Córdoba 7 de Junio del 2022

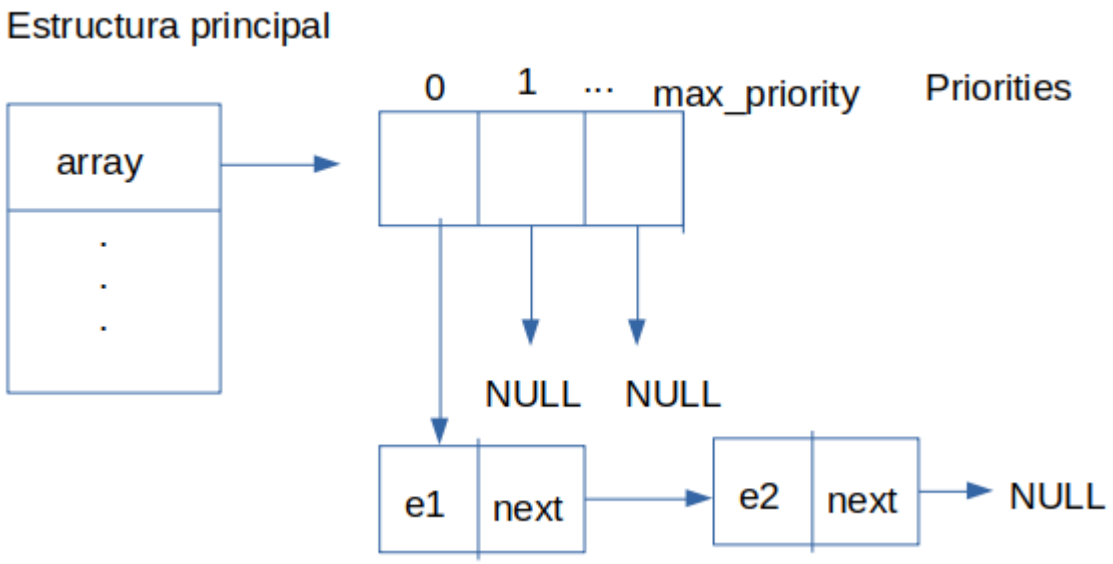
Segundo Parcial de Laboratorio

Algoritmos y Estructura de Datos II

TEMA D

Ejercicio

Implementar el TAD ***pstack*** que representa una pila de prioridades. Una pila de prioridades (**pstack**) es un tipo especial de pila en la que cada elemento está asociado con una prioridad asignada. En una pila de prioridades un elemento con mayor prioridad será desapilado antes que un elemento de menor prioridad. Sin embargo, si dos elementos tienen la misma prioridad, se desapilarán siguiendo el orden habitual de la pila. En este examen vamos a implementar **pstack** usando una estructura principal que contendrá un array de punteros a nodo por cada prioridad desde 0 hasta MAX_PRIORITY con la siguiente estructura:



La estructura principal contiene un array dinámico de punteros a nodo. Cada posición del array representa una prioridad siendo 0 la prioridad más baja y MAX_PRIORITY la más alta.

AYUDA: Puede resultar útil guardar elementos adicionales en la estructura principal como por ejemplo max_priority. De esa manera podremos saber si podemos guardar un elemento con una prioridad determinada. Toda prioridad debe ser a lo sumo max_priority o menor.

El TAD ***pstack*** tiene la siguiente interfaz

Función	Descripción
`pstack pstack_empty(priority_t max_priority)`	Crea una pila de prioridades vacía para almacenar hasta max_priority (inclusive)
`pstack pstack_push(pstack s, pstack_elem e, priority_t priority);`	Inserta un elemento a la pila con su correspondiente prioridad.
`bool pstack_is_empty(pstack s);`	Indica si la pila de prioridades está vacía
`size_t pqueue_size(pstack s)`	Obtiene el tamaño de la pila de prioridades
`pstack_elem pstack_top(pstack s)`	Obtiene el elemento con mayor prioridad
`priority_t pstack_top_priority(pqueue q)`	Obtiene el valor de la prioridad del elemento con mayor prioridad.
`pstack pstack_pop(pstack s)`	Quita un elemento con mayor prioridad más recientemente apilado.
`pstack pstack_destroy(pstack s)`	Destruye una instancia del TAD ***pstack***

En **`pstack.c`** se da una implementación incompleta del TAD ***pstack*** que deben completar **siguiendo la representación explicada anteriormente.** Además deben asegurar que la función **`pstack_size()`** sea de orden constante (*O(1)*). Por las dudas se aclara que no es necesario que la función `pstack_push()` sea de orden constante ya que puede ser muy complicado lograrlo para la representación que se utiliza y no recomendamos intentarlo.

Para verificar que la implementación del TAD funciona correctamente, se provee el programa (**main.c**) que toma como argumento de entrada el nombre del archivo cuyo contenido tiene por encabezado "**max_priority:** <int>" con la máxima prioridad permitida en el archivo y sigue el siguiente formato:

<number>	<priority>

El archivo de entrada es una lista de elementos con prioridades de atención. Entonces el programa lee el archivo, carga los datos en el TAD ***pstack*** y finalmente muestra por pantalla la pila de prioridades de los pacientes.

El programa resultante no debe dejar *memory leaks ni lecturas/escrituras inválidas*.

Una vez compilado el programa puede probarse ejecutando:

```
$ ./pstack_test inputs/example_a.in
```

Obteniendo como resultado:

```
length: 4
[ (686, 4), (345, 3), (456, 2), (454, 1)]
```

Max_priority: 4

Otro ejemplo de ejecución:

```
$ ./pstack_test inputs/example_b.in
length: 7
[ (700, 2), (600, 2), (500, 2),
  (400, 2), (300, 2), (200, 2),
  (100, 2)]
```

Max_priority: 2

Otro ejemplo:

```
$ ./pstack_test inputs/example_c.in
length: 7
[ (234,4), (345, 3), (454, 1),
  (686, 1), (789, 1), (456, 1),
  (234, 1)]
```

Consideraciones:

- Se provee el archivo **`Makefile`** para facilitar la compilación.
- Se recomienda usar las herramientas **`valgrind`** y **`gdb`**.
- Si el programa no compila, no se aprueba el parcial.
- Los *memory leaks* bajan puntos
- Entregar código muy improlijo puede restar puntos
- Si **`pstack_size()`** no es de orden constante baja muchísimos puntos
- Para promocionar **se debe** hacer una invariante que chequee la propiedad fundamental de la representación de la pila de prioridades.