Esercizi – Strutture Dati

Anno Accademico 2022/2023

Dott. Staccone Simone

TOR VERGATA
UNIVERSITÀ DEGLI STUDI DI ROMA

Esercizi Liste

1. Si scriva una procedura che, data una lista L di interi, restituisca un'altra lista M che contenga solo gli elementi di L i cui valori compaiono esattamente due volte.

2. Si scriva una procedura iterativa/ricorsiva che data una lisa L di interi, la modifichi eliminando ogni elemento pari, sostituendolo con il primo elemento dispari che successivo nella lista. (Se non ci sono elementi successivi lo si inizializzi a 1). (Hint: la lista è doppiamente collegata?)

3. Si scriva una versione in C, delle strutture queue e stack. (Vedi codice per stack)

Esercizi Liste

1. Si scriva una procedura che, data una lista L di interi, restituisca un'altra lista M che contenga solo gli elementi di L i cui valori compaiono esattamente due volte.

```
fun countElements(list L) : list
    newList ← NewList(L)  //Creo una nuova lista esattamente uguale ad L
    curr ← newList
    while(curr.next != NULL):
        new_curr ← curr
        count ← 1
        while(new_curr.next != NULL):
            if(new_curr.value == curr.value)
                    count++;
                    deleteNode(newList,new_curr) //Elimino i nodi della seconda lista per
non contare più volte lo stesso valore
        if(count == 2)
            addNode(M,curr.value) //Aggiungo ad M il valore contato esattamente 2 volte
        curr ← curr.next
    return M
```

Esercizi Liste

1. Si scriva una procedura che, data una lista L di interi, restituisca un'altra lista M che contenga solo gli elementi di L i cui valori compaiono esattamente due volte.

```
fun countElements(list L) : list
    newList ← NewList(L)
    curr ← newList
    while(curr.next != NULL):
        new_curr ← curr
        count ← 1
        while(new_curr.next != NULL):
            if(new_curr.value == curr.value)
                    count++;
                    deleteNode(newList,new_curr)
        if(count == 2)
            addNode(M,curr.value)
        curr ← curr.next
    return M
```

Complessità?

Esercizi Liste

1. Si scriva una procedura che, data una lista L di interi, restituisca un'altra lista M che contenga solo gli elementi di L i cui valori compaiono esattamente due volte.

```
fun countElements(list L) : list
    newList ← NewList(L)
    curr ← newList
    while(curr.next != NULL):
        new_curr ← curr
        count ← 1
        while(new_curr.next != NULL):
            if(new_curr.value == curr.value)
                    count++;
                    deleteNode(newList,new_curr)
        if(count == 2)
            addNode(M,curr.value)
        curr ← curr.next
    return M
```


Complessità? O(n^2) Dove n è il numero degli elementi della lista L. (Anche O(n) in spazio per l'istanziazione della nuova lista, che serve se non si vuole modificare L)

Esercizi Liste - ottimizzazione

1. Si scriva una procedura che, data una lista L di interi, restituisca un'altra lista M che contenga solo gli elementi di L i cui valori compaiono esattamente due volte.

L'idea è quella di utilizzare una struttura dati di appoggio che contiene il numero di volte in cui compare un valore all'interno della lista L.

Es.
L = [1,3,4,1,4,5]
M = {1:2,3:1,4:2,5:1}

Due problematiche: se implemento questo dizionario devo usare una tabella di hash/una lista con coppia chiave-valore → O(n) ma con utilizzo di spazio

Un altro approccio è quello di ordinare il vettore prima e poi inserire nella lista M i valori di L in ordine → O(n*log n) ? Dipende dall'algoritmo d'ordinamento usato e dall'istanza in esame

Esercizi Liste

2. Si scriva una procedura iterativa/ricorsiva che data una lisa L di interi, la modifichi eliminando ogni elemento pari, sostituendolo con il primo elemento dispari che successivo nella lista. (Se non ci sono elementi successivi lo si inizializzi a 1). (Hint: la lista è doppiamente collegata?)

```
struct list{
    head : node
}

struct node{
   value : int
   pred : node //NULL se primo elemento
   next : node //NULL se ultimo elemento
}
```

Esercizi Liste - ricorsivo

2. Si scriva una procedura iterativa/ricorsiva che data una lisa L di interi, la modifichi eliminando ogni elemento pari, sostituendolo con il primo elemento dispari che successivo nella lista. (Se non ci sono elementi successivi lo si inizializzi a 1). (Hint: la lista è doppiamente collegata?)

```
fun replaceEven(list L) :
    curr ← L.head
    count ← 0
    isOdd(curr , count, L)
```

Esercizi Liste - ricorsivo

```
fun isOdd(node n, int count, list L) :
    if(n == NULL):
        return
    else:
        if(n.value%2 == 0):
            count ← count + 1
            if(n.pred != NULL):
                n.pred.next ← n.next          // Delate
                n.next.pred ← n.pred          // node
            else:
                L.head ← n.next //Necessario per i nodi pari in testa, si può evitare?
            isOdd(n.next, count, L)
        else:
            if(count > 0):
                addOdd(n)  //Aggiunge un nodo in testa ad n
                count ← count – 1
            else:
                isOdd(n.next, count, L)
```

Entro in isOdd() o quando il numero è pari per scorrere la lista, o quando il contatore è 0, ovvero quando ho finito di aggiungere nodi in testa

Esercizi Liste - ricorsivo

2. Si scriva una procedura iterativa/ricorsiva che data una lisa L di interi, la modifichi eliminando ogni elemento pari, sostituendolo con il primo elemento dispari che successivo nella lista. (Se non ci sono elementi successivi lo si inizializzi a 1). (Hint: la lista è doppiamente collegata?)

```
fun addOdd(node n):
    new ← Node()
    new.value ← n.value
    new.next = n
    new.pred = n.pred
    if(n.pred != NULL):  //Se ci troviamo all'inizio della lista
        n.pred.next = new
        n.pred = new
```