Stack

Lo stack (o pila) è una struttura che implementa la strategia LIFO (last in, first out).

→ viene usato per esempio per gestire le ricorsioni (si parla infatti di stack delle chiamate ricorsive)

Ogni stack implementa due metodi (maggiori con nomi diversi o qualche diff.):

- push(x) → aggiungi in cima alla pila
- pop() → toglie e restituisce l'elem. in cima alla pila.

Coda

A diff. dello stack, implemento la FIFO (first in, first out)

Implemento due metodi:

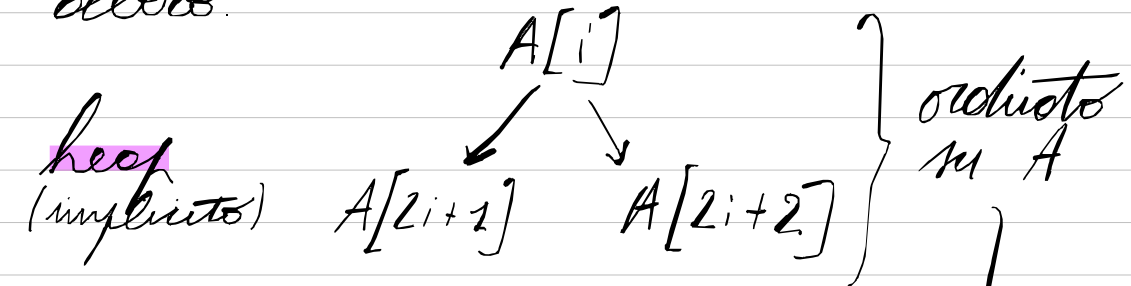
- enqueue(x) → aggiunge x in fondo alla coda
- dequeue() → toglie e restituisce il primo della coda

→ le implementa poss. usano array, liste o vector.

Code di priorità (priority queue)

→ ogni elemento ha una sua priorità e si decide come trovare un elemento in base a tale priorità (esempio: accesso con i codici di pronto soccorso).

Per creare una tale struttura si usano spesso degli array tenendo a mente una struttura di albero:



heap (implicito)

quindi $padre(i) = \left\lfloor \frac{i-1}{2} \right\rfloor$.

- se la priorità è crescente, heap-max, altr. heap-min

Vale padre > figli o < in base a queste esigenze

→ nell'heap imp. l'altezza è $O(\log(n))$ poiché padre $(i) = \left\lfloor \frac{i-1}{2} \right\rfloor \leq \frac{i}{2} \Longrightarrow$ a partire dalla foglia $n-1$ verso la radice "si dimezza ogni volta".

→ in `enqueue(x)` l'idea è quella di aggiungere x come ultima foglia possibile e poi riorganizzare l'heap (o meglio, per mantenere le proprietà dell'heap).

- si fa solire la foglia finché che supera i padri (`sift-up`)

→ in `dequeue()` si restituisce la radice, si mette l'ultima foglia nella radice e la si fa scendere scambiandola ogni volta con il figlio con più priorità (`sift-down`).

Entrambi gli algoritmi eseguono al più un numero di passi pari all'altezza, e sono dunque $O(\log(n))$.

→ basato sulla struttura dell'heap si può creare un alg. di ordinam. per conf. chiamato heapsort (si aggiungono i valori nell'heap e poi si estrae finché si può):

- $n \cdot O(\log(n)) = O(n \log n)$ di tempo
- $O(1)$ di spazio aggiuntivo

meglio del merge sort

Nella pratica è però più lento del quick sort.