|  |
| --- |
| ICASS |
| Evaluación ICASS – Ejemplos a Estructura de Datos |
|  |

|  |
| --- |
| VALENTÍN MACAYA RODOLFO A  20-3-2024 |

Contenido

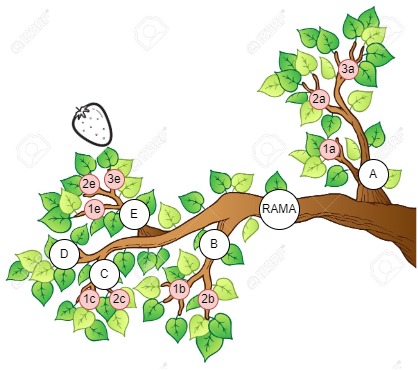
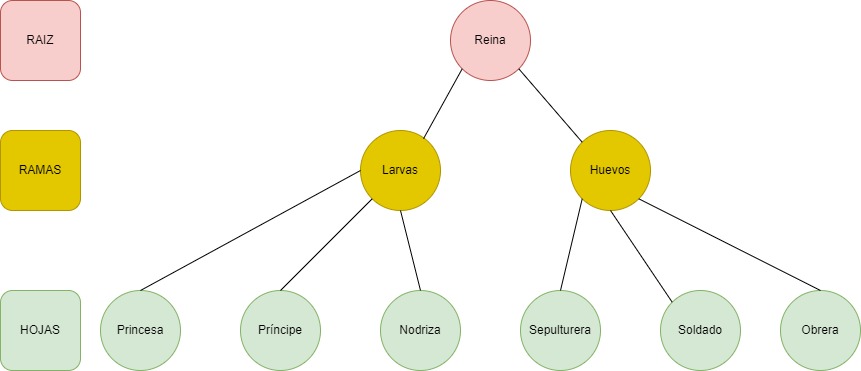
[**Ejemplos** 2](#_Toc161939377)

[**Referencias** 6](#_Toc161939378)

# **Ejemplos**

En el presente escrito se darán ejemplos de como un sistema jerárquico como lo es un hormiguero puede ser representado o modelarse virtualmente siguiendo estructuras de datos.

Se agregan como supuestos que cada una de las hormigas sigue sus tareas de manera estricta y sin excepciones, así como que cada uno de las estructuras mencionadas pueden funcionar en conjunto para algún sistema o algoritmo.

* **Array**: Un arreglo nos permite guardar distintos datos dentro de una sola “variable”, esta puede contener distintos tipos de datos como INT, STRING o FLOAT, pero es importante tener en cuenta de que estos son estáticos y debe ser definido su tamaño previamente (algunos lenguajes de programación pueden considerarse excepciones a esta norma) y que los datos ingresados dentro del mismo deben ser siempre los mismos.   
  Para el caso de un hormiguero podemos aprovechar estas características para guardar información como por ejemplo con respecto a los alimentos que se tienen almacenados, donde la variable o constante sería nuestro Alimento, y que dentro de este arreglo en cada posición puede guardar la cantidad de alimento según la posición, por ejemplo:  
    
  - ***Lista de comida:*** 0 (frambuesa), 1 (uva), 2 (cubo de azúcar), 3 (migas de pan), …- ***array:*** alimento[5, 2, 4, 1];  
    
  Con esta información, podemos saber que según la posición del array ([0,1,2,3,4,…]) podemos conocer cuantos alimentos quedan de cada cosa, en este ejemplo, podemos ver que nos quedan 5 frambuesas, 2 uvas, 4 cubos de azúcar y 1 miga de pan.  
    
  Otro ejemplo de uso se puede ver con las feromonas, donde nuestro array es feromona[] y en cada espacio se guarda una feromona distinta.
* **Lista**: La principal diferencia entre un Array y una Lista es que esta última no es estática, sino que dinámica, lo que significa que le permite modificar su tamaño según avanza el código/algoritmo, así como también permiten guardar información o valores de distinto tipo.  
  Para nuestro caso de hormiguero, una lista podría usarse a la hora de la búsqueda del alimento y los distintos caminos que debe recorrer una obrera exploradora para encontrar alimentos. Supongamos que nuestra lista se llama *caminoAlimento* y en esta lista lo que nuestra obrera quiere guardar es el camino que puede llevarla potencialmente hasta la comida, para así poder dejar un rastro de feromonas para seguir más adelante. Dicho sea el caso, supondremos que nuestro alimento se encuentra en la rama de un árbol como se muestra en la siguiente figura.  
    
     
    
  Nuestra hormiga obrera desea llegar a la frutilla, pero primero debe revisar las otras ramas hasta dar con el camino correcto, supongamos que esta hormiga ya ha recorrido por completo la rama A, por lo que nuestra lista podría partir como algo así:  
    
  - **Lista sin actualizar**: rama= [ A, 1a, 2a, 3a, ]   
    
  Pero al no tener la fruta deseada, la hormiga puede actualizar esta lista y eliminar aquellos valores que no le sean de utilidad, que en este caso sería toda la rama A, para esto la hormiga puede hacer uso del comando *remove().  
    
  - rama.remove(A) | rama.remove(1a) | rama.remove(2a) | rama.remove(3a)*Ahora supongamos que nuestra obrera ya pasó por todas las otras ramas menos la rama E que es justamente donde está la frutilla que tanto anhela, para agregar la ruta de esta rama ella usaría:  
    
  - *rama.append(E) | rama.append(1e) | rama.append(2e) | rama.append(3e)*  
  Y por fin encontraría la fruta, aunque como debe volver a la colonia o avisarle a sus amigas, primero debe limpiar la lista para que los datos y la ruta sean los correctos para llegar a la fruta, por lo que esta usaría un *remove()* para quitar los valores innecesarios y su ruta quedaría tan simple como:  
    
  - *rama = [E, 3e]*
* **Cola**: Una cola o una “*queue*” es una estructura de datos que sigue la filosofía FIFO (First In First Out), que quiere decir que sigue la idea de que aquel que llega primero, es el primero en salir. Esta estructura espera recibir datos en un orden de llegada, donde aquellos que son recibidos primero serán asegurados de salir de los primeros.  
  Para nuestro caso de hormiguero esto lo podemos representar en la forma en cómo las hormigas obreras recolectan los alimentos. Supongamos que tenemos a una hormiga que salió a explorar y encuentra comida, esta se devolvería a la colonia o a buscar a sus compañeras para pedir ayuda a traer la comida siguiendo el rastro de feromonas.  
  Entonces, al llegar al alimento (que para este caso diremos que son migajas de pan), nuestra estructura se podría ver algo como esto:  
    
  - **Se crea la cola**: *pan = Queue();  
  -* **Llegan las hormigas**: *pan.put(‘obreraA’); | pan.put(‘obreraB’); | pan.put(‘obreraC’); | ...*  
  Si quisiéramos ver cuantas hormigas están entrando a la cola de *pan*, podemos hacer uso (en Python) del comando .qsize():  
    
  - O**btener cantidad de hormigas**: print(*pan.qsize()) | “5”*  
  Ahora, si quisiéramos que las hormigas comiencen su regreso a la colonia con el alimento, debemos respetar el orden de llegada de cada hormiga, por lo que al usar el comando *get* siempre obtendremos a aquellos elementos que se encuentren primeros en la cola *pan*.  
    
  - ***Desencolar cola***: *pan.get() | ‘obreraA’*De esta manera, obtenemos el mismo orden que de salida de hormigas.
* **Arboles**: Los arboles son una forma de estructura de datos compleja pero que permite almacenar nodos de manera jerárquica y no lineal como lo pueden llegar a ser las listas o las colas. Los arboles poseen distintos elementos, entre los cuales podemos encontrar a los mismos nodos, que son los elementos que componen al árbol.  
  Se posee un ***nodo raíz*** que le da el punto de inicio a los demás nodos hijos.  
  Los ***nodos hijos*** son aquellos nodos que poseen un nodo superior a estos.  
  ***Nodo hermano*** son aquellos que poseen a otros nodos compartiendo en el mismo nivel de la jerarquía.  
  Los ***nodos*** ***rama*** se les dice a todo aquel nodo que no son raíces pero que poseen por lo menos un nodo hijo.  
  Los ***nodos hoja*** se tratan de los nodos de más abajo en la jerarquía, aquellos que no poseen hijos.  
    
  Bajo estos conceptos, es fácil ver el cómo podemos relacionar a los árboles con el concepto de hormiguero, ya que cada una de las integrantes de la Colonia le deben la vida a su reina, por la que esta actuaría como nuestra nodo Raíz, a continuación la seguirián sus larvas o huevos que actuarían como nodos rama, y finalmente todo el resto de la jerarquía funcionarían como los nodos hoja del sistema.  
  Esta descripción se puede ver en mayor detalle en la siguiente imagen.  
  
* **Diccionarios**: Una lista nos permite guardar distintos datos, necesitando una llave y un parámetro (o clave-valor), donde cada clave está asociado a un valor en específico. Estas claves son únicas por lo que no deben o pueden existir claves iguales. En el contexto de un hormiguero, podemos utilizar esta estructura de datos en una gran variedad de casos, pero el que más fácil puede verse representado es el sistema del mismo Hormiguero.  
  Nuestro Hormiguero sería nuestro diccionario, y como ítems del mismo podríamos encontrar:  
    
  - *hormiguero = { “Hormiga Reina”: 1, “Hormiga Princesa”: 4, “Hormiga Príncipe”: 6, “Hormiga Nodriza”: 50, “Huevos/Larvas”: 150, “Hormiga Obrera”: 845, “Hormiga Soldado”: 457, “Hormiga Sepulturera”: 125, “Alimento Dulce”: 50, “Alimento Salado”: 96, “Días sin ataques a la colonia”: 451}*  
  Como se puede apreciar en el ejemplo, el hormiguero contiene información variada de varios aspectos del mismo, desde la cantidad de hormigas de todo tipo, así como también la cantidad de tipo de comida que queda en la colonia. Cada una de las claves son únicas y no se repiten, en cambio los valores de estos pueden repetirse sin problemas, e incluso de ser necesario, el usuario puede acceder a los datos de este diccionario Hormiguero y consultar solo algunos datos o incluso modificarlos a su gusto. Por ejemplo, supongamos que la colonia fue atacada recientemente, y en dicho ataque se perdieron 50 soldados, 20 obreras y 50 huevos, entonces el usuario puede ingresar los siguientes parámetros para modificar dichos ítems:  
    
  - *hormiguero [“Hormiga Soldado”] = 450 | hormiguero [“Hormiga Obrera”] = 825 | hormiguero[“Huevos/Larvas”] = 100 | hormiguero [“Días sin ataques a la colonia”] = 0*

# **Referencias**

* Ender Look. (s. f.). *¿Cuál es la diferencia entre una lista y un arreglo programación?* Quora. <https://es.quora.com/Cuál-es-la-diferencia-entre-una-lista-y-un-arreglo-programación>
* Oblancarte. (2014, 1 agosto). *Estructura de datos - Queue (Cola) – Oscar  
   Software Architecture*. Oscar Blancarte - Software Architecture  
  <https://www.oscarblancarteblog.com/2014/08/01/estructura-de-datos-queue-cola/>
* Oblancarte. (2020, 27 agosto). *Estructura de datos - Árboles - Oscar Blancarte - Software Architecture*. Oscar Blancarte - Software Architecture. <https://www.oscarblancarteblog.com/2014/08/22/estructura-de-datos-arboles/>