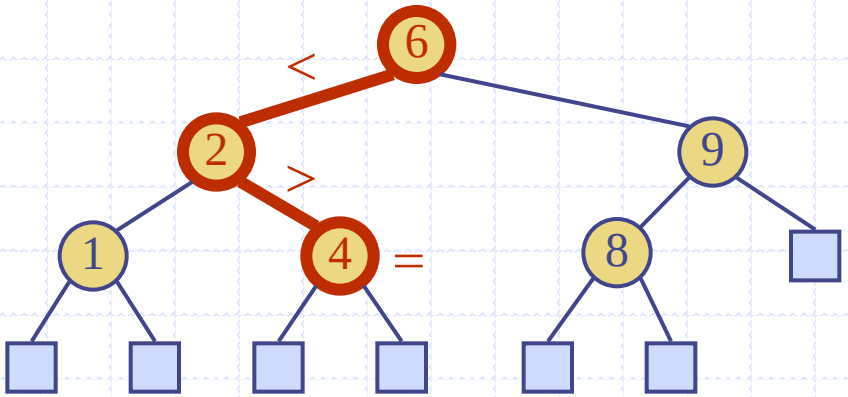
Dizionari



Il TdA Dizionario (9.3)

- Il TdA dizionario modella una collezione di entry chiave-valore su cui è possibile eseguire ricerche
- Le principali operazioni su un dizionario sono ricerca, inserimento e cancellazione di entry
- **Sono** consentite entry multiple con la stessa chiave
- Applicazioni:
  - coppie parola-definizione
  - autorizzazioni per carte di credito
  - trasformazioni DNS da nomi di host (ad es., datastructures.net) ad indirizzi IP su Internet (ad es., 128.148.34.101)

- Metodi del TdA dizionario: :
  - find(*k*): se il dizionario ha una entry con chiave *k* allora la restituisce, altrimenti restituisce null
  - findAll(*k*): restituisce un Iterable per tutte le entry aventi chiave *k*
  - insert(*k*, *o*): inserisce e restituisce la entry (*k*, *o*)
  - remove(*e*): rimuove la entry *e*, restituendola, o restituisce null se non presente
  - entries(): restituisce un insieme iterabile di tutte le entry
  - size(), isEmpty()

Esempio

Operazione	***Output***	***Dizionario***
insert(5,*A*)	(5,*A*)	(5,*A*)
insert(7,*B*)	(7,*B*)	(5,*A*),(7,*B*)
insert(2,*C*)	(2,*C*)	(5,*A*),(7,*B*),(2,*C*)
insert(8,*D*)	(8,*D*)	(5,*A*),(7,*B*),(2,*C*),(8,*D*)
insert(2,*E*)	(2,*E*)	(5,*A*),(7,*B*),(2,*C*),(8,*D*),(2,*E*)
find(7)	(7,*B*)	(5,*A*),(7,*B*),(2,*C*),(8,*D*),(2,*E*)
find(4)	**null**	(5,*A*),(7,*B*),(2,*C*),(8,*D*),(2,*E*)
find(2)	(2,*C*)	(5,*A*),(7,*B*),(2,*C*),(8,*D*),(2,*E*)
findAll(2)	(2,*C*),(2,*E*)	(5,*A*),(7,*B*),(2,*C*),(8,*D*),(2,*E*)
size()	5	(5,*A*),(7,*B*),(2,*C*),(8,*D*),(2,*E*)
remove(find(5))	(5,*A*)	(7,*B*),(2,*C*),(8,*D*),(2,*E*)
find(5)	**null**	(7,*B*),(2,*C*),(8,*D*),(2,*E*)

Un dizionario basato su lista

- Un file di log (o audit trail) è un dizionario implementato tramite una sequenza non ordinata
  - si memorizzano le entry in una sequenza (sfruttando una lista doppiamente collegata o un array), in un ordine arbitrario
- Prestazioni:
  - insert richiede tempo $O(1)$ poiché si può inserire la nuova entry all'inizio o alla fine della sequenza
  - find e remove richiedono tempo $O(n)$ poiché nel caso peggiore (chiave non trovata) è necessario attraversare l'intera sequenza alla ricerca della chiave specificata
- Il file di log è efficace solo per dizionari di piccola dimensione o per dizionari in cui l'operazione più comune è l'inserimento, mentre ricerche ed eliminazioni sono eseguite raramente (ad es., la sequenza storica dei login in una workstation)

L'algoritmo findAll(k)

Dato un dizionario D, basato su una lista non ordinata S

Algorithm findAll(*k*):
Input: Una chiave *k*
Output: Un iteratore per le entry con chiave *k*
Crea una lista vuota L
for each entry *e* in D.entries() **do**
    if *e*.getKey() = *k* **then**
        L.addLast(*e*)
return *L*

Metodi insert e remove

Algorithm insert(*k*,*v*):
Input: Una chiave *k* ed un valore *v*
Output: La entry (*k*,*v*) aggiunta a *D*
Crea una nuova entry *e* = (*k*,*v*)
S.addLast(*e*) {*S* è non ordinata }
return *e*

Algorithm remove(*e*):
Input: Una entry *e*
Output: La entry rimossa *e* o **null** se *e* non è presente in *D*
{ Assumiamo che *e* non contenga la sua locazione in *S* }
for each posizione *p* in S.positions() **do**
    if *p*.element() = *e* **then**
        S.remove(*p*)
        return *e*
return null {non esiste una entry *e* in *D*}

Implementazione basata su tabella hash

- Si possono considerare implementazioni di dizionari basate su tabelle hash
- Gestendo le collisioni con la concatenazione separata, le operazioni di dizionario possono essere delegate ai dizionari basati su lista presenti nelle celle della tabella hash

Ricerca binaria

- La ricerca binaria realizza l'operazione find(k) su un dizionario implementato come sequenza basata su array, ordinata per chiave
  - simile al gioco "alto-basso"
  - ad ogni passo, il numero di candidati viene dimezzato
  - termina perciò dopo un numero logaritmico di passi
- Esempio: find(7)

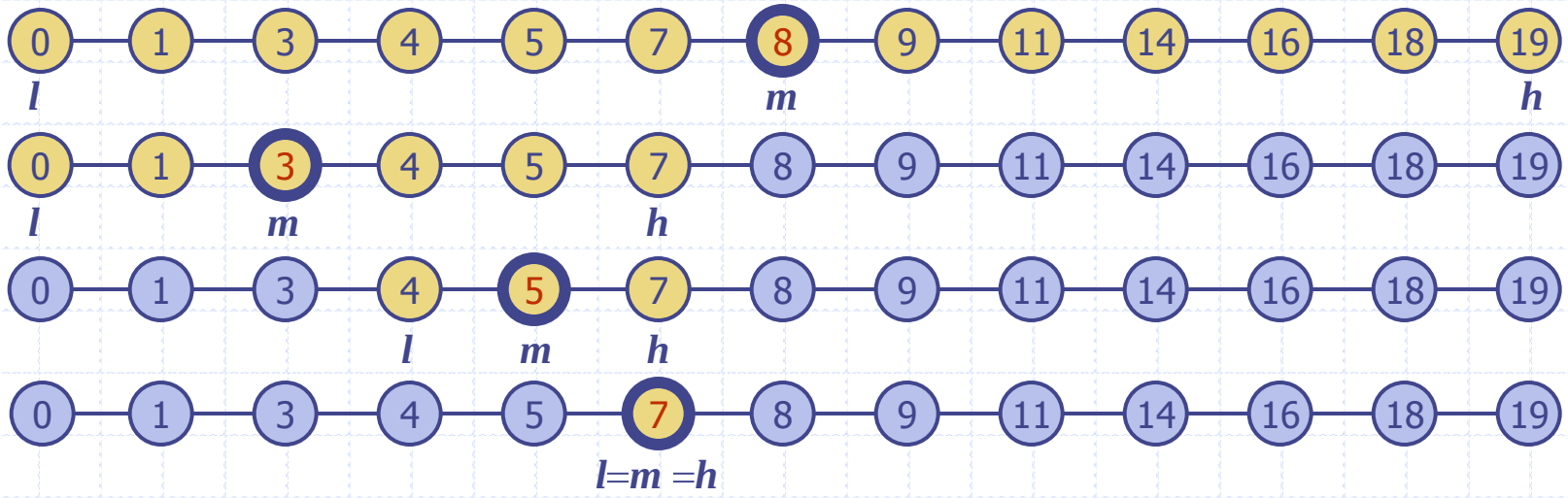


Tabelle di ricerca

- Una tabella di ricerca è un dizionario implementato tramite un array ordinato
  - si memorizzano le entry in una sequenza basata su array, ordinata per chiave
  - si usa apposito oggetto Comparator per le chiavi
- Prestazioni:
  - find richiede tempo $O(\log n)$, usando la ricerca binaria
  - insert richiede tempo $O(n)$ perché nel caso peggiore è necessario spostare n elementi per far spazio al nuovo
  - remove richiede tempo $O(n)$ poiché nel caso peggiore è necessario spostare n elementi per ricompattare l'array dopo l'eliminazione
- Una tabella di ricerca è efficace solo per dizionari di piccole dimensioni o per dizionari dove la ricerca è l'operazione più frequente, mentre inserimenti e rimozioni sono eseguiti raramente (ad es., autorizzazioni per carte di credito)