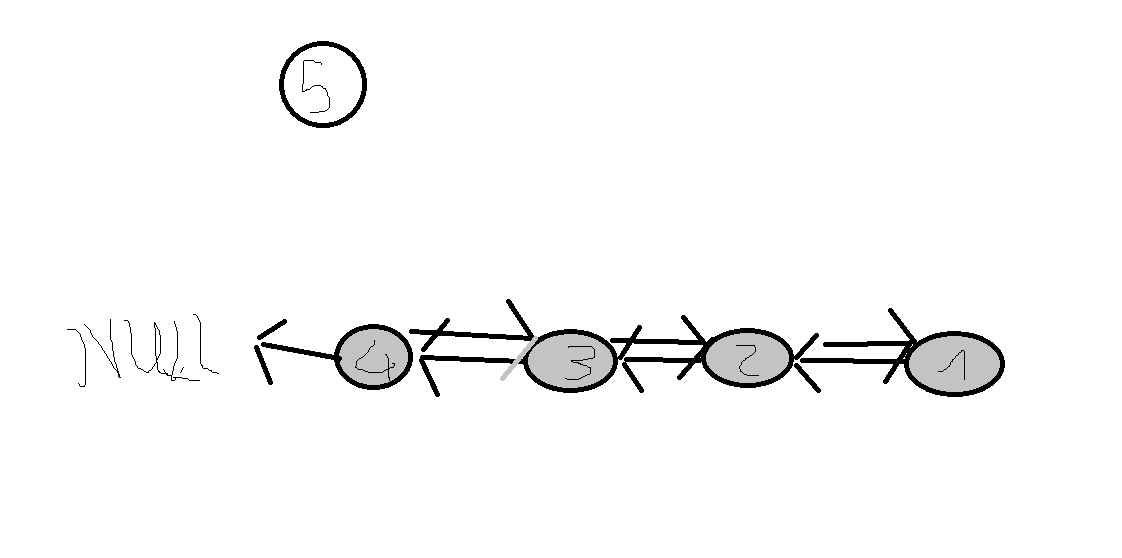
|  |  |  |
| --- | --- | --- |
|  | **Hại** | **Lợi** |
| **Singly** | **Không truy vấn ngược** | **Ít tốn bộ nhớ**  **Code dễ hơn** |
| **Doubly** | **X2 bộ nhớ** | **Truy vấn ngược** |

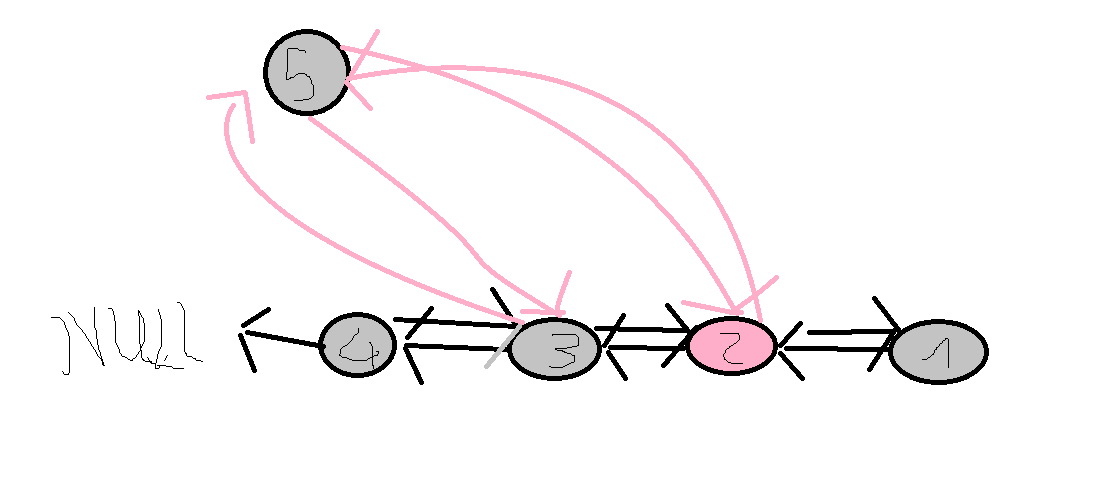
**Insert singly**

****

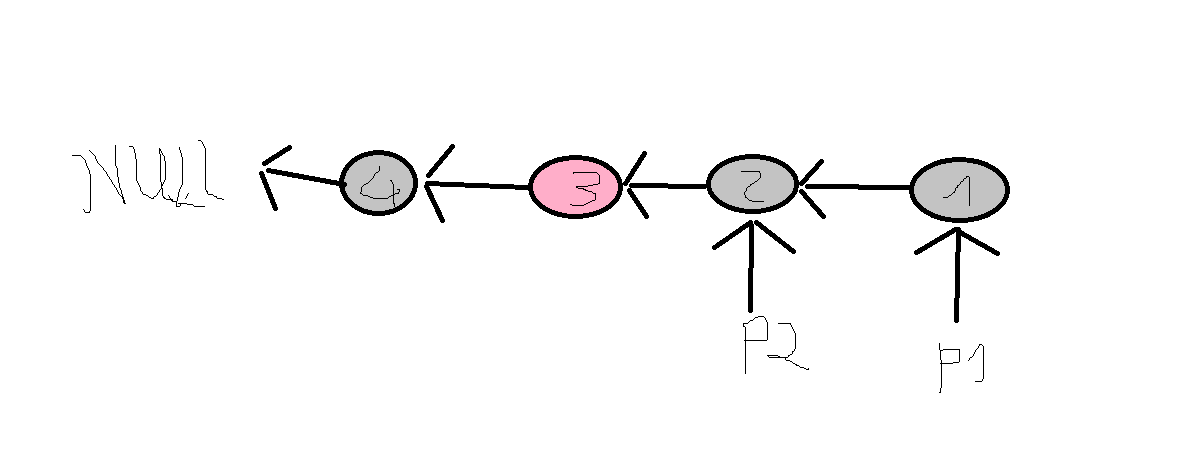
****

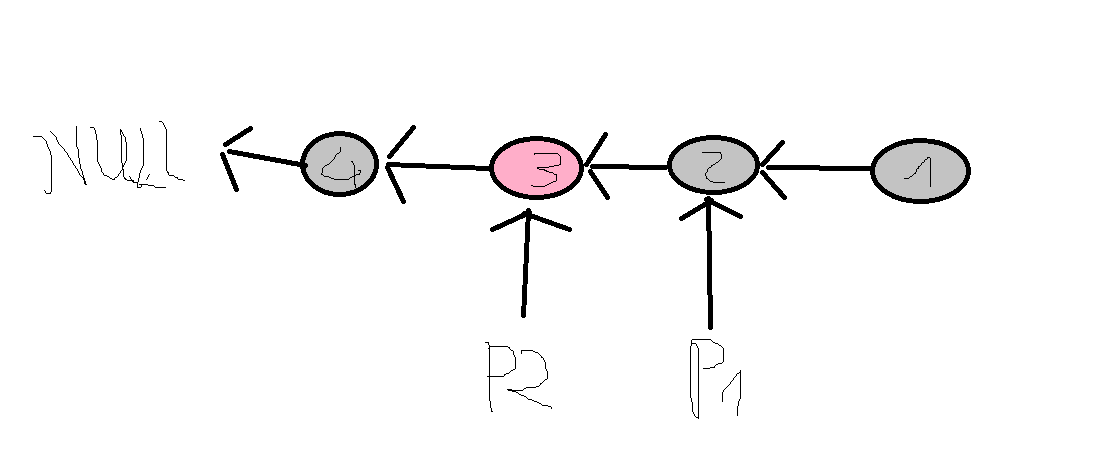
**Insert Doubly**

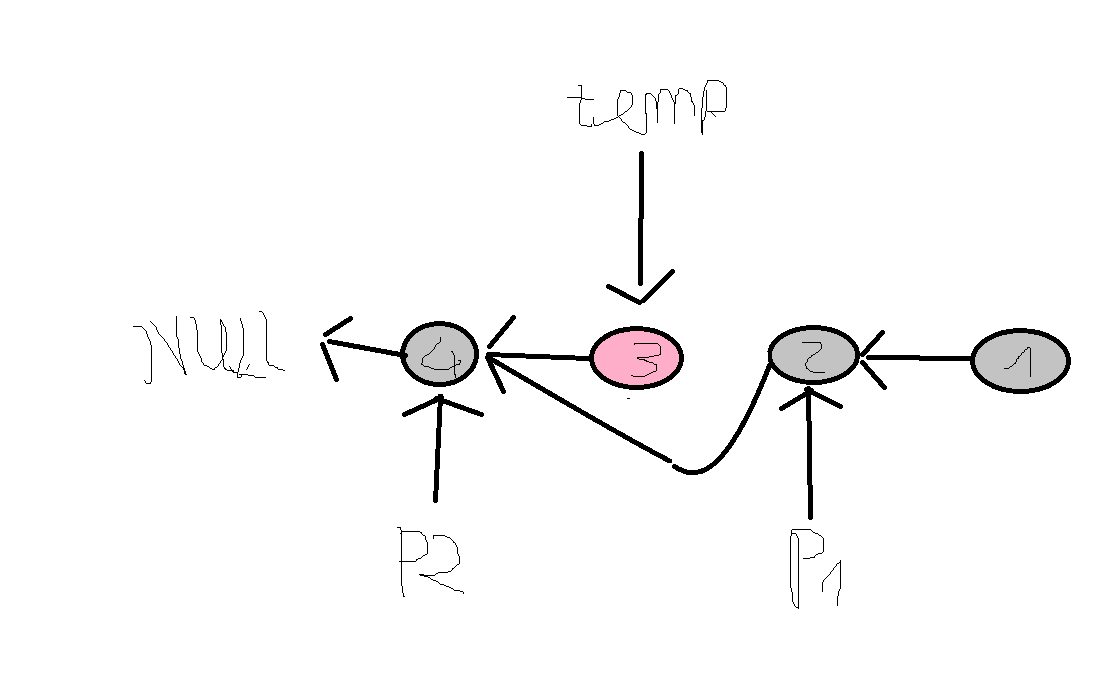
****

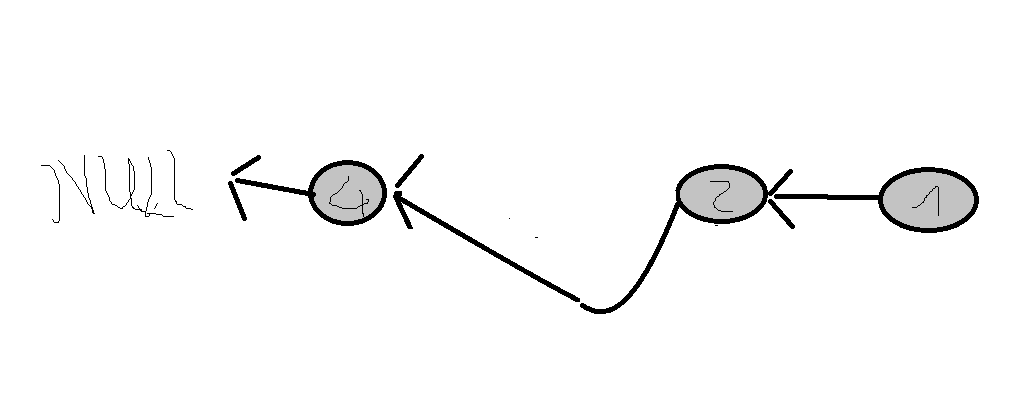
****

**Remove singly**

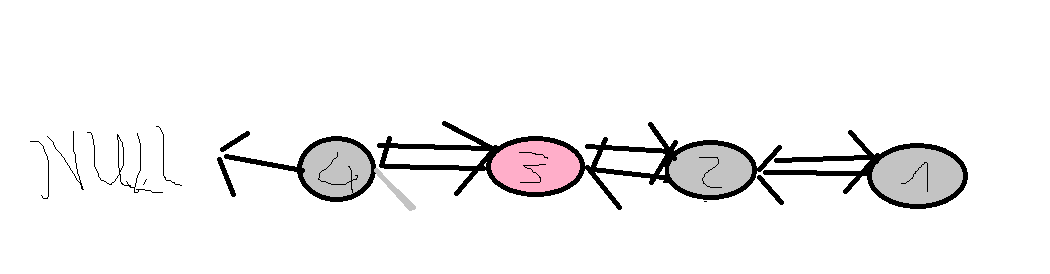
****

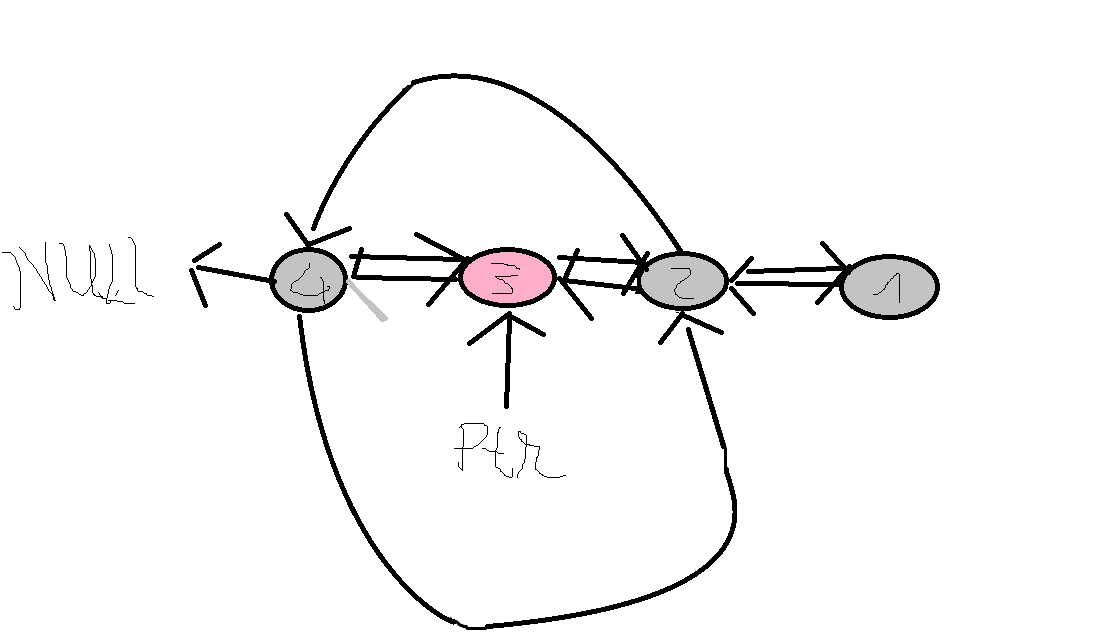
****

****

****

**Remove Doubly**

****

****

**Độ phức tập của thuật toán**

**Insert**

|  |  |  |
| --- | --- | --- |
|  | **Singly** | **Doubly** |
| **Head** | **O(1)** | **O(1)** |
| **Tail** | **O(1)** | **O(1)** |
| **Middle** | **O(n)** | **O(n)** |

**Remove**

|  |  |  |
| --- | --- | --- |
|  | **Singly** | **Doubly** |
| **Tail** | **O(n)** | **O(1)** |
| **Head** | **O(1)** | **O(1)** |
| **Middle** | **O(n)** | **O(n)** |

**Stack – Ngăn sếp**

**Stack là 1 cấu trúc trừu tượng, các đối tượng trong stack phải làm việc theo 1 LIFO (LAST IN FIRST OUT) – đối tượng nào vào sau thì sẽ ra trước**

**Queue – Hàng đợi**

**Queue là 1 cấu trúc trừu tượng, các đối tượng trong queue phải làm việc theo 1 FIFO (FIRST IN FIRST OUT) – đối tượng nào vào trước thì sẽ ra trước**