1-

Tanto os algoritmos de merge como o de quicksort são algoritmos de divisão, ou seja os elementos são divididos ordenados e rearranjados, uma diferença clara entre os dois que utilizam a divisão é que no mergesort o array é sempre dividido na metade do seu tamanho diferente do quicksort em que ele pode ser dividido(pivô) em outra parte que não seja a metade, já o processo de heapsort usa uma estrutura de dados baseada em uma árvore binária para realizar a ordenação.

No caso do Quicksort, uma das vantagens é a complexidade do seu tempo de execução que é rápida(quick) por justamente ter um laço interno muito simples e por conta de ser muito recursivo tem o pior caso em n² e por mais que seu pior caso seja raro é uma desvantagem do método.

Já o Mergesort a sua eficiência não depende da ordem inicial dos elementos realizando em seu pior caso 39% menos comparações que o quicksort em um caso médio, porém o mesmo possui um gasto superior de memória, por conta da função merge requerer um vetor auxiliar.

Por último o heapsort tem seu comportamento sempre como O(n log n) sendo mais confiável em casos críticos, porém ele não é um algoritmo estável por não preservar a ordem de registros de chaves iguais.

Em relação a complexidade, o quicksort possui O(n²) no seu pior caso o heapsort

O(n) e o mergesort complexidade de tempo O(n) .

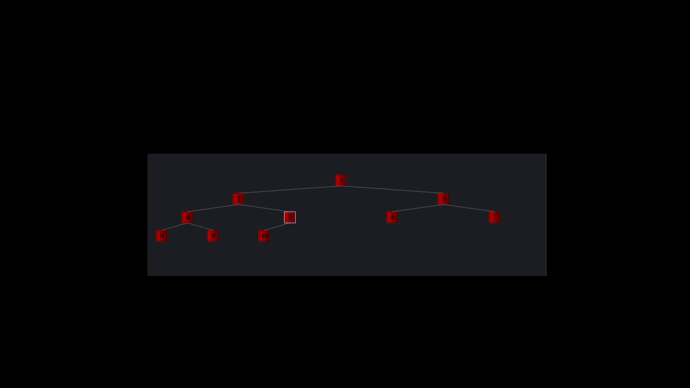
2-

O algoritmo de Heap, é justamente a estrutura de dados utilizada em forma de árvore binária completa para a organização dos elementos. Sendo o heap uma árvore binária perfeitamente balanceada onde um nó filho é sempre maior ou igual a um nó pai e as folhas de último nível estão todas nas posições mais à esquerda.

Ele é utilizado no processo de ordenação heapsort e é uma ótima alternativa para implementação de filas de prioridade e gerenciamento de memória em sistemas operacionais.

No Heapsort o seu papel é ordenar os elementos de um array, respeitando as características do heap os elementos vão ser ordenados de forma que sempre a raiz seja o maior número e os nós pais sejam maiores que os nós filhos.Com a árvore é realizado o seguinte processo:

É verificado iniciando pelos nós inferiores e comparando-os aos nós pais e caso sejam os maiores realizando a troca,por exemplo na primeira ramificação temos um nó pai 4 com duas ramificações 8 e 9 filhas,sendo ambas as filhas maiores é trocado pelo 4 a maior das duas ramificações e esse processo de comparação com nós pais e filhos vai se repetindo e sendo realizado em todas as ramificações, chegando ao fim o maior número da raiz é trocado com o último número da árvore e retirado o número reiniciando o processo, como vídeo abaixo:



3-

**Diferenças ->**

Pilhas é uma estrutura de dados LIFO (último a entrar, primeiro a sair).

Fila é uma estrutura de dados FIFO (primeiro a entrar, primeiro a sair).

Lista é dinâmica, não segue uma ordenação.

**Aplicação computacional ->**

**Listas:**

As mais diversas, como manipulação de dados, controle de registros, tamanho da lista significa o número de elementos presentes na lista. Listas encadeadas tem a vantagem de ter um tamanho variável, novos itens podem ser adicionados, o que aumenta seu tamanho.

Cada elemento numa lista possui um índice, um número que identifica cada elemento da lista. Usando o índice de um elemento da lista é possível buscá-lo ou removê-lo.

**Filas:**

Para ordenação de dados, controle de chegada, dentre outros. Fila são estruturas de dados bastante utilizadas na computação, onde o primeiro elemento a ser inserido, será também o primeiro a ser retirado. Desta forma, serão adicionados elementos no fim e removê-los pelo início.

**Pilhas:**

O último que entra é o primeiro que sai, caracterizando um empilhamento de dados. Por exemplo o Ctrl + Z do computador, utiliza de pilhas para remover a última alteração feita, e com isso voltar seu texto no editor, por exemplo. Ela é amplamente usada na arquitetura de sistemas de computação.