Algoritmi per liste

CORDA – Informatica

A. Ferrari

Testi da

Marco Bernardo Edoardo Bontà

Dispense del Corso di

Algoritmi e Strutture Dati

Lista - definizione

- Si dice lista una tripla $L = (E, t, \mathcal{S})$ dove E è un insieme di elementi, $t \in E$ è detto testa ed $\mathcal{S}$ è una relazione binaria su E – cioè $\mathcal{S} \subseteq E \times E$ – che soddisfa le seguenti proprietà:
  - Per ogni $e \in E$, $(e, t) \notin \mathcal{S}$.
  - Per ogni $e \in E$, se $e \neq t$ allora esiste uno ed un solo $e' \in E$ tale che $(e', e) \in \mathcal{S}$.
  - Per ogni $e \in E$, esiste al più un $e' \in E$ tale che $(e, e') \in \mathcal{S}$.
  - Per ogni $e \in E$, se $e \neq t$ allora e è raggiungibile da t, cioè esistono $e'_1, \ldots, e'_k \in E$ con $k \geq 2$ tali che $e'_1 = t$, $(e'_i, e'_{i+1}) \in \mathcal{S}$ per ogni $1 \leq i \leq k-1$, ed $e'_k = e$.

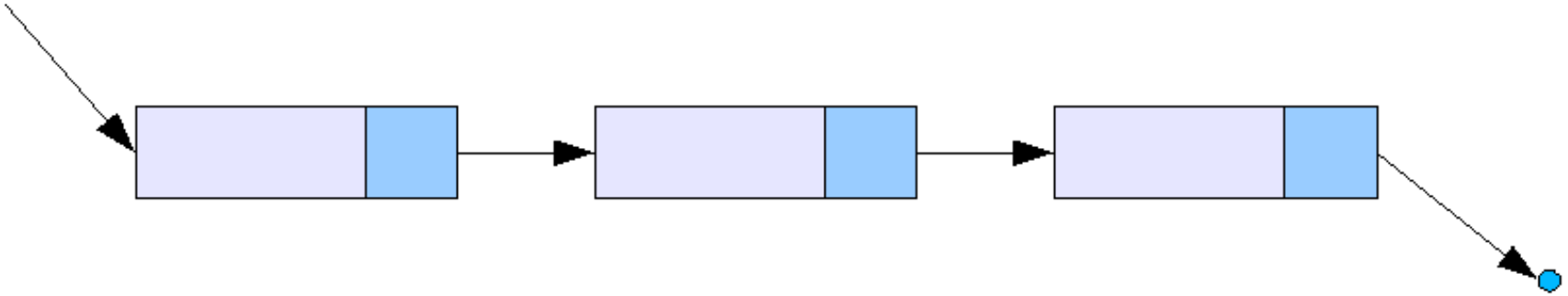
Liste ordinate

Una lista $L = (E, t, \mathcal{S})$ è detta ordinata se le chiavi contenute nei suoi elementi sono disposte in modo tale da soddisfare una relazione d'ordine totale: per ogni $e_1, e_2 \in E$, se $(e_1, e_2) \in \mathcal{S}$ allora la chiave di e_1 precede la chiave di e_2 nella relazione d'ordine totale.

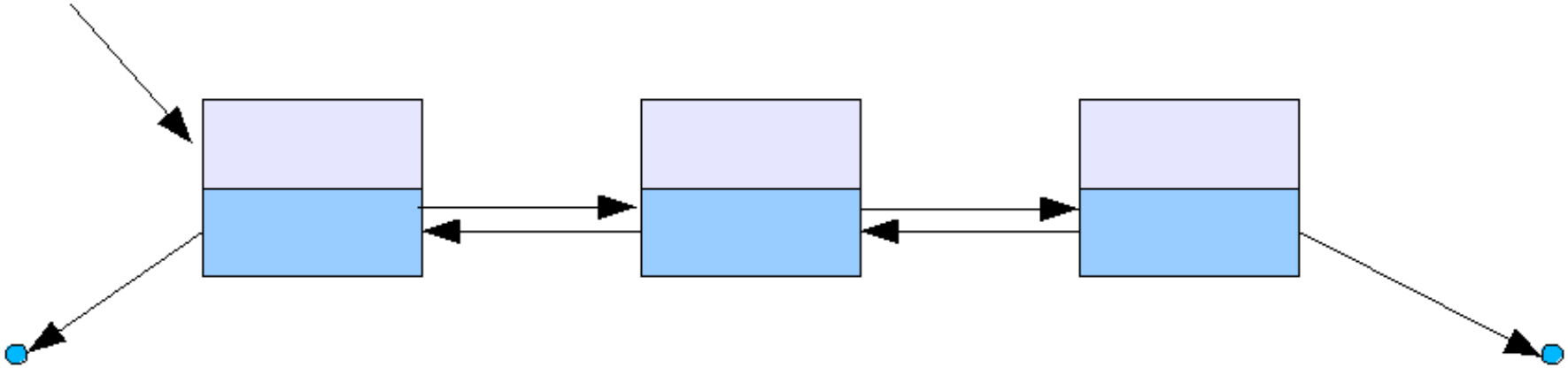
Liste - rappresentazione

- Una lista viene rappresentata come una struttura dati dinamica lineare, in cui ogni elemento contiene solo il riferimento all'elemento successivo (lista singolarmente collegata) oppure anche il riferimento all'elemento precedente (lista doppiamente collegata)

Lista semplice



Lista bidirezionale



Lista - caratteristiche

- L'indirizzo dell'elemento successivo contenuto nell'ultimo elemento di una lista è indefinito, così come l'indirizzo dell'elemento precedente contenuto nel primo elemento di una lista doppiamente collegata. Fa eccezione il caso dell'implementazione circolare di una lista, nella quale l'ultimo elemento è collegato al primo elemento.
- Gli elementi di una lista non sono necessariamente memorizzati in modo consecutivo, quindi l'accesso ad un qualsiasi elemento avviene scorrendo tutti gli elementi che lo precedono. Questo accesso indiretto necessita di un riferimento al primo elemento della lista, detto testa, il quale è indefinito se e solo se la lista è vuota.

Esercitazione 1

- Definizione di una lista in linguaggio C
- Definizione della struttura Nodo e della struttura Lista
- Nella struttura Nodo l'informazione è di tipo int
- Per la struttura Lista definire le funzioni
  - insTesta, insCoda, eliTesta, …

Liste in C - definizione

```
struct Nodo{
 int key;   // informazione associata al Nodo
 struct Nodo* next; // puntatore al prox. nodo
};
// Una lista è rappresentata da un puntatore
// al suo nodo di testa (o NULL se vuota)
typedef Nodo* Lista;
```

Liste in C - utilizzo

```
// Creazione di una lista di due nodi
struct Nodo* e1 = malloc(sizeof(Nodo));
struct Nodo* e2 = malloc(sizeof(Nodo));
Lista lis;
e1->key = 33; e1->next = e2;
e2->key = 12; e2->next = NULL;
lis = e1;
// Inserimento in testa di un nodo con valore 4
Struct Nodo* e3 = malloc(sizeof(Nodo));
e3->key = 4; e3->next = lis;
lis = e3;
```

Esercizi su liste

- // visualizzare le chiavi di tutti i nodi di lis
  - void print(Lista lis);
- // inserire l'elemento e in coda o in testa a lis
- // e restituire la nuova lista
  - Lista insertBack(Lista lis, Nodo *e);
  - Lista insertFront(Lista lis, Nodo *e);
- // liberare lo spazio occupato da tutti i nodi di lis
  - void deallocate(Lista lis);

Problemi sulle liste

- Problema della visita:
  - data una lista, attraversare tutti i suoi elementi esattamente una volta.
- Problema della ricerca:
  - dati una lista e un valore, stabilire se il valore è contenuto in un elemento della lista, riportando in caso affermativo l'indirizzo di tale elemento.
- Problema dell'inserimento:
  - dati una lista e un valore, inserire (se possibile) nella posizione appropriata della lista un nuovo elemento in cui memorizzare il valore.
- Problema della rimozione:
  - dati una lista e un valore, rimuovere (se esiste) l'elemento appropriato della lista che contiene il valore.

Esercitazione 2

- Utilizzando le strutture dati viste precedentemente implementare le funzioni che permettono di risolvere i problemi di:
  - visita
  - ricerca
  - inserimento
  - rimozione