Azzolini Riccardo 2019-03-11

Pila e coda

1 Liste ad accesso ristretto

Le **liste ad accesso ristretto** sono strutture dati dinamiche elementari. Rispetto a una comune lista, esse hanno solo operazioni che agiscono su posizioni specifiche. Le principali sono la *pila* e la *coda*.

2 Pila

La **pila** è una struttura **LIFO**, con le operazioni:

$$\textsc{Is\_empty}(S) = \begin{cases} 1 & \text{se } S = \Lambda \\ 0 & \text{altrimenti} \end{cases}$$

$$\textsc{Top}(S) = \begin{cases} a_n & \text{se } S = (a_1, \ldots, a_n) \\ \bot & \text{se } S = \Lambda \end{cases}$$

$$\textsc{Pop}(S) = \begin{cases} (a_1, \ldots, a_{n-1}) & \text{se } S = (a_1, \ldots, a_n) \\ \bot & \text{se } S = \Lambda \end{cases}$$

$$\textsc{Push}(S,\, a) = \begin{cases} (a_1, \ldots, a_n, a) & \text{se } S = (a_1, \ldots, a_n) \\ (a) & \text{se } S = \Lambda \end{cases}$$

Osservazione: Queste operazioni possono essere definite equivalentemente sul primo elemento della lista, anziché sull'ultimo.

2.1 Implementazione con una lista concatenata

Procedura Is_empty(S)
  if $S = nil$ **then return** `Vero`
    else return `Falso`

Procedura Top(S)

```
    return (*S).elem

Procedura POP(S)
    if IS_EMPTY(S) then
        return −1;
    else
        begin
            S := (*S).succ;
            return 0;
        end

Procedura PUSH(S, a)
    begin
        X := CREA_NODO(a);
        (*X).succ := S;
        S := X;
    end
```

Tutte queste operazioni hanno costo costante, cioè $O(1) = \Theta(1)$.

3 Coda

La **coda** è una struttura **FIFO**, sulla quale sono definite le operazioni:

$$\textsc{Is\_empty}(Q) = \begin{cases} 1 & \text{se } Q = \Lambda \\ 0 & \text{altrimenti} \end{cases}$$

$$\textsc{Top}(Q) = \begin{cases} a_1 & \text{se } Q = (a_1, \ldots, a_n) \\ \bot & \text{se } Q = \Lambda \end{cases}$$

$$\textsc{Dequeue}(Q) = \begin{cases} \Lambda & \text{se } Q = (a_1) \\ (a_2, \ldots, a_n) & \text{se } Q = (a_1, \ldots, a_n) \\ \bot & \text{se } Q = \Lambda \end{cases}$$

$$\textsc{Enqueue}(Q, a) = \begin{cases} (a_1, \ldots, a_n, a) & \text{se } Q = (a_1, \ldots, a_n) \\ (a) & \text{se } Q = \Lambda \end{cases}$$

3.1 Implementazione con una lista concatenata

Una coda può essere implementata con una lista concatenata, in modo simile a una pila.

Per poter effettuare l'operazione ENQUEUE in tempo costante, senza dover scorrere l'intera lista, è necessario un puntatore anche all'ultimo nodo.

3.2 Implementazione con un vettore

Se è nota la dimensione massima della coda, si può utilizzare un'implementazione basata su un vettore gestito in modo circolare, sulla quale le operazioni hanno uguale complessità asintotica ma fattori costanti minori.

Dato un vettore di n elementi, per utilizzarlo come coda è necessario tenere traccia di due indici:

- p: l'indice del primo elemento;
- u: l'indice in cui verrà inserito il prossimo elemento.

L'operazione ENQUEUE inserisce quindi un elemento in posizione u e incrementa questa variabile, assegnando come nuovo valore $u + 1 \mod n$.

L'operazione DEQUEUE, invece, incrementa allo stesso modo la variabile p, rimuovendo effettivamente il primo elemento.

Se $p = u$, la coda può essere piena o vuota: per distinguere tra queste due situazioni è necessaria un'apposita variabile booleana.