Liste Concatenate

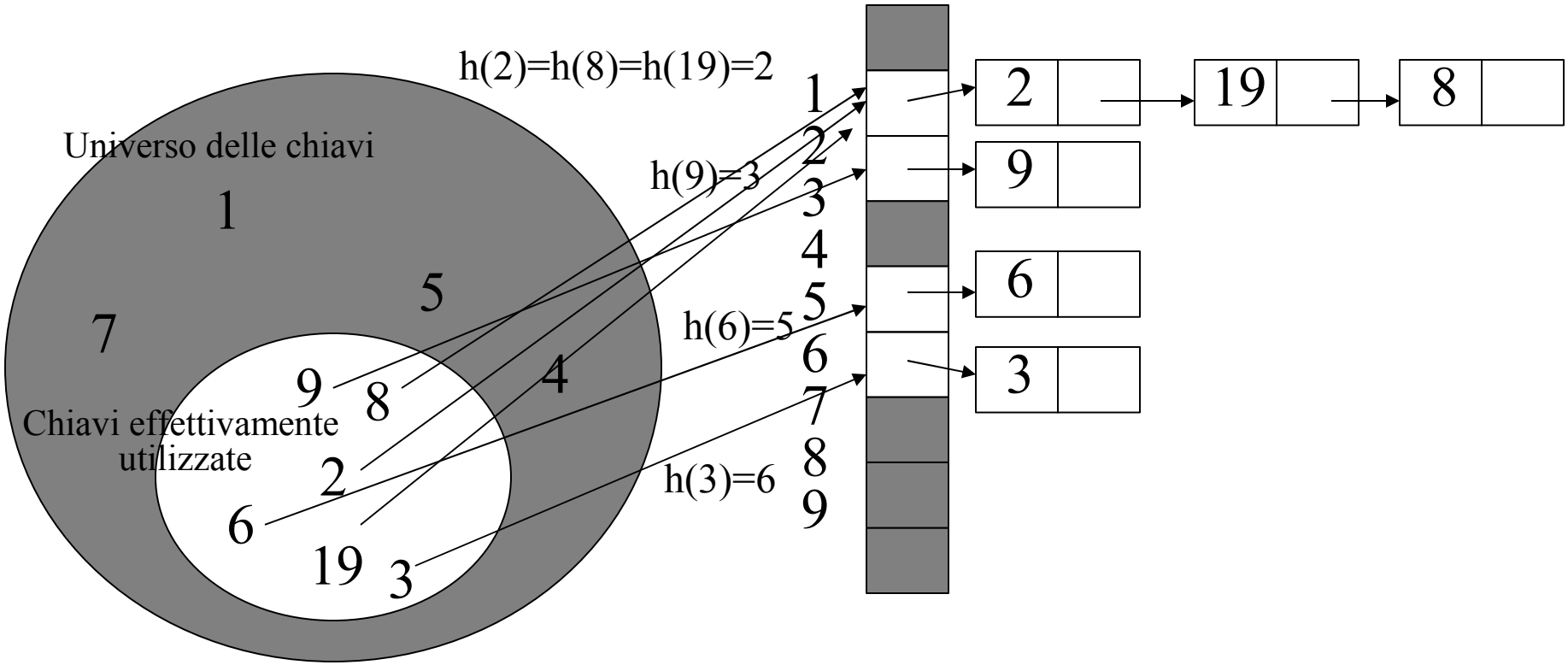
Metodi per la risoluzione delle collisioni

- Per risolvere il problema delle collisioni si impiegano principalmente due strategie:
  - metodo di concatenazione
  - metodo di indirizzamento aperto

Metodo di concatenazione

- L'idea è di mettere tutti gli elementi che collidono in una lista concatenata
- La tabella contiene in posizione j
  - un puntatore alla testa della j-esima lista
  - oppure un puntatore nullo se non ci sono elementi

Visualizzazione



Pseudo codice

```
Chained-Hash-Search(T,k)
1  ricerca un elemento con chiave k in lista T[h(k)]

Chained-Hash-Insert(T,x)
1  inserisci x in testa alla lista T[h(key[x])]

Chained-Hash-Delete(T,x)
1  elimina x dalla lista T[h(key[x])]
```

Fattore di carico

- Data una tabella hash T con m posizioni che memorizza n elementi, si definisce il *fattore di carico* α per T

$$\alpha = n/m$$

- α è anche il numero medio di elementi memorizzati in ogni lista concatenata
- Per il metodo con liste concatenate α è in genere ≥ 1
- L'analisi della complessità viene fatta in funzione di α
- Si suppone che α sia definito anche se n e m tendono all'infinito

Tempo di calcolo per l'inserimento/cancellazione

- Si fa l'ipotesi che il *tempo di calcolo di h* sia O(1) così che il tempo di calcolo delle varie operazioni dipende solo dalla lunghezza delle liste
- Il tempo di esecuzione per *l'inserimento* è O(1) nel caso peggiore
- Il tempo di esecuzione per la *cancellazione* è O(1) se le liste sono *bidirezionali,* senza contare il tempo per trovare l'elemento da cancellare
  - Nota: se le liste sono *semplici* si deve prima trovare il predecessore di x per poterne aggiornare il link next. In questo caso la cancellazione e la ricerca hanno lo stesso tempo di esecuzione

Tempo di calcolo per la ricerca

- Il *caso peggiore* si ha quando tutte le chiavi collidono e la tabella consiste in una unica lista di lunghezza n
- In questo caso il tempo di calcolo è O(n)

- Il *caso medio* dipende da quanto in media la funzione hash distribuisca l'insieme delle chiavi sulle m posizioni
- Se la funzione hash soddisfa l'ipotesi di *uniformità semplice* allora ogni lista ha lunghezza media α

Tempo di calcolo per la ricerca con insuccesso

- Il tempo medio è il tempo impiegato per generare il valore hash data la chiave (che è indipendente dal numero di elementi e quindi assimilabile ad una costante) ed il tempo necessario per scandire una sequenza fino alla fine
- Dato che la lunghezza media di una sequenza è α si ha $O(1+\alpha)$

Tempo di calcolo per la ricerca con successo

- Il *tempo medio* è il tempo impiegato per *generare* il valore hash data la chiave più il tempo necessario per *scandire* una sequenza fino a determinare l'elemento con chiave cercata
- Si faccia l'ipotesi che gli elementi vengano *inseriti in coda* alla lista *(in realtà si dimostra che è equivalente se l'inserimento avviene in testa alla lista)*
- In media un elemento e' posizionato verso la meta' della lista e quindi nuovamente otteniamo $O(1+\alpha)$
- ..ma possiamo cercare di capire se esiste un limite inferiore sfruttando il fatto che inizialmente gli elementi sono messi in liste inizialmente corte che si allungano via via

Tempo di calcolo per la ricerca con successo

- Dapprima gli elementi sono inseriti all'inizio della lista
  - dunque verranno ritrovati dopo pochi confronti
- ...dopo molte operazioni di inserimento gli elementi si troveranno verso la fine della coda
- Il tempo impegato per la ricerca gli elementi dipende dunque dall'elemento cercato e da quando questo è stato inserito nella tabella
- Si procede calcolando il tempo di ricerca per ogni elemento inserito in ordine *i*

Tempo di calcolo per la ricerca con successo

- Si indica con i, l'elemento i-esimo secondo l'ordine di inserimento
- Se le liste crescono in modo uniforme dopo (i-1) inserimenti avro' liste lunghe in media (i-1)/m
- Il numero medio di elementi esaminati prima di inserire il nuovo elemento i-esimo sarà allora (i-1)/m
- Il numero medio di elementi esaminati durante la *ricerca con successo* dell'elemento i-esimo sarà pertanto 1+ (i-1)/m
  - cioè si esamina l'elemento stesso e gli (i-1)/m elementi precedenti

Tempo di calcolo per la ricerca con successo

- In media si avrà:

  $1/n \sum_{i=1..n} 1+ (i-1)/m =$

  $1+ 1/(nm) \sum_{i=1..n} (i-1) =$

  $1+ 1/(nm)\ n(n-1)/2 =$

  $1 + \alpha/2 -1/2m =$

  $O(1+ \alpha)$

- L'analisi piu' accurata mostra che esaminiamo meno di $\alpha/2$ elementi in media
- ..ma l'ordine di grandezza non cambia

Tempo di calcolo per hash per concatenazione

- In sintesi, se il numero n di elementi nella tabella è proporzionale al numero di posizioni m, allora n = O(m) e α = n/m ovvero

$$\alpha = O(m)/m = O(1)$$

- Quando questo accade, ovvero quando il fattore di carico è una costante (piccola) la ricerca richiede un tempo O(1)

- Tutte le operazioni di inserzione, cancellazione e ricerca richiedono un tempo O(1)