* + 1. Trình bày về interchangeSort, + hỏi đáp
    2. Trình bày về SelectionSort, + hỏi đáp
    3. Trình bày về InsertionSort, + hỏi đáp
    4. Trình bày về BubbleSort, + hỏi đáp
    5. Trình bày về HeapSort, + hỏi đáp
    6. Trình bày về ShellSort, + hỏi đáp
    7. Trình bày về QuickSort, + hỏi đáp
    8. Trình bày về LinkList, + hỏi đáp
    9. Trình bày về Stack, + hỏi đáp
    10. Trình bày về Queue, + hỏi đáp
    11. Trình bày về ALVTree, + hỏi đáp

CHUẨN BỊ

Những thuật toán sắp xếp:

1/ Ý tưởng sắp xếp.

i. Interchange Sort:  
 Bắt đầu từ phần tử ở vị trí đầu, tính từ vị trí đoạn chưa được sắp xếp, so sánh với các phần tử còn lại trong danh sách.

* Trong các cặp so sánh, nếu phần tử ở vị trí đầu lớn hơn phần tử ở vị trí sau thì sẽ hoán vị.
* Ngược lại, phần tử sau lớn hơn thì không hoán vị.

ii. Selection Sort:

Thuật toán selection sort sắp xếp một mảng bằng cách đi tìm phần tử có giá trị nhỏ nhất (giả sử với sắp xếp mảng tăng dần) trong đoạn đoạn chưa được sắp xếp và đổi cho phần tử nhỏ nhất đó với phần tử ở đầu đoạn chưa được sắp xếp (không phải đầu mảng).

iii. Insertion Sort:  
 Thuật toán sắp xếp chèn thực hiện sắp xếp dãy số theo cách duyệt từng phần tử và chèn từng phần tử đó vào đúng vị trí trong mảng con (dãy số từ đầu đến phần tử phía trước nó) đã sắp xếp sao cho dãy số trong mảng sắp đã xếp đó vẫn đảm bảo tính chất của một dãy số tăng dần.

iv. Bubble Sort:

Thuật toán sắp xếp bubble sort thực hiện sắp xếp dãy số bằng cách lặp lại công việc đổi chỗ 2 số liên tiếp nhau nếu chúng đứng sai thứ tự (số sau bé hơn số trước với trường hợp sắp xếp tăng dần) cho đến khi dãy số được sắp xếp.

Duyệt từ cuối mảng đến đầu mảng. Sau mỗi lần duyệt như vậy chúng ta sẽ được phần tử nhỏ nhất lên đầu mảng.

v. Heap Sort:

Một Binary Heap là một cây nhị phân hoàn chỉnh trong đó các mục được lưu trữ theo một thứ tự đặc biệt sao cho giá trị trong nút cha lớn hơn (hoặc nhỏ hơn) so với giá trị trong hai nút con của nó.

Thuật toán:

1. Xây dựng một heap tối đa từ dữ liệu đầu vào. (duyệt từ vị trí n/2 – 1 trở về 0)

2. Tại thời điểm này, mục lớn nhất được lưu trữ ở gốc của heap. Thay thế nó bằng mục cuối cùng của heap, sau đó giảm kích thước của heap đi 1. Cuối cùng, ta có một gốc heap.

3. Lặp lại bước 2 trong khi kích thước của heap lớn hơn 1

vi. Shell Sort

Đầu tiên, giải thuật này sử dụng giải thuật sắp xếp chọn trên các phần tử có khoảng cách xa nhau, sau đó sắp xếp các phần tử có khoảng cách hẹp hơn. Khoảng cách này còn được gọi là **khoảng (interval) được tính bằng vòng lặp**

do

{

interval = 2 \* interval + 1;

} while (2 \* interval + 1 < n);

vii. Quick Sort

Thuật toán sắp xếp quick sort là một thuật toán chia để trị (Divide and Conquer algorithm). Nó chọn một phần tử trong mảng làm điểm đánh dấu (pivot). Thuật toán sẽ thực hiện chia mảng thành các mảng con dựa vào pivot đã chọn. Việc lựa chọn pivot ảnh hưởng rất nhiều tới tốc độ sắp xếp.

1. Luôn chọn phần tử đầu tiên của mảng.
2. Luôn chọn phần tử cuối cùng của mảng.
3. Chọn một phần tử random.
4. Chọn một phần tử có giá trị nằm giữa mảng (median element). (sẽ sử dụng)

viii. Linked List

\* Về bản chất, danh sách liên kết có chức năng như một mảng, có thể thêm và xóa các phần tử ở bất kỳ vị trí nào khi cần thiết.

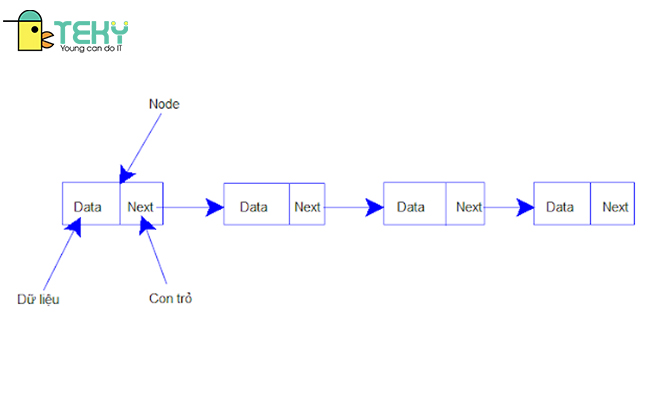
\* Tuy nhiên 1 vài điểm khác so với mảng

* *Kích thước thay đổi trong quá trình thêm/ xóa phần tử*
* *Kích thước tối đa phụ thuộc vào bộ nhớ*
* *Cấp phát bộ nhớ động: Bộ nhớ được cấp phát trong quá trình chạy Được lưu trữ trên các ô nhớ ngẫu nhiên*
* *Thứ tự và sắp xếp: Được lưu trữ trên các ô nhớ ngẫu nhiên*
* Truy cập tới phần tử ngẫu nhiên cần phải duyệt từ đầu/cuối đến phần tử đó: O(n)
* Chỉ có thể tìm kiếm tuyến tính

Ứng dụng của DSLK:

* + Tạo ra CTDL hàng đợi (Queue)
  + Tạo ra CTDL ngăn xếp (stack)
  + blockchain

Danh sách liên kết đơn là một tập hợp các Node được phân bố động, được sắp xếp theo cách sao cho mỗi Node chứa “một giá trị”(Data) và “một con trỏ”(Next). Con trỏ sẽ trỏ đến phần tử kế tiếp của danh sách liên kết đó. Nếu con trỏ mà trỏ tới NULL, nghĩa là đó là phần tử cuối cùng của linked list.



ix. Stack:

Stack là một cấu trúc dữ liệu hoạt động theo nguyên lý LIFO (Last In First Out)

Là node nào vào sau thì node đó ra trước (Giống như 1 cái hộp đựng trái cầu lông)

Stack có 2 thao tác:  
 Push: Thêm node vào trên cùng của stack   
 Pop: Lấy node trên cùng của stack ra ngoài

x. Queue:

Hàng đợi (tiếng anh: Queue) là một cấu trúc dữ liệu dùng để lưu giữ các đối tượng theo cơ chế **FIFO** (viết tắt từ tiếng Anh: *First In First Out*), nghĩa là “vào trước ra trước”.

Với cấu trúc Hàng đợi(Queue), chúng ta có các chức năng sau:

+ Thêm phần tử vào cuối của hàng đợi.

+ Xóa phần tử khỏi đầu của hàng đợi.

2/ Chạy từng bước.

3/ Code