Università del Piemonte Orientale

Dipartimento di Scienze e Innovazione Tecnologica

Simulazione d'Esame di Algoritmi 1 – Sperimentazioni (VC)

24 gennaio 2025

Testo d'Esame

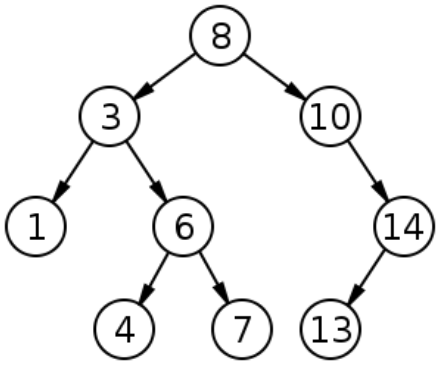

Figura 1: Un esempio di BST.

Esercizio 1 (max 15 punti)

Implementare un algoritmo che, dato un albero binario di ricerca (BST) e due chiavi k_1 e k_2, restituisca la chiave del loro **minimo comune antenato (Lowest Common Ancestor - LCA)**. L'LCA di due nodi n_1 e n_2 è il nodo più lontano dalla radice del BST che ha sia n_1 che n_2 come discendenti (N.B. un nodo può essere discendente di se stesso). Per esempio, dato l'albero di Figura 1, si ha che l'LCA di 1 e 7 è 3, l'LCA di 4 e 10 è 8, l'LCA di 10 e 13 è 10 e l'LCA di 6 e 18 è *NULL*.

L'algoritmo implementato dev'essere ottimo: 1) deve visitare l'albero una sola volta, 2) non deve visitare parti del BST inutili ai fini dell'esercizio e 3) la complessità temporale nel caso peggiore dev'essere $O(n)$, dove n è il numero di chiavi nel BST.

* * *

La funzione da implementare si trova nel file `exam.c` e ha il seguente prototipo:

```
void *upo_bst_lca(const upo_bst_t tree, const void *key1, const void *key2)
```

Parametri:

- `tree`: BST.
- `key1` e `key2`: puntatori alla prima e seconda chiave di cui si vuole trovare il minimo comune antenato.

Valore di ritorno:

- Se il BST non è vuoto e le chiavi `key1` e `key2` sono contenute nel BST: il puntatore alla chiave del loro LCA.
- Se il BST è vuoto o le chiavi `key1` e/o `key2` non sono contenute nel BST: *NULL*.

Il tipo `upo_bst_t` è dichiarato in `include/upo/bst.h`. Per confrontare il valore di due chiavi (qualora fosse necessario) si utilizzi la funzione di comparazione memorizzata nel campo `key_cmp` del tipo `upo_bst_t`.

Nella propria implementazione è possibile utilizzare tutte le funzioni dichiarate in `include/upo/bst.h`. Nel caso s'implementino nuove funzioni, i prototipi e le definizioni devono essere inserite nel file `exam.c`.

Il file `test/bst_common_ancestor.c` contiene alcuni casi di test tramite cui è possibile verificare la correttezza della propria implementazione. Per compilarlo con la propria implementazione, è sufficiente eseguire il comando: `make clean all`

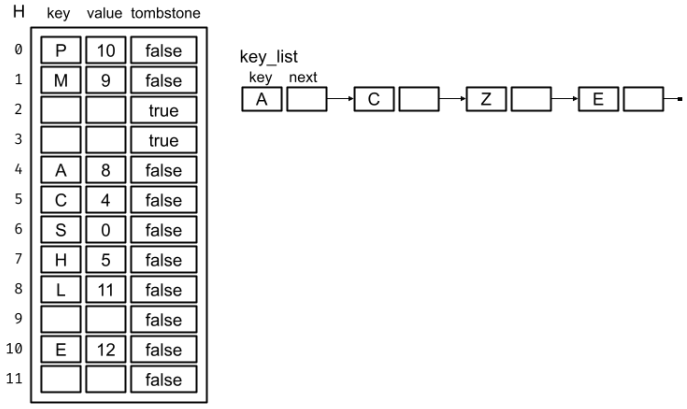

Figura 2: Un esempio di HT-LP.

Esercizio 2 (max 15 punti)

Implementare un algoritmo che, data una tabella hash H, con gestione delle collisioni basata su indirizzamento aperto (i.e. open addressing) e scansione lineare (i.e. linear probing) con uso di *tombstone* (HT-LP), e una lista di chiavi l_{keys}, calcoli la media del numero di collisioni delle chiavi k contenute in l_{keys} in H. In particolare:

- Se una chiave k in l_{keys} non è contenuta in H, non dev'essere considerata nel calcolo della media del numero di collisioni.
- Se H è vuota, o se l_{keys} è vuota, o se nessuna chiave di l_{keys} è contenuta in H, l'algoritmo deve restituire il valore -1.

Si noti che nel calcolare il numero di collisioni di una chiave k non si deve tenere conto dello slot in cui k è memorizzata e che uno slot *tombstone*, se attraversato, è considerato una collisione.

Per esempio, date la tabella e la lista di chiavi in Figura 2, e supponendo che il valore hash delle chiavi A, C, e E sia rispettivamente $0, 0$, e 10, **il numero medio di collisioni è** 3 in quanto:

- il numero di collisioni di A è 4;
- il numero di collisioni di C è 5;
- il numero di collisioni di E è 0;
- Z non è contenuta in H quindi non viene conteggiata nel calcolo della media.

L'algoritmo implementato deve essere ottimo, nel senso che non deve visitare parti di HT-LP inutili ai fini dell'esercizio.

* * *

La funzione da implementare si trova nel file `exam.c` e ha il seguente prototipo:

```
double upo_ht_linprob_avg_collisions(const upo_ht_linprob_t ht, const upo_ht_key_list_t key_list)
```

Parametri:

- `ht`: HT-LP
- `key_list`: lista concatenata di chiavi.

I tipi `upo_ht_linprob_t` e `upo_ht_key_list_t` sono dichiarati in `include/upo/hashtable.h`.

Nella propria implementazione è possibile utilizzare tutte le funzioni dichiarate in `include/upo/hashtable.h`. Nel caso si implementino nuove funzioni, i prototipi e le definizioni devono essere inserite nel file `exam.c`.

Il file `test/ht_linprob_avg_collisions.c` contiene alcuni casi di test tramite cui è possibile verificare la correttezza della propria implementazione. Per compilarlo con la propria implementazione, è sufficiente eseguire il comando:

```
make clean all
```

Istruzioni per la Consegna

- L'unico elaborato da consegnare è il file `exam.c`.
- La consegna avviene tramite il caricamento del file `exam.c` nell'apposito form sul sito D.I.R. indicato dal docente.

Gli elaborati consegnati che non rispettano tutte le suddette istruzioni o che vengono consegnati in ritardo, non saranno soggetti a valutazione.

Comandi utili

- Comando di compilazione tramite GNU GCC:

```
gcc -Wall -Wextra -std=c11 -pedantic -g -I./include -o eseguibile sorgente1.c sorgente2.c ... -L./lib
    -lupoalglib
```

- Comando di compilazione tramite GNU Make:

```
make clean all
```

- Comando di debug tramite GNU GDB:

```
gdb ./eseguibile
```

- Verifica di memory leak e accessi non validi alla memoria tramite Valgrind:

```
valgrind --tool=memcheck --leak-check=full ./eseguibile
```

- Manuale in linea di una funzione standard del `C`:

```
man funzione
```

Regolamento d'Esame

1. Lo studente deve presentarsi all'esame con un documento di riconoscimento valido.
2. Durante la prova d'esame **non è consentito**:
   - uscire dall'aula;
   - comunicare in qualunque modo con altri individui (docente escluso);
   - utilizzare, o avere a portata di utilizzo, dispositivi elettronici che permettano l'accesso a Internet o lo scambio di comunicazioni (ad es., computer, tablet, telefoni cellulari, smartwatch, ...);
   - utilizzare libri, appunti e altro materiale didattico (cartaceo o digitale), ad eccezione del materiale eventualmente fornito dal docente.
3. Durante la prova d'esame **è consentito** tenere una bottiglia di acqua.
4. È necessario consegnare (**anche in caso di ritiro**) tutti i fogli ricevuti, inclusi quelli per la brutta copia, i quali devono essere esplicitamente segnalati come tali (scrivendo "BRUTTA" su ciascuna delle loro facciate), nonchè il testo della prova d'esame.

Qualora lo studente violi una delle suddette condizioni, o sia colto in flagranza durante l'atto di copiare o se ne appuri a posteriori durante la correzione della prova d'esame, il docente ha la facoltà di bocciarlo e di segnalare il fatto agli organi d'Ateneo competenti (come il Consiglio del Corso di Studi), i quali potranno prendere ulteriori provvedimenti. Le stesse regole si applicano anche agli studenti che permettono che la loro prova d'esame venga copiata o che si prestino a svolgere la prova per conto di altri. Inoltre, qualora lo studente consegni la sua prova d'esame priva dei suoi dati identificativi, o la consegni in ritardo, dopo che il docente ha già effettuato il ritiro delle altre, la sua prova non sarà valutata e il suo tentativo d'esame verrà conteggiato al pari di un ritiro.