Estructuras de Datos

Julio 27 / 2023

Listas:

- Secuencia de valores que puedan ser recorridos arbitrariamente en el orden en el que están en la secuencia.
- Suelen ser implementados mediante estructuras enlazadas (listas enlazadas)
- Operaciones:
  - Agregar elemento al inicio: appendleft O(1)
  - Agregar elemento al final: append O(1)
  - Eliminar elemento al inicio: popleft O(1)
  - Eliminar elemento al final: pop O(1)
  - Insertar en posición arbitraria: insert O(k)
  - Eliminar de posición arbitraria: erase O(k)
  - Tamaño: len O(1)
  - Determinar si está vacía: len(l) > 0 O(1)

- Creación:

```
from collections import deque
l = deque()
```

Pilas:

- Secuencia de valores en la que solo se tiene acceso por un extremo (tope)
- LIFO: Last In First Out
- Típicamente se implementan como listas enlazadas
- Operaciones:

push	append	O(1)	deque	O(1) amort.	listas nativas
top	l[-1]	O(1)		O(1)	
pop	pop	O(1)		O(1)	
empty	len	O(1)		O(1)	
size	len(l) > 0	O(1)		O(1)	

Colas:

- Secuencia de valores en la que se tiene acceso por 2 extremos, uno para agregar elementos, otro para remover elementos.
- FIFO: First In First Out
- Típicamente se implementan como listas enlazadas
- Operaciones:

push	append	O(1)
front	l[0]	O(1)
pop	popleft	O(1)
size	len	O(1)
empty	len(l) > 0	O(1)

Tablas de Direccionamiento Directo:

* Son estructuras de datos en las que se asocian claves a valores. Estas claves determinan la ubicación en la que se almacenan los valores.
* Es requerido que en la implementación se pueda obtener de forma eficiente los valores a partir de sus claves.
* Los arreglos (vectores) y mapas en C++, y las listas nativas y diccionarios de Python son implementaciones de esta estructura de datos.
* Operaciones:
  * assign
  * update
  * query

Operación	Python (listas nativas)	Python (Diccionarios)
assign	`t = []` `t.append(10)` O(1) amort. `t[0] = 10` O(1)	`t = dict()` # t = {} `t["hola"] = 5` O(1) amort.
update	`t[0] = 8` O(1)	`t["hola"] = 5` O(1)
query	`t[pos]` O(1)	`t[clave]` `clave in t` O(1)

Colas de Prioridad

* Corresponden a colas en las que los elementos no salen en el orden en el que entraron sino de acuerdo a una prioridad.
* La prioridad es determinada con respecto a algún criterio que puede ser el valor mismo del elemento o alguna cantidad que se pueda calcular sobre el valor del elemento.
* Suelen ser implementadas mediante montículos (heaps), árboles binarios de búsqueda balanceados u otras estructuras que soporten operaciones $O(\log n)$.
* Operaciones:
  * push `heappush(t,10)` $O(\log n)$
  * top `t[0]` $O(1)$
  * pop `heappop(t)` $O(\log n)$
  * size `len(t)` $O(1)$
  * empty `len(t)==0` $O(1)$
* Creación:

```
from heapq import heappush, heappop
t = []
```

Conjuntos:

- Agrupaciones de elementos en las que no hay elementos repetidos y no hay orden para los elementos.
- Asociación con conjuntos matemáticos
- Operaciones:

insert	add	O(1)
erase	discard, remove	O(1)
empty	len(s) > 0	O(1)
size	len	O(1)