Hash

⇒ 用"key"快速找資料(不用traversal)

⇒ key → hash function → index → value

⇒ 用質數作表格大小m避免重複模式 (2的次方數容mix bits)

- 整數 index = (key) % m (大數分解后 % m)
- 非整數 ⇒ 轉成ASCII code后相加 (易collision ⇒ CAT = ACT)
  - ⇒ 加權式字串Hash

⇒ load factor 負載因子 (α) — 用以衡量表格有多滿

時間複雜度(搜尋/插入/刪除)
- Seperate chaining
  - Best: O(1)
  - Average: O(1+α) → α是linked list平均長度
  - Worst: O(n) → 所有資料都在同條linked上
- Open Addressing
  - Average: O(1)
  - Worst: O(n)
  - → 刪除要使用"Lazy Delete"(標記刪除)

collsion 處理
- Chaining 鏈結法 ⇒ collision了就用linked list串在一起
- Open addressing 開放定址法 ⇒ 往后找空位 (方法: 1. 線性探測 2. 平方探測 3. 双重雜湊
- composite key 複合鍵 ⇒ 一次用2個特徵
- Hash Refinement 改良Hash ⇒ 重寫

⇒ Open addressing 開放定址法
- 1. linear probing 線性探測
  - 优: 簡單
  - 缺: 主群聚
- 2. Quadratic probing 平方探測
  - 优: 減少主群聚
  - 缺: 有很多漏掉的空位、出現次群聚
- 3. Double hashing 双重雜湊
  - 优: 分佈 good
  - 缺: 難計算

⇒ 問題: 1. α越高、collision越多、效能越差
2. 群聚效應

⇒ Primary clustering 主群聚 ⇒ 連續佔用的位置形成一個群集使新插入的key需要探測更長的時間，並使群集變得更大。

Hash function 種類
- 除法: k % m
- 乘法: 利用小數打散分佈
- 折疊法: 把key切段后相加
- string hashing: 多項式

應用: 1. DNS快取

优點: 1. 插入、刪除、查找平均時間複雜度 O(1)

ADT { Hashing Dictionary ⇒ Create 建立空字典 / IsEmpty / Search / Delete / Insert

Seperate Chaining ⇒ Create(m) 建立 m 個桶 / Insert 計算 hash function (&更新) / Retrieve (Search)
Delete