Linguaggi I2

Serie in classe: memoria dinamica

1. **Lista lineare contenente degli interi:**
   a) Scrivere un programma che legge una sequenza di interi da tastiera (termina quando viene inserito un valore sentinella pre-impostato, es. 0)
   b) Creare una lista lineare che li contenga
   c) Ordinare la lista - dalla testa alla coda - nello stesso modo con cui i numeri sono stati inseriti, e visualizzarla
   d) Cancellare tutti gli elementi pari dalla lista (2, 4, 6, ecc), e visualizzarla

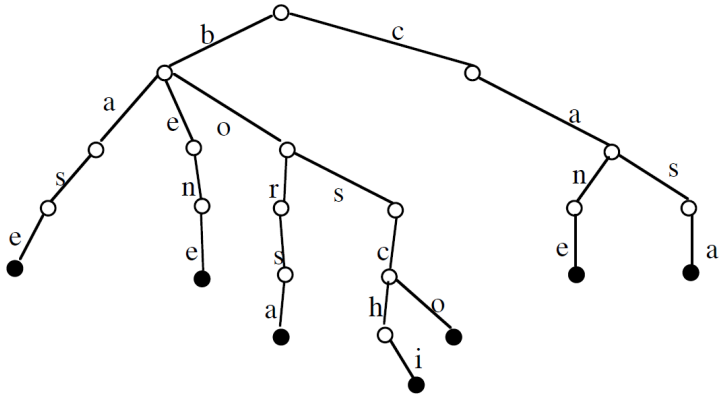
2. **Lista lineare contenente testo e numeri, gestita con puntatori:**
   a) Scrivere un programma che apre e legge un file di testo. Il nome del file viene passato come argomento da linea di comando
   b) Definiamo con il termine “pseudo-words” delle stringhe separate da uno o più spazi vuoti. Ogni “pseudo-word” va inserita in una lista lineare solo se non è già presente nella stessa. Per semplificazione, nel caso in cui la parola è già presente, essa viene considerata come “non esistente”, e non viene quindi processata
   c) Creare una lista lineare i quali nodi contengano due tipi di campo, un puntatore al “word” e un puntatore al prossimo nodo. L’ordine della lista deve corrispondere esattamente all’ordine delle parole nel file. Visualizzare la lista
   d) Ordinare la lista in ordine alfabetico, prendendo come riferimento ll primo carattere della “pseudo-word”, e mettendo le cifre prima delle lettere a-z. Visualizzare la lista

3. Trie con parole:

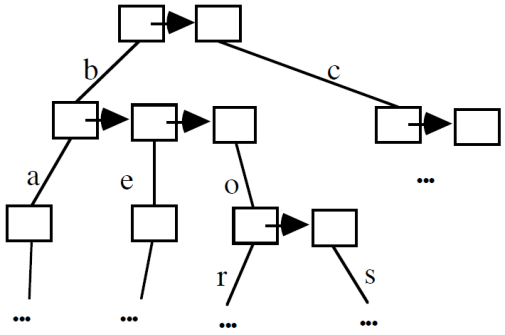
a) Scrivere un programma che apre e legge un file di testo. Il nome del file viene passato come argomento da linea di comando

b) Definiamo con il termine “pseudo-words” delle stringhe separate da uno o più spazi vuoti. Ogni “pseudo-word” va inserita nel trie solo se non è già presente

c) Considerare l’esempio di “Trie” indicato nello script:



Ogni singolo nodo deve contenere un solo puntatore a una lista dinamica di tutti i suoi possibili figli



d) Implementare le funzioni di inserimento, cancellazione e ricerca

e) Visualizzare la lista in modo appropriato

4. **Closed hashing:**

Date delle stringhe di 20 caratteri (es. dei cognomi di persona), da utilizzare come chiavi “key”, e dei dati utili associati ai cognomi “data” di tipo int (es. il valore dello stipendio),

a) creare una struttura apposita per ospitare key e data

b) leggere i dati key e data da un file di testo, contenente al massimo 1000 dati

c) creare una funzione di hash che prende il valore ASCII del primo carattere e lo moltiplica per 101, lo somma al valore ASCII dell’ultimo carattere moltiplicato per 103, e lo somma alla lunghezza della stringa moltiplicato per 107. Al risultato applicare il modulo di 1000

Es. hash(“ABC”)=(65*101+67*103+3*107)%1000=787

d) Considerare le collisioni, per esempio il caso in cui

hash(“ABC”)= hash(“ARC”)=787 e gestirlo secondo il principio del closed hashing

e) inserire i dati letti da file in una tabella apposita, verificando che se una key è presente nel file due volte non va inserita (dare un messaggio di errore), idem se la tabella è piena

d) richiedere all’utente di inserire una key, trovare il campo corrispondente e visualizzare il dato corrispondente