Array dinamici

Array la cui dimensione cambia in base alla necessità. Se al momento dell'inserzione di un elemento non c'è abbastanza spazio allora creiamo un nuovo array con lunghezza doppia di quella corrente; se rimuovendo un elemento scendiamo sotto una determinata soglia allora dimezziamo la lunghezza dell'array. Dato un oggetto A con proprietà *size*, *num*, *arr*, definiamo le seguenti procedure per inserire e cancellare un elemento alla fine dell'array:

INSERT(A, x)

```
1  if A.size == 0
2      allocate A.arr with 1 slot
3      A.size = 1
4  if A.num == A.size
5      allocate new-arr with 2 · A.size slots
6      for i = 1 to A.num
7          new-arr[i] = A.arr[i]
8      free A.arr
9      A.arr = new-arr
10     A.size = 2 · A.size
11 A.num = A.num + 1
12 A.arr[A.num] = x
```

DELETE(A)

```
1  if A.num == 0
2      return
3  if A.num == ⌊A.size/4⌋
4      allocate new-arr with ⌊A.size/2⌋ slots
5      for i = 1 to A.num
6          new-arr[i] = A.arr[i]
7      free A.arr
8      A.arr = new-arr
9      A.size = ⌊A.size/2⌋
10 A.num = A.num − 1
```

La complessità al caso pessimo è $\mathrm{O}(n)$, ma se non c'è bisogno di ridimensionare l'array sono entrambe $\mathrm{O}(1)$. Una volta modificata la lunghezza servono *almeno* $n/2$ inserzioni o $n/4$ rimozioni perché avvenga un nuovo ridimensionamento, quindi il costo ammortizzato è $\frac{\mathrm{O}(n)}{\Omega(n)} = \mathrm{O}(1)$.