Estructuras de Datos y Algoritmos Fundamentales

Luis Barajas - A01235589
María Fernanda Vela Calderón - A01377958
Jaime Eduardo López Castro - A00833173

ALGORITMOS

Busqueda

Lineal: O(n)
Se recorre todos los elementos de los datos, uno por uno.

Lineal en Orden: O(n)
Se recorre todos los elementos de los datos ordenados.

Binaria/ Binaria recursiva : O(log2 n)
Divide por la mitad los datos y busca si el elemento es mayor o menor a la mitad.

BINARY SEARCH TREE

Preorden: O(n)
imprime nodo raíz, recorre preorden subárbol izq, recorre preorden subárbol derecho

Inorden: O(n)
Recorre inorder subárbol izq, visita raíz, recorre inroder el subárbol derecho

Postorden: O(n)
recorre postorden el subárbol izq, recorre en postorden subárbol derecho, visita la raíz

GRAFOS

BFS: O(|V|+|E|).
A partir de un nodo de inicio, visita un nodo y luego a todos sus vecinos, usa un Queue.

DFS: O(|V| + |E|).
A partir de un nodo de inicio, visita el nodo y luego visita recursivamente a todos su vecinos NO procesados. Usa un stack

Ordenamiento

Selection: O(n^2)
Encuentra repetidamente el elemento mínimo y colocándolo al principio.

Bubble: O(n^2)
Compara dos datos adyacentes y si están en el orden incorrecto los intercambia.

Insertion: O(n^2)
La matriz de divide en parte ordenada y no ordenada. Los datos de la parte no ordenada se colocan en posición correcta de la ordenada

Merge: O(n log n)
Divide y vencerás. Recursiva. Divide continuamente la matriz original en submatrices hasta que ya no se pueda dividir mas. Se ordenan las mitades y se funcionan.

Quick: O(n log n)
Divide y vencerás. Utiliza un pivote y divide la matriz alrededor del pivote. Compara los datos con el pivote.

MAX AND MIN HEAPS

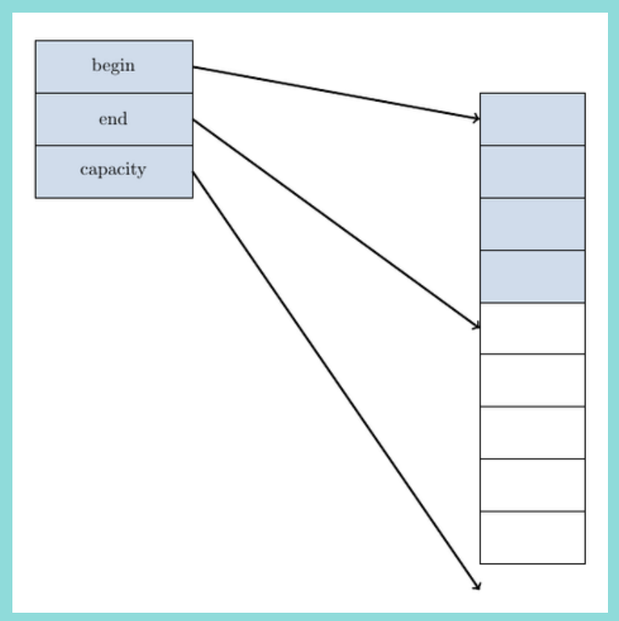
Heap sort: O(n logn)
Ordenamiento de forma ascendente o descendente. Convertir la lista en un Heap, intercambiar la raiz con la última hoja, disminuir el tamaño, restablecer el heap

GRAFOS

Dijkstra: O((|V| + |E|) log |V|).
Calcula el camino más corto entre un vértice y el resto de los nodos

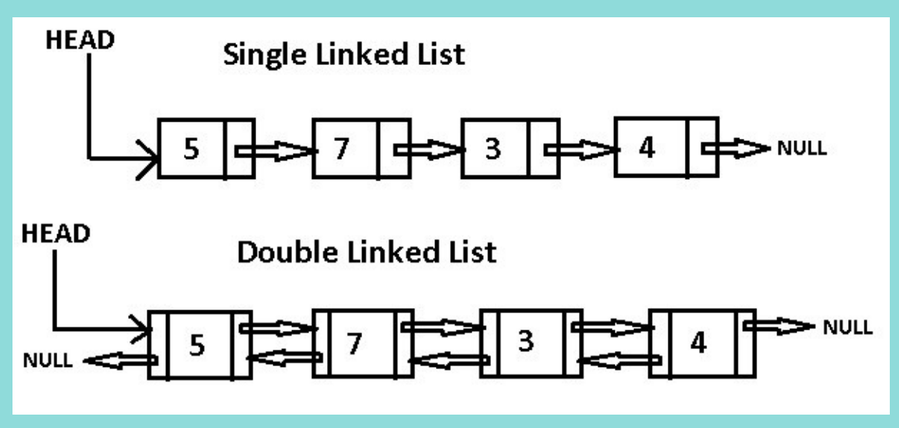
ESTRUCTURAS DE DATOS

Vectores



- Estructuras lineales y contiguas.
- Accesar elementos: O(1) - Indexado.
- Añadir/Borrar elementos: O(n).
- Buscar elementos: O(n) - O(logn)

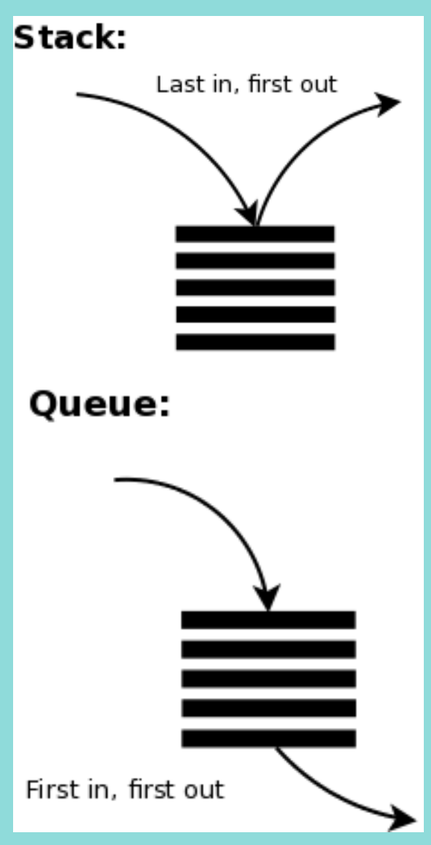
Listas enlazadas dobles y sencillas



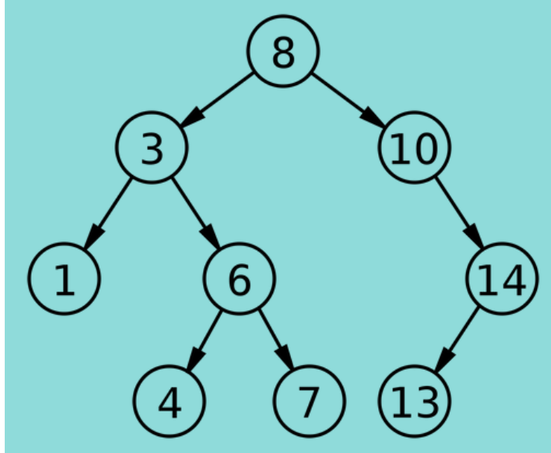
- Estructuras lineales, no necesariamente contiguas.
- Accesar elementos: O(n) - No indexado.
- Añadir/Borrar elementos: O(1) - Manejo de apuntadores.
- Buscar elementos: O(n)
- Uso de nodos

Pilas y filas (Stacks y Queues)

- Usualmente basadas en Listas Enlazadas.
- Estructuras lineales, no necesariamente contiguas.
- Accesar elementos: O(n) - No indexado.
- Añadir/Borrar elementos: O(1) - Manejo de apuntadores.
- Buscar elementos: O(n) - O(logn)
- Uso de nodos
- Stack: LIFO ----- Queue: FIFO



Árboles (BST, AVL, Splay, Heap)



BST:

- Hijo izquierdo menor, hijo derecho mayor
- Búsqueda: O(logn) ----- Añadir/Eliminar elemento: O(n) - O(logn) en promedio

AVL:

- Hijo izquierdo menor, hijo derecho mayor
- Búsqueda: O(logn) ----- Añadir/Eliminar elemento: O(logn)
- Evita el desfase de niveles mayor a 1 balanceando el árbol: O(logn)

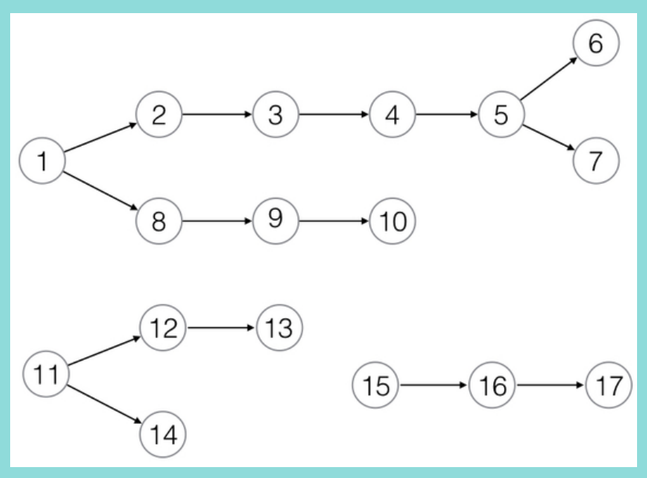
Splay Tree:

- Hijo izquierdo menor, hijo derecho mayor
- Búsqueda: O(logn) ----- Añadir/Eliminar elemento: O(n) - O(logn) en promedio
- Los elementos recientemente visitados están arriba.

MaxHeap:

- Hijos menores o iguales al padre
- Búsqueda: O(n) --- Push/Pop: O(logn)
- Muy útil para encontrar el máximo elemento de un conjunto: O(1)
- Se puede convertir en un MinHeap invirtiendo las operaciones

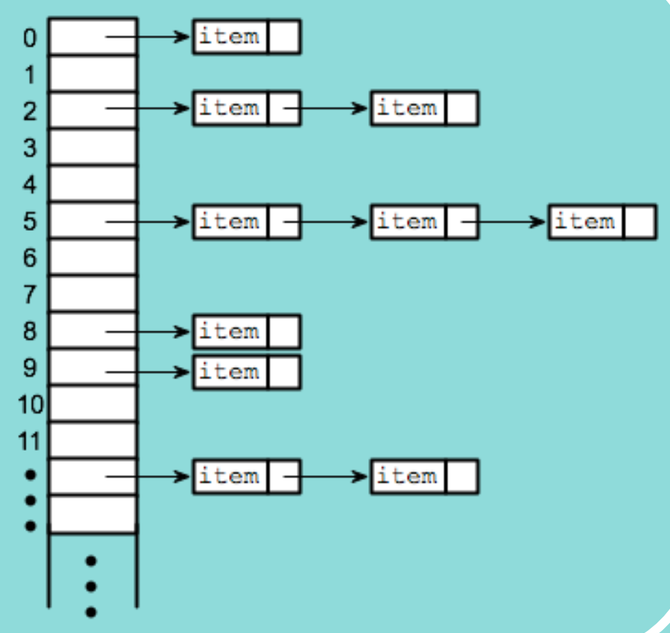
Grafos



- Perfecto para representar conexiones entre datos
- Búsqueda: O(n) --- Añadir/Eliminar elemento: O(1)
- Representación por Lista o Matriz de Adyacencia.
- Puede ser ponderado y dirigido.

Tablas Hash

- Sistema de indexado mediante una función Hash
- Puede ser de dirección abierta o cerrada
- Añadir/Buscar elementos: O(1) en promedio
- Búsqueda a través de llaves



La guerra de los bots: ataques cibernéticos

Con el aumento del uso de la tecnología en la vida cotidiana, también ha surgido un incremento en los delitos cibernéticos, uno de ellos capaz de infectar cientos de dispositivos electrónicos mediante malware para obtener su control y conectarlo a una red de dispositivos utilizada para propagar virus, cometer fraudes, generar spam, entre otros delitos, todo sin que que el dueño del dispositivo lo sepa, pues este tipo de ataque puede no generar alteraciones visibles en el funcionamiento del dispositivo.

Evidencia 1.3 - vector

- Se almacenan los registros en un vector.
- Algoritmo de ordenamiento Merge Sort - O(nlogn)
- Búsqueda Binaria - O(log2 n)

Salidas en consola

```
> make -s
> ./main
Jun 01 00:22:36
Jun 01 01:23:03
Jun 01 00:22:36 49.121.182.153:6021 Failed password for illegal user guest
Jun 01 00:34:43 254.243.231.221:7416 Failed password for illegal user guest
Jun 01 00:49:31 15.113.211.66:1795 Failed password for illegal user root
Jun 01 00:59:02 159.72.70.232:99 Failed password for illegal user guest
Jun 01 01:06:03 65.57.18.239:1163 Failed password for illegal user root
Jun 01 01:18:39 168.51.35.137:512 Illegal user
Jun 01 01:22:22 123.81.238.176:9497 Failed password for illegal user root
Jun 01 01:23:03 249.27.6.194:7341 Failed password for illegal user guest
>
```

resultado_busqueda.txt

```
Jun 01 00:22:36 49.121.182.153:6021 Failed password for illegal user guest
Jun 01 00:34:43 254.243.231.221:7416 Failed password for illegal user guest
Jun 01 00:49:31 15.113.211.66:1795 Failed password for illegal user root
Jun 01 00:59:02 159.72.70.232:99 Failed password for illegal user guest
Jun 01 01:06:03 65.57.18.239:1163 Failed password for illegal user root
Jun 01 01:18:39 168.51.35.137:512 Illegal user
Jun 01 01:22:22 123.81.238.176:9497 Failed password for illegal user root
Jun 01 01:23:03 249.27.6.194:7341 Failed password for illegal user guest
Jun 01 01:34:06 246.21.58.234:2986 Illegal user
Jun 01 01:52:37 111.89.38.165:1772 Failed password for illegal user guest
Jun 01 02:04:02 244.67.92.254:8034 Failed password for illegal user guest
Jun 01 02:10:33 186.115.187.178:7655 Failed password for illegal user root
Jun 01 02:10:34 234.204.180.48:3154 Failed password for admin
Jun 01 02:18:16 89.52.87.155:4329 Failed password for illegal user guest
Jun 01 02:18:20 128.47.216.109:6740 Failed password for illegal user root
Jun 01 02:29:09 122.135.158.172:1461 Failed password for illegal user guest
Jun 01 02:37:14 129.84.230.64:1135 Failed password for illegal user root
Jun 01 02:49:11 12.12.202.218:2548 Illegal user
Jun 01 03:23:38 88.171.169.174:6369 Failed password for admin
Jun 01 03:27:20 56.15.203.204:6494 Failed password for illegal user guest
Jun 01 03:33:55 22.109.15.199:868 Failed password for illegal user guest
Jun 01 03:34:02 13.60.65.222:2326 Failed password for illegal user root
Jun 01 03:47:00 202.112.127.217:1970 Failed password for illegal user guest
Jun 01 03:52:12 130.136.119.233:9222 Illegal user
Jun 01 03:54:52 57.134.210.237:2310 Failed password for illegal user guest
Jun 01 03:55:27 252.120.15.197:1819 Failed password for admin
Jun 01 04:04:27 49.231.173.75:8966 Failed password for illegal user root
Jun 01 04:06:11 238.56.243.235:4244 Failed password for illegal user guest
Jun 01 04:14:55 194.182.219.58:692 Failed password for admin
Jun 01 04:18:08 85.198.80.74:9438 Failed password for illegal user guest
Jun 01 04:21:48 196.71.136.156:3305 Failed password for admin

```

Evidencia 2.3 - Doubly Linked List

- Se almacenan los registros en una Lista doblemente enlazada.
- Algoritmo de ordenamiento Merge Sort - O(nlogn)
- Búsqueda Binaria - O(log n)

Evidencia 3.4 - BST

- Se almacenan los registros un vector y se ordena por IP usando Heap Sort - O(nLogn)
- Recuento de repeticiones por IP.
- Se hace push de IPs en Max Heap - O(logn)
- Aplicando cinco veces Top() y Pop() al Max Heap se obtienen las IPs con más accesos - O(logn)

Salidas en consola

```
> ./main

 ---> Creando un MaxHeap por default: 0x7ffcc05a88a0
 ---> Creando un MaxHeap por default: 0x7ffcc05a88d8
 ---> Creando un MaxHeap por default: 0x7ffcc05a88f8

    IP          Repeticiones
10.15.187.246       38
10.15.176.241       38
10.15.183.241       37
10.15.176.230       37
10.15.177.224       37

 ---> Liberando la memoria del MaxHeap: 0x7ffcc05a88f8
 ---> Liberando la memoria del MaxHeap: 0x7ffcc05a88d8
 ---> Liberando la memoria del MaxHeap: 0x7ffcc05a88a0
>
```

Evidencia 4.3 - Grafos

- Mediante el método readGraph(fileName) - O(n^2) se inicializan nodesInfo: map<unsigned int, Ip>), vectorIps: vector<Ip> y la lista de adyacencias adjList: vector<LinkedList<std::pair<int, int>>> y se va incrementando el valor del atributo adyacencias.
- Se genera un Max Heap usando como prioridad el grado del IP - O(nlogn)
- Mediante Top y Pop se imprime el Boot Master y se obtienen las cinco IPs con mayor grado -O(logn)
- Se emplea el algoritmo Dijkstra- O((|V| + |E|) log |V|)
- para obtener el camino más corto entre Boot Master e IPs.
- Se genera un Max Heap usando como prioridad el la distancia con Boot Master - O(nlogn) y con Top se obtiene la IP más lejana. - O(1)

Salidas en consola

```
> ./main

 ---> Creando un MaxHeap por default: 0x7fff2cb3b508
 ---> Creando un MaxHeap por default: 0x7fff2cb3b540
 ---> Creando un MaxHeap por default: 0x7fff2cb3b408

 ---> Liberando la memoria del MaxHeap: 0x7fff2cb3b408

Ip de Boot Master: 73.89.221.25   Grado: 18

Ip mas lejano: 244.223.133.50    Distancia: 922

 ---> Liberando la memoria del MaxHeap: 0x7fff2cb3b540
 ---> Liberando la memoria del MaxHeap: 0x7fff2cb3b508
>
```

Evidencia 5.2- Códigos Hash

- Se retoma el método readGraph(fileName) -O(n^2) de la Act. 3.4 y genera aumento en el atributo accesos.
- Se inicializa Hash table (de tipo Método de Dirección Abierta con prueba cuadrática) con capacidad máxima de numero primo siguiente de numNodes*2.
- Por cada nodo del grafo, se recupera el valor numérico de la IP y su resumen (IP, accesos, adyacencias) y se hace add a la habla Hash. - O(n)
- Impresión de cantidad de elementos, tamaño de Hash Table y cantidad total de colisiones generadas.
- Recibe IP y tras comprobar que existe en bitácora, con su key realiza find en la tabla - O(n).
- Con el índice resultante se llama a getDataAt de Hash table - O(1), se imprime el resumen de la IP - O(1) y se accede a esa posición de la lista de adyacencias para recorrer la linked list que contiene las IPs adyacentes.
- Las ingresa en un vector con el que se crea un Max Heap y con Top y Pop se imprimen en orden descendente. - O(nlogn)

Salidas en consola

```
Al insertar 13370 resumenes de IPs
en una tabla hash de tamaño 26759,
el total de colisiones generadas es de 2445 .

Ingrese la direccion IP que desea buscar
(En formato ###.##.###.### ): 73.89.221.25

Dirección IP: 73.89.221.25
Numero de aristas llegando a la IP: 7
Numero de aristas saliendo de la IP: 18

La direccion Ip reibida accedio a las siguientes direcciones Ip:
    232.206.245.1
    227.80.157.34
    211.216.72.37
    194.148.27.194
    180.33.175.33
    175.91.13.247
    170.30.170.40
    167.145.185.181
    153.110.251.73
    119.155.134.193
    117.190.87.99
    91.205.173.253
    84.37.173.55
    66.229.48.147
    39.106.157.143
    30.179.23.66
    28.45.214.107
    7.137.44.113

 ---> Liberando la memoria del MaxHeap: 0x7ffd74a2d290
 ---> Liberando la memoria del MaxHeap: 0x7ffd74a2d640
 ---> Liberando la memoria del MaxHeap: 0x7ffd74a2d608
```

Conclusión

Para detectar estas Botnets es de gran utilidad recopilar información y realizar "búsquedas optimizadas" que permitan detectar accesos maliciosos.

- Evidencia 1.3 - Vector: Aplicación de conceptos Básicos y Algoritmos Fundamentales.
- Evidencia 2.3 - Doubly Linked List: Registros dentro de un rango de tiempo.
- Evidencia 3.4 - BST: Identificación de direcciones IP infectadas (cantidad anormal de accesos).
- Evidencia 4.3 - Grafos: Identificación del Boot Master (IP con mayor cantidad de adyacencias) y dirección más difícil de atacar (IP más lejada a Boot Master).
- Evidencia 5.2 - Códigos Hash: Buscando Boot Master obtenemos las direcciones IPs que han sido accesadas por él, por lo que es probable que estén infectadas y bajo su control.