Mappe ordinate e insiemi

Luca Becchetti

Presentazione tratta dalle slide che accompagnano il testo Data Structures and Algorithms in Java, 6th edition, by M. T. Goodrich, R. Tamassia, and M. H. Goldwasser, Wiley, 2014

Alcuni limiti delle mappe

- Gestione di coppie aventi la medesima chiave
- Elenco delle coppie ordinato in base alla chiave
  - Operazione non ovvia usando ad esempio una tabella hash
  - *Range query*: restituire l'insieme delle coppie aventi chiave $\geq k_1$ e $\leq k_2$
- Rappresentare insiemi
  - Realizzare in modo efficiente operazioni tra insiemi
    - Es.: unione, intersezione, differenza

Mappe ordinate

Range query - esempio

Prezzo

‹ Qualsiasi prezzo
200 - 500 EUR
Più di 500 EUR

EUR 300 | EUR 500 | Vai

Nuovi arrivi

Ultimi 30 giorni
Ultimi 90 giorni

Disponibilità

☐ Includi non disponibili

Sistema operativo

☐ Android

Connettività

☐ 3G
☐ 4G
☐ WiFi

Capacità Hard Disk Notebook

Hp 255 G7 Notebook hp Display da 15.6” Fino A 2.60GHz,Ram 8Gb D
R3,Pc portatile Hp,Hdmi,DVD,Cd RW,Wi fi,Bluetooth,Windows 10 pro

★★★★☆ 700

Ulteriori opzioni di acquisto
428,99 € (3 offerte prodotti nuovi)

Acer Aspire 3 A315-42-R4D6 Notebook con Processore AMD Athlon
PCIe NVMe SSD, Display 15,6" FHD LED LCD, Scheda Grafica AMD Ra

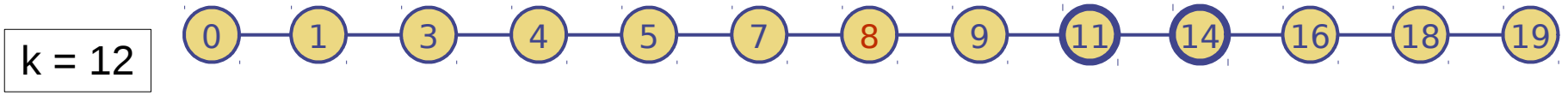
★★★★½ 16

429,00€

✓prime
Spedizione GRATIS da Amazon
Attualmente non disponibile

Mappe ordinate

- Gli oggetti sono ancora una volta coppie (chiave, valore)
- Si suppone che sulle chiavi sia definito un *ordinamento totale*
- Ha senso definire interrogazioni di tipo *nearest neighbour*
  - Dato k: oggetto con chiave massima tra quelle ≤ k
  - Dato k: oggetto con chiave minima tra quelle ≥ k

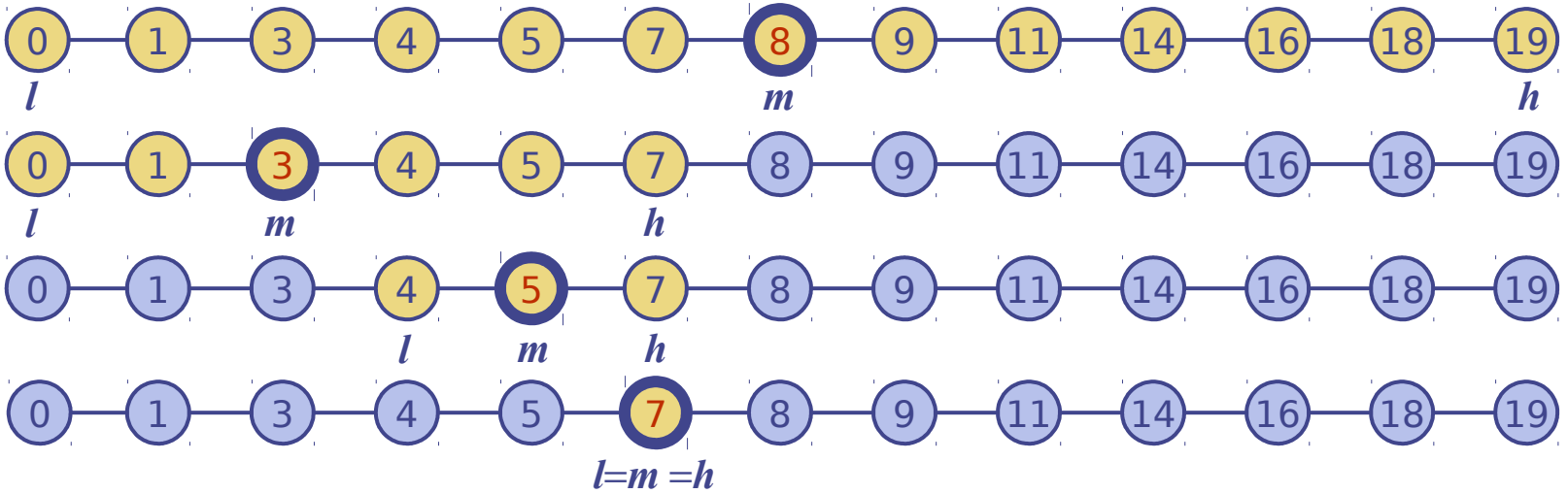


Mappe ordinate: operazioni tipiche

- get(k): restituisce la coppia (le coppie) avente chiave k, se esiste
- put(k, o): inserisce la coppia (k, o)
- remove(k): rimuove la coppia (le coppie) avente chiave k, se esiste
- subMap(k_1, k_2): restituisce tutte le coppie aventi chiave $k_1 \leq k \leq k_2$
  - Tale operazione è tipicamente chiamata *range query*

Tabelle ordinate

- Mappe ordinate realizzate con array ordinato rispetto alle chiavi
  - Nearest neighbour si può effettuare con ricerca binaria
  - Es.: find(7)



Ricerca binaria

```
Algorithm find(key, low, high) {
  if (high < low) return high + 1;          // no entry qualifies
  mid = (low + high) / 2;
  comp = compare(key, table.get(mid));
  if (comp == 0)
     return mid;                            // found exact match
  else if (comp < 0)
     return findIndex(key, low, mid - 1);   // answer is left of mid (or possibly mid)
  else
     return findIndex(key, mid + 1, high);  // answer is right of mid
}

/** Version of findIndex that searches the entire table */
Algorithm find(key) { return find(key, 0, table.size() - 1); }
```

Range query: subMap(k_1, k_2)

1) Trova posizione più a sinistra → i_1 = find(k_1)
   - Se i_1 > size() → k_1 > (massima chiave presente)
   - In tal caso → return
2) Trova la posizione più a destra: i_2 = higherEntry(k_2)
3) Restituisci la porzione della mappa nelle posizioni [i_1...i_2]

```
Algorithm higherEntry(k) {
  j = find(k); // Restituisce posizione ultimo elemento == k o primo > k
  while (j < size() - 1 && k == table[j].getKey())
    j++;
  return j;
}
```

Prestazioni

- get(k): O(log n)
- put(k, o)
  - O(n) se k assente
    - Può essere necessario spostare un numero lineare di coppie
  - O(log n) se k presente
- remove(k): O(n)
  - Implementazione con array → può essere necessario spostare un numero lineare di coppie
- subMap(k_1, k_2): O(log n + s)
  - O(log n) per find(k_1)
  - Determinare la posizione più a destra → complessità lineare nel numero di chiavi comprese tra k_1 e k_2

Insiemi

Insieme

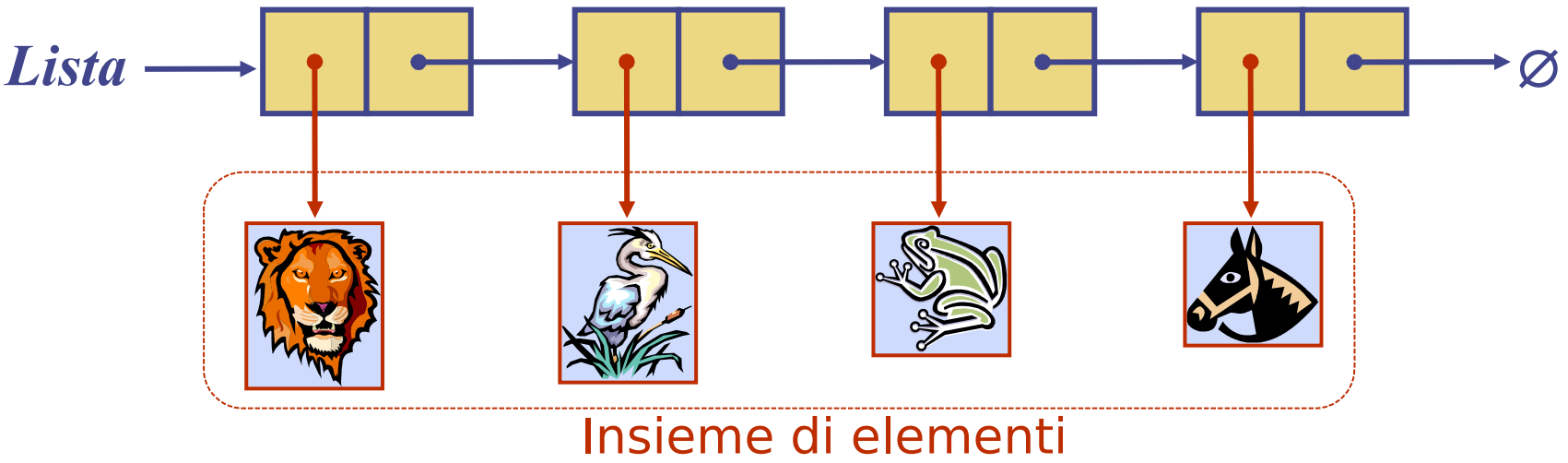
- Collezione di oggetti (priva di duplicati) appartenenti a un certo universo
- Operazioni tipiche tra insiemi
  - Unione $S \cup T = \{ e : e \in S \vee e \in T \}$
  - Intersezione $S \cap T = \{ e : e \in S \wedge e \in T \}$
  - Differenza $S - T = \{ e : e \in S \wedge e \notin T \}$

Insieme ADT

- add(e, S): aggiunge un elemento all'insieme S
- remove(e, S): elimina e da S (se presente)
- contains(e, S): restituisce *true* se e appartiene a S
- union(S, T)
- intersection(S, T)
- difference(S, T)
- Queste e altre operazioni sono implementate in java.util.Set
- Insieme con chiavi non ordinate
  - In effetti si può rappresentare con una mappa le cui chiavi sono gli elementi dell'insieme

Insiemi con chiavi ordinate

- Si può usare una lista
- Spazio O(n)



Operazioni su insiemi ordinati → Merge

- Merge di due liste ordinate A e B
- Complessità $O(n_A+n_B)$
  - Suppone che append e drop abbiano costo O(1)
  - Vero se A, B ed S rappresentati con liste ordinate

```
algorithm merge(A, B) is
    inputs A, B : list
    returns list

    C := new empty list
    while A is not empty and B is not empty do
        if head(A) ≤ head(B) then
            append head(A) to C
            drop the head of A
        else
            append head(B) to C
            drop the head of B

    // By now, either A or B is empty. It remains to empty the other input list.
    while A is not empty do
        append head(A) to C
        drop the head of A
    while B is not empty do
        append head(B) to C
        drop the head of B

    return C
```

Intersezione e differenza

- Si può applicare lo stesso algoritmo
- Intersezione
  - Si *copiano* soltanto gli elementi presenti presenti sia in A che B
- Differenza
  - Si copiano gli elementi presenti in A *ma non* in B
- Esercizio: scrivere lo pseudo-codice per intersezione e differenza