Algoritmos y estructuras de datos

Colas y colas dobles 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

3 de mayo de 2021

Citas

Robert A. Heinlein

Haga esto. No haga aquello. Regrese a la fila… Fórmese y pague su multa.

Julian Barnes

Si estos fueran los espíritus de los ingleses y las inglesas que han pasado a mejor vida, seguramente sabrían cómo formarse correctamente en fila.

George Mikes

Un inglés, aún si está solo, forma una fila ordenada de uno.

Orson Scott Card

Parece que los holandeses ahora se enorgullecen de ser mejores para formarse que los ingleses, lo que es absurdo, pues pararse alegremente en fila es el deporte nacional inglés.

Tipo de datos abstracto cola

Operaciones de colas

Una cola s tiene dos operaciones:

1. forma(s, x) agrega el dato x a la cola s y
2. desforma(s) regresa el primer dato agregado a s y lo elimina.

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

Nota de traducción

En inglés se llama *queue* (fila), *enqueue* (forma) y *dequeue* (desforma). También se conoce como FIFO por las siglas de *First In First Out* (lo primero que entra es lo primero que sale). Aunque una cola también puede ser una fila, en realidad una fila tiene cola (por donde uno se forma).

Tipo de datos abstracto cola

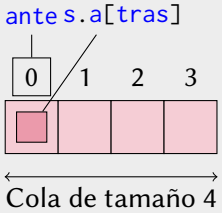
Aplicaciones

- Sistemas donde el orden de atención coincide con el orden de llegada.
- Cola de impresión (recurso compartido de uso exclusivo).
- Cola de procesos (en el planificador de tareas).
- Mensajes entre procesos.
- Atención de solicitudes (web o bases de datos).
- Productores y consumidores de datos (*buffer* del teclado).

Tipo de datos abstracto cola

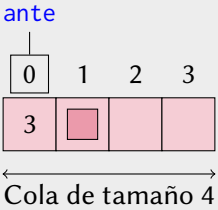
Ejemplo

Crea una cola vacía *s*



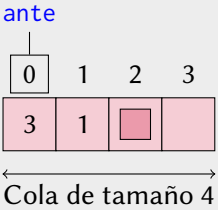

```
s = creaColaAE(4);
```

Forma 3 en *s*




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

Forma 1 en *s*

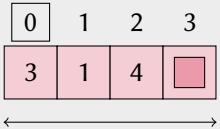



```
formaAE(&s, 1);
```

Tipo de datos abstracto cola

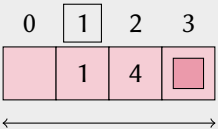
Ejemplo

Forma 4 en s



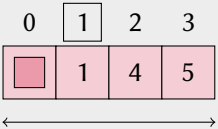

```
formaAE(&s, 4);
```

Desforma x de s




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

Forma 5 en s




```
formaAE(&s, 5)
```

Tipo de datos abstracto cola doble

Generalización de pilas y colas

Una pila es una fila donde los datos entran y salen del mismo lado. Una cola es una fila donde los datos entran y salen por lados opuestos. Una generalización permitiría que los datos entren y salgan por cualquier lado.

Operaciones de colas dobles

Una cola doble s tiene dos lados (ante y tras) y cuatro operaciones:

1. adelante(s, x) y atrás(s, x) agregan el dato x a los lados ante y tras de s, respectivamente, y
2. delante(s) y detrás(s) regresan el primer dato de los lados ante y tras de s, respectivamente, y lo eliminan.

Además queremos crear y destruir una cola doble y saber si está vacía.

Tipo de datos abstracto cola doble

Generalización de pilas y colas

Cola doble como pila

Escogemos un lado, por ejemplo tras, y entonces atrás(s, x) hace la función de apila(s, x) y detrás(s) hace la función de desapila(s).

Cola doble como cola

En este caso, atrás(s, x) hace la función de forma(s, x) y delante(s) hace la función de desforma(s).

Nota de traducción

En inglés se llama *deque* (de *Double Ended QUEue*, fila doblemente acabada o con dos lados o cabos), con lados *back* (atrás) y *front* (enfrente). Las operaciones se suelen llamar *push back*, *push front*, *pop back* y *pop front* (como si hubiera sido una pila con dos lados).

Cola doble en un arreglo estático

Estructura de datos

Necesitamos almacenar cinco datos: el tamaño máximo del arreglo, la cantidad de elementos del arreglo, los índices de ante y tras y el arreglo. Esto lo podemos hacer así:

```
typedef struct {
  int max;  // maxima cantidad de elementos
  int n;    // cantidad actual de elementos
  int ante; // indice para el lado ante
  int tras; // indice para el lado tras
  int *a;   // apuntador a un arreglo
} cdoble;
```

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

Cola doble en un arreglo estático

Crear una cola doble vacía de tamaño dado

```
cdoble creaCdoble(int max) {
  cdoble s;
  s.a = (int *) malloc(max*sizeof(int));
  s.max = (s.a != NULL) ? max : 0;
  s.n = s.ante = s.tras = 0;
  return s;
}
```

Cola doble en un arreglo estático

Colas dobles vacías

Saber si una cola doble está vacía

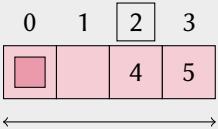
```
int esVaciaCdoble(cdoble *s) {
  return (s->n == 0); // falso si tiene elementos
}
```

Destruir una cola doble

```
void destruyeCdoble(cdoble *s) {
  free(s->a);  // libera el arreglo
  s->a = NULL; // no lo uses otra vez
  s->max = 0;  // no le caben elementos
  s->n = 0;    // no tiene elementos
  s->ante = s->tras = 0;
}
```

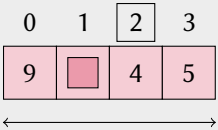
Cola doble en un arreglo estático

Arreglos circulares



Cola doble de tamaño 4

`delanteAE(&s, &x);`



Cola doble de tamaño 4

`atrasAE(&s, 9);`

Si hemos agregado atrás tantos datos que llegamos al final del arreglo, entonces parece que debemos aumentarlo de tamaño. No es necesario si hemos quitado datos delante.

- Si hay datos, `ante` señala al de delante.
- Además `tras` señala después de detrás.
- Los datos están en las posiciones `ante`, `ante`+1, ..., `tras`−1 (módulo `max`).
- Sin datos, `ante` y `tras` son iguales.
- Si `ante` o `tras` rebasan el fin del arreglo se vuelven `0`. Si `ante` o `tras` rebasan el inicio del arreglo se vuelven `max`−1.

Cola doble en un arreglo estático

Agregar un dato a una cola doble del lado tras

```
int atrasAE(cdoble *s, int x) {
  if (s->n == s->max) // si la cola esta llena
    return 0;         // no se pudo agregar x
  s->a[s->tras] = x;  // pon x en el tras de s
  s->tras++;          // mueve tras hacia atras
  if (s->tras == s->max)
    s->tras = 0;      // arregla tras si rebasa
  s->n++;             // un elemento mas en s
  return 1;           // si se pudo agregar x
}
```

Cola doble en un arreglo estático

Agregar un dato a una cola doble del lado ante

```
int adelanteAE(cdoble *s, int x) {
  if (s->n == s->max) // si la cola esta llena
    return 0;         // no se pudo agregar x
  s->ante--;          // mueve ante adelante
  if (s->ante < 0)    // arregla ante si rebasa
    s->ante = s->max-1;
  s->a[s->ante] = x;  // pon x en el ante de s
  s->n++;             // un elemento mas en s
  return 1;           // si se pudo agregar x
}
```

Cola doble en un arreglo estático

Eliminar un dato de una cola doble del lado tras

```
int detrasAE(cdoble *s, int *x) {
  if (s->n == 0)        // si la cola esta vacia
    return 0;           // no se pudo eliminar
  s->tras--;            // mueve tras delante
  if (s->tras < 0)      // arregla tras si rebasa
    s->tras = s->max-1;
  *x = s->a[s->tras]; // toma x de atras de s
  s->n--;               // un elemento menos en s
  return 1;             // si se pudo eliminar
}
```

Cola doble en un arreglo estático

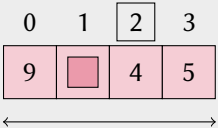
Eliminar un dato de una cola doble del lado ante

```
int delanteAE(cdoble *s, int *x) {
  if (s->n == 0)        // si la cola esta vacia
    return 0;           // no se pudo eliminar
  *x = s->a[s->ante]; // toma x de delante de s
  s->ante++;            // mueve ante hacia atras
  if (s->ante == s->max)
    s->ante = 0;        // arregla ante si rebasa
  s->n--;               // un elemento menos en s
  return 1;             // si se pudo eliminar
}
```

Cola doble en un arreglo dinámico

Cola doble que aumenta de tamaño

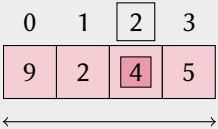
Cola doble casi llena



Cola doble de tamaño 4

Se llena si hacemos

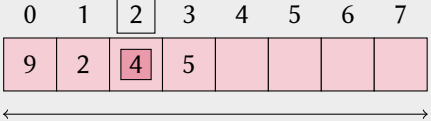
```
atrasAD(&s, 2);
```



Cola doble de tamaño 4

¿Qué pasa si hacemos crecer el arreglo?

Cuando crecimos el arreglo en una pila, los datos que ya estaban en la pila siguen en el lugar correcto. Al crecer el arreglo en una cola doble, los datos que ya estaban dejan de formar un arreglo circular.



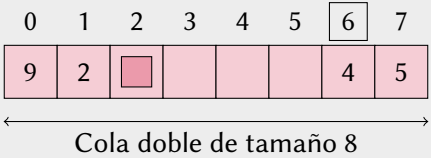
Cola doble de tamaño 8

Si hacemos `adelanteAD(&s, 6);` primero crece la cola

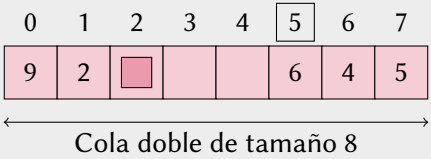
Cola doble en un arreglo dinámico

Primera forma de recuperar el arreglo circular

Mover `ante` y los datos `a[ante..max-1]` a `a[max+ante..2*max-1]`.



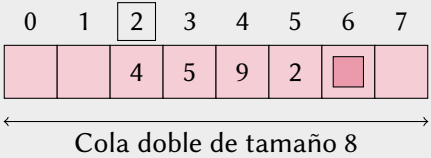
Y completar la operación `adelanteAD(&s, 6);`



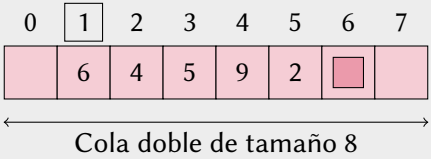
Cola doble en un arreglo dinámico

Segunda forma de recuperar el arreglo circular

Mover `tras` y los datos `a[0..tras-1]` a `a[max..max+tras-1]`.



Y completar la operación `adelanteAD(&s, 6);`



Cola doble en un arreglo dinámico

Implementación de la primera forma de crecer una cola doble

```
int creceCdoble(cdoble *s) {
  int *q = (int *) realloc(s->a, 2*(s->max)*sizeof(int));
  if (q == NULL)                    // si falla duplicar a[]
    return 0;                       // no se pudo crecer a[]
  s->a = q;                         // nueva ubicacion de a[]
  for (int i = s->ante; i < s->max; i++)
    s->a[s->max+i] = s->a[i];       // mueve una parte
  s->ante += s->max;                // nueva posicion de ante
  s->max = 2*(s->max);              // nuevo tamano de a[]
  return 1;                         // si se pudo crecer a[]
}
```

Cola doble en un arreglo dinámico

Modificación para agregar un dato

Tanto en atrasAE como en adelanteAE cambia

```
if (s->n == s->max) // si la cola esta llena
  return 0;         // no se pudo agregar x
```

por lo siguiente en atrasAD y en adelanteAD:

```
if (s->n == s->max)     // si la cola esta llena
  if (!creceCdoble(s)) // y no crecio al doble
    return 0;           // no se pudo agregar x
```

Nota que como la cola doble *s se mandó por referencia (¿todavía recuerdas que &*s = s?) entonces creceCdoble hizo su trabajo.

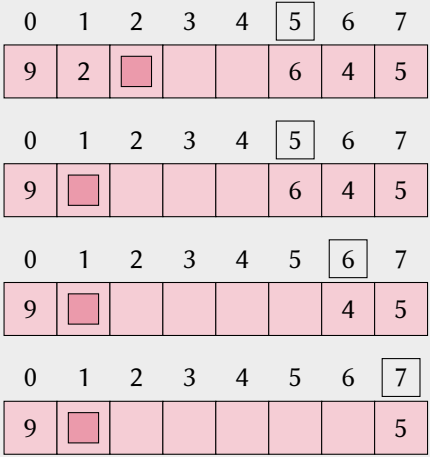
Cola doble en un arreglo dinámico

Ejercicios

1. Implementa la segunda forma de recuperar un arreglo circular.
2. Combina las dos implementaciones eligiendo una de las dos formas según cuál mueva menos datos. Nota que la primera forma mueve `n-ante` datos mientras que la segunda forma mueve `tras` datos.
3. ¿Cuál es la máxima cantidad de datos que se mueven con esta combinación?
4. Implementa una función `invierte(a,i,j)` que invierta el contenido del subarreglo `a[i..j]`, es decir, intercambia `a[i]` con `a[j]`, `a[i+1]` con `a[j-1]`, etc.
5. Hay una tercera forma de recuperar un arreglo circular en `a[0..n-1]` usando tres llamadas a `invierte`. ¿Cómo?

Cola doble en un arreglo dinámico

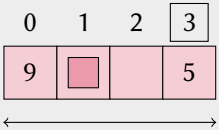
Colas dobles que disminuyen de tamaño



Cola doble muy vacía

Si eliminamos muchos datos, ahora se usa apenas un cuarto de la capacidad de la cola doble. Igual que con las pilas, bajaremos el tamaño a la mitad, pero antes debemos reacomodar los datos.

Cola doble más corta



Cola doble de tamaño 4

Cola doble en un arreglo dinámico

Reacomodar los datos al reducir el tamaño

El nuevo tamaño de la cola doble será `nmax = max/2`. Como queremos mantener un arreglo circular después de cambiar el tamaño, resulta que hay cuatro casos de acuerdo a los valores relativos de `ante`, `tras`, `nmax` y `max`:

1. Si `ante < tras < nmax` entonces no se necesita reacomodar nada.
2. Si `ante < nmax <= tras` entonces se necesita reacomodar el lado `tras` de la cola doble. Esto mueve `tras - nmax` datos.
3. Si `nmax <= ante <= tras < max` entonces se necesita reacomodar toda la cola doble. Esto mueve `n = tras - ante` datos.
4. Si `tras < nmax < ante < max` entonces se necesita reacomodar el lado `ante` de la cola doble. Esto mueve `max - ante` datos.

En el peor de los casos se deben mover todos los datos.

Dos representaciones de colas dobles

Resumen de resultados

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

Operación	Arreglo estático	Arreglo dinámico
crear	1	1
destruir	1	1
vacía	1	1
adelante/atrás	1	$\frac{3}{2}n + 1$
delante/detrás	1	$n + 1$

Los $n + 1$ y $\frac{3}{2}n + 1$ en la cola doble en arreglo dinámico se deben a que a veces se copian los n datos que ya estaban y se arregla la mitad. Sin embargo, se puede demostrar que en promedio se copian menos de 3 datos por operación.