Algoritmos y estructuras de datos

Pilas en arreglos

Francisco Javier Zaragoza Martínez

Universidad Autónoma Metropolitana Unidad Azcapotzalco
Departamento de Sistemas

Universidad Autónoma Metropolitana
Casa abierta al tiempo Azcapotzalco

Posgrado en Optimización

30 de abril de 2021

Citas

Larry Niven

Cualquier cosa es mejor que una pila de papel caro.

Bill Gates

Sin importar si estoy en la oficina, en casa o en el camino, siempre tengo conmigo una pila de libros que quiero leer.

Michael Carbonaro

Descubrí que si apilo los momentos correctamente, la gente creerá en las cosas más disparatadas.

Stephanie Szostak

Mi nombre es polaco. La mayoría lo dice *Zaw-stak* pero es *Show-stack*, como si fueras a un espectáculo comiendo una pila de *hot cakes*.

Tipo de datos abstracto pila

Operaciones de pilas

Una pila s tiene dos operaciones:

1. apila(s, x) agrega el dato x a la pila s y
2. desapila(s) regresa el último dato agregado a s y lo elimina.

Adicionalmente queremos crear y destruir una pila y saber si está vacía.

Nota de traducción

En inglés esto se llama *stack* (pila), *push* (empuja) y *pop* (aparece de súbito). Al último elemento agregado se le llama el tope de la pila (*top*). Una pila también se conoce como LIFO por las siglas de *Last In First Out* (lo último que entra es lo primero que sale). Alan Turing llamó originalmente a las operaciones *bury* (entierra) y *unbury* (desentierra).

Tipo de datos abstracto pila

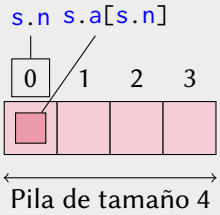
Aplicaciones

- Si apilas una secuencia $a, b, \ldots, z$ y luego la desapilas sale al revés.
- La operación *deshacer* deshace la última operación hecha.
- Evaluación de expresiones (en particular, notación polaca inversa).
- Cuando se hace una llamada a una función se anota en una pila la dirección de regreso. Así, cuando termina la ejecución de la función llamada, se regresa al último lugar del que se llamó. Ya sea en microprocesadores reales o en máquinas virtuales.
- Importante para simular funciones recursivas sin recursión.
- Para almacenar datos cuando el orden de recuperación no importe.

Tipo de datos abstracto pila

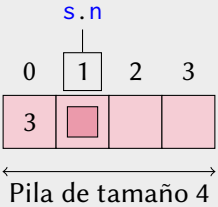
Ejemplo

Crea una pila vacía *s*



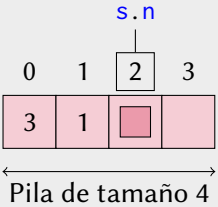

```
s = creaPilaAE(4);
```

Apila 3 en *s*




```
apilaAE(&s, 3);
```

Apila 1 en *s*

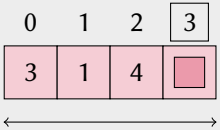



```
apilaAE(&s, 1);
```

Tipo de datos abstracto pila

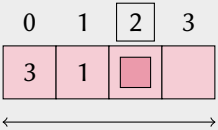
Ejemplo

Apila 4 en s



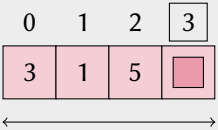

```
apilaAE(&s, 4);
```

Desapila x de s




```
desapilaAE(&s, &x);
```

Apila 5 en s




```
apilaAE(&s, 5);
```

Pila en un arreglo

Estructura de datos

Necesitamos almacenar tres datos: el tamaño del arreglo, la cantidad de elementos del arreglo y el arreglo. Esto lo podemos hacer así:

```
typedef struct {
  int max; // maxima cantidad de elementos
  int n;   // cantidad actual de elementos
  int *a;  // apuntador a un arreglo
} pilaAE;  // AE = arreglo estatico
```

Por supuesto, también lo podemos hacer así:

```
typedef arreglo pilaAE; // AE = arreglo estatico
```

Nos haremos cargo por separado de pedir y liberar la memoria del arreglo.

Pila en un arreglo

Pilas vacías

Crear una pila vacía, destruirla y saber si está vacía se hace fácilmente:

```
pilaAE creaPilaAE(int max) {
  return creaArreglo(max);
}

void destruyePilaAE(pilaAE *s) {
  destruyeArreglo(s);
}

int esVaciaPilaAE(pilaAE *s) {
  return (s->n == 0); // falso si tiene elementos
}
```

Pila en un arreglo

Agregar un dato a una pila

Hagamos una función que agregue un elemento a una pila si es que cabe. Debe avisar si no lo puede agregar por falta de espacio. Observa que el contador n también nos dice cuál es el lugar del arreglo donde debería ir el siguiente dato. A este contador también se le llama el *tope* de la pila.

```
int apilaAE(pilaAE *s, int x) {
  if (s->n == s->max) // si la pila esta llena
    return 0;         // no se pudo agregar x
  s->a[s->n] = x;     // pon x en el tope de s
  s->n++;             // un elemento mas en s
  return 1;           // si se pudo agregar x
}
```

Como la pila se modifica se manda por referencia.

Pila en un arreglo

Desapilar un dato de una pila

Hagamos una función que regrese por *referencia* el último elemento de una pila si es que hay. Debe avisar si no se puede porque la pila está vacía.

```
int desapilaAE(pilaAE *s, int *x) {
  if (s->n == 0)    // si la pila esta vacia
    return 0;       // no se pudo desapilar
  s->n--;           // un elemento menos en s
  *x = s->a[s->n]; // toma *x del tope de s
  return 1;         // si se pudo desapilar
}
```

Como la pila se modifica se manda por referencia.

Arreglos estáticos y dinámicos

Arreglos de tamaño constante (estáticos)

Un arreglo pedido con memoria estática, es decir, declarado como `int a[MAX]` donde `MAX` es una constante, no puede cambiar de tamaño. Un arreglo pedido con memoria dinámica, es decir, declarado como `a = (int *)malloc(MAX*sizeof(int))` también tiene capacidad para `MAX` elementos, pero sí puede cambiar de tamaño.

Arreglos de tamaño variable (dinámicos)

Si `p` apunta a un bloque de `b` bytes pedido con memoria dinámica, entonces `q = realloc(p, c)` intenta cambiar el tamaño del bloque a `c` bytes.

- Si lo logra, `q` apunta a ese bloque y los primeros `min(b,c)` bytes tienen el mismo contenido que antes (si `p` y `q` son distintos, entonces se hizo copiando esos bytes).
- Si falla, `q` vale `NULL` y el bloque original no cambia.

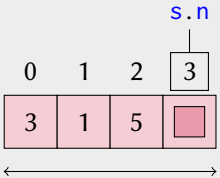
Pila en un arreglo dinámico

Ejemplo

Declaración de tipo

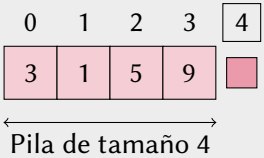
```
typedef arreglo pilaAD; // AD = arreglo dinamico
```

Pila casi llena



Con alguna operación anterior.

La pila se llena

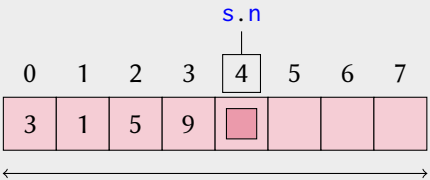


Con `apilaAD(&s, 9);`

Pila en un arreglo dinámico

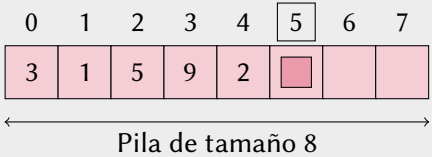
Ejemplo

Primero crece la pila



Con `apilaAD(&s, 2);`

Y luego pone el dato



Pila después de crecer y apilar.

Pila en un arreglo dinámico

Apilar un dato

Si la pila está llena, duplicamos su capacidad.

```
int apilaAD(pilaAD *s, int x) {
  if (s->n == s->max) {                  // si la pila esta llena
    int *q = (int *) realloc(s->a, 2*(s->max)*sizeof(int));
    if (q == NULL)                       // si falla duplicar a[]
      return 0;                          // no se pudo agregar x
    s->max = 2*(s->max);                 // actualiza el tamano y
    s->a = q;                            // la ubicacion de a[]
  }
  s->a[s->n] = x;                        // pon x en el tope de s
  s->n++;                                // un elemento mas en s
  return 1;                              // si se pudo agregar x
}
```

Pila en un arreglo dinámico

Desapilar un dato

Si la pila queda a un cuarto de su capacidad la reducimos a la mitad.

```
int desapilaAD(pilaAD *s, int *x) {
  if (s->n == 0)                  // si la pila esta vacia
    return 0;                     // no se pudo desapilar
  s->n--;                         // un elemento menos
  *x = s->a[s->n];                // toma *x del tope de s
  if (s->n <= (s->max)/4) {       // si usa un cuarto de a
    s->max = (s->max)/2;          // reduce a[] a la mitad
    s->a = (int *) realloc(s->a, (s->max)*sizeof(int));
  }
  return 1;                       // si se pudo desapilar
}
```

Dos representaciones de pilas

Resumen de resultados

Cantidad de copias de datos en el peor de los casos, actuando sobre una pila s de hasta n elementos y un elemento x.

Operación	Arreglo estático	Arreglo dinámico
crear	1	1
destruir	1	1
vacía	1	1
apilar	1	$n + 1$
desapilar	1	1

El $n + 1$ de la pila en un arreglo dinámico se debe a que a veces, cuando crece, se deben copiar los n datos que ya estaban. ¿Qué tan malo es esto?

Pila en un arreglo dinámico

Cantidad de copias adicionales promedio

Supongamos que comenzamos con una pila de tamaño 1. ¿Cuántas copias adicionales se hacen al apilar el dato i? ¿Cuántas copias adicionales en total y en promedio se han hecho hasta ese momento?

	1	2	3	4	5	6	7	8	9
copias	0	1	2	0	4	0	0	0	8
total	0	1	3	3	7	7	7	7	15
promedio	0.00	0.50	1.00	0.75	1.40	1.17	1.00	0.88	1.67

No es difícil convencerse de que el promedio sube cuando $i = 2^k + 1$ con $k \geq 0$ y baja el resto del tiempo. Entonces, el total de copias adicionales es $2^{k+1} - 1$ y el promedio es $\frac{2^{k+1}-1}{2^k+1} = 2 - \frac{3}{2^k+1} < 2$. No es tan malo.

Pila en un arreglo dinámico

Ejercicios

1. Escribe una función `int tamano(arreglo *s, int t)` que cambie el tamaño de un arreglo `*s` a `t` si es posible, es decir, si su contenido cabe y hay suficiente memoria.
2. Reescribe `apilaAD` y `desapilaAD` para usar esa función.
3. Algunas implementaciones de arreglos dinámicos, en lugar de aumentar el tamaño al doble, lo aumentan un 50 %. ¿Cuál sería el nuevo promedio de copias adicionales?
4. Algunas implementaciones de arreglos dinámicos, en lugar de aumentar el tamaño al doble, lo aumentan en un factor de 1.125. ¿Cuál sería el nuevo promedio de copias adicionales?
5. Algunas personas sugieren que el factor de aumento debiera ser la razón dorada $\varphi = \frac{1}{2}(1 + \sqrt{5}) \approx 1.618$. ¿Cuál sería el nuevo promedio de copias adicionales?