Villanueva Andrade Daniela Estefanía.

1684742

02/Octubre/2017

**Reporte de Análisis de Algoritmos.**

En ésta etapa realizamos análisis de distintos algoritmos. Vimos lo qué es una pila, una fila, un grafo y los métodos para explorar en ellos.

A continuación se presentan procesos de elaboración y resultados de la realización de los algoritmos antes mencionados.

**Análisis de algoritmos.**

Para comenzar fue importante conocer y comprender para qué es útil el análisis de algoritmos, pues bien, el objetivo principal a lograr con ello es poder estimar el tiempo y el espacio de los recursos que se emplean.

**¿Qué es una fila?**

Un conjunto de elementos ordenados de manera secuencial (Datos ordenados).

Se puede explorar en la fila a través del método BFS que nos permite saber su amplitud.

**¿Qué es una pila?**

Es una secuencia de datos que no están relacionados, sólo se puede acceder al dato superior (Datos no ordenados).

Se puede explorar en la pila a través del método DFS que nos permite saber su profundidad

**Programación de una fila y una pila.**

Los algoritmos de fila y pila nos permiten realizar la búsqueda de elementos dentro de un arreglo, permitiéndonos a su vez conocer la longitud del recorrido de cada uno de ellos, para esto creamos dos clases, fila y pila y definimos las funciones de ingresar (meter), obtener y a su los empleamos para la definición de la función longitud,

**Resultados:**

**Fila…**

**class fila:**

**def \_\_init\_\_(self):**

**self.fila = [ ]**

**def obtener(self):**

**return self.fila.pop()**

**def meter(self,e):**

**self.fila.insert(0,e)**

**return len(self.fila)**

**@property**

**def longitud(self):**

**return len(self.fila)**

**f=fila()**

**f.meter(1)**

**1**

**f.meter(2)**

**2**

**f.meter(100)**

**3**

**print(f.longitud)**

**3**

**Pila…**

**class pila:**

**def \_\_init\_\_(self):**

**self.pila = [ ]**

**def obtener(self):**

**return self.pila.pop()**

**def meter(self,e):**

**self.pila.insert(0,e)**

**return len(self.pila)**

**@property**

**def longitud(self):**

**return len(self.pila)**

**p= pila()**

**p.meter(1)**

**1**

**p.meter(2)**

**2**

**p.meter(100)**

**3**

**p.obtener()**

**1**

**p.longitud**

**2**

**¿Qué es un grafo?**

Un grafo está conformado por un par de conjuntos de los cuales uno de ellos corresponde a los vértices y el otro a las aristas que nos permite visitar a través de sus conexiones a cada uno de sus elementos.

**Programación de un grafo.**

Consiste en crear dos conjuntos, uno de aristas y otros más de vértices o nodos e ir visitando a cada uno de sus elementos (hijos, vecinos).

**Resultados obtenidos.**

**class Grafo:**

**def \_\_init\_\_(self):**

**self.V=set()**

**self.E=dict()**

**self.vecinos=dict()**

**def agrega(self,v):**

**self.V.add(v)**

**if not v in self.vecinos:**

**self.vecinos[v]=set()**

**def conecta(self,v,u,peso=1):**

**self.agrega(v)**

**self.agrega(u)**

**self.E[(v,u)]=self.E[(u,v)]=peso**

**self.vecinos[v].add(u)**

**self.vecinos[u].add(v)**

**G=Grafo()**

**G.conecta('a','b',5)**

**G.conecta('a','c',7)**

**G.conecta('b','c',2)**

**G.conecta('c','d',4)**

**print(G.vecinos['a'])**

**{'c', 'b'}**

**print(G.V)**

**{'d', 'c', 'b', 'a'}**

**print(G.E)**

**{('a', 'b'): 5, ('b', 'a'): 5, ('a', 'c'): 7, ('c', 'a'): 7, ('b', 'c'): 2, ('c', 'b'): 2, ('c', 'd'): 4, ('d', 'c'): 4}**

**BFS Y DFS**

**BFS…**

**def BFS(g,ni):**

**visitados=[]**

**f=fila()**

**f.meter(ni)**

**while(f.longitud>0):**

**na=f.obtener()**

**visitados.append(na)**

**ln=G.vecinos[na]**

**for nodo in ln:**

**if nodo not in visitados:**

**f.meter(nodo)**

**return visitados**

**DFS…**

**def DFS(g,ni):**

**f=pila()**

**f.meter(ni)**

**while(f.longitud>0):**

**na=f.obtener()**

**visitados.append(na)**

**ln=G.vecinos[na]**

**for nodo in ln:**

**if nodo not in visitados:**

**f.meter(nodo)**

**return visitados**

**Conclusiones.**

Resultaron muy interesantes las maneras en que puede ser analizado un algoritmo, cómo se puede viajar en él a través de conexiones cómo lo es en el grafo, explorar cada conjunto ya sea de una manera ordenada o no gracias a la pila y a la fila. Observamos cómo analogías para una mayor comprensión que la esencia de los algoritmos está presente en muchas cosas cotidianas, por ejemplo un videojuego.