

study note

deque

deque

map

map size

my off

ysize

→ 指向的 data

→ 有多少格

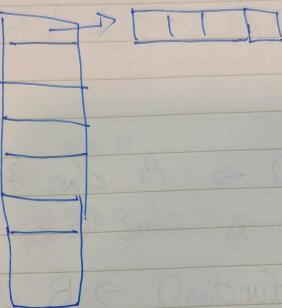
→ 第一個元素放在哪？

→ 哪個元素

ex: long = 4
long long = 8

↓
deque

map
mapsize 8
my off X
ysize y



deque size ⇒ 跟距資料型態不同而
改

copy deque 太大時

insert

→ 位置 (底王里的) 表格上半

off <= mdata.mySize 前半 ⇒ 把 往上提
/ / > / / 後半 ⇒ / / 下 往下推

erase

off < / /
off > = / /

往下推
往上提

hash

insert (deque1.begin, x)

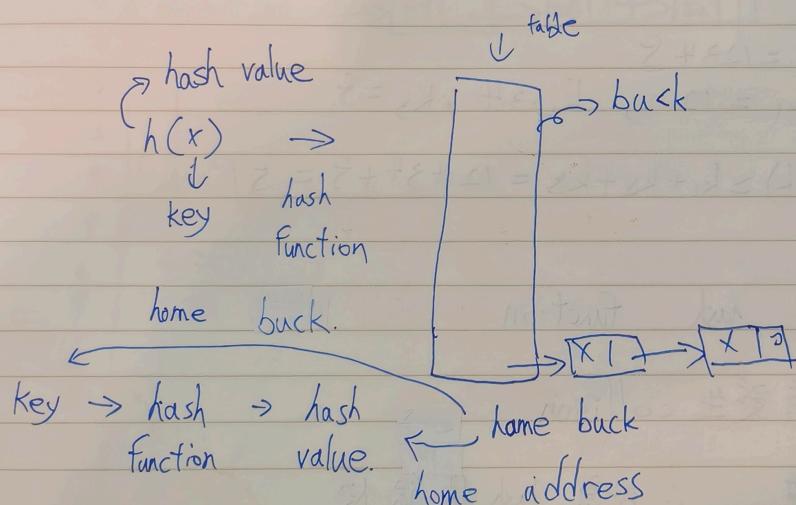
如果一開始為空 deque 則最後一個
else 放在第一個 element 前

insert (deque1.end, y)

一開始為空，則放入最後一個
else 放在最後 element 後

hash function

hash collision \Rightarrow 兩個 hash value share 1 buck



hash function 通常的 3 種 method

(位址)

collision 碰撞：當二筆不同的資料雜湊值相同時

Overflow 溢位：當雜湊值在 Bucket 中的 slot 已滿時
又出現一個時

常見的雜湊函式 (最常見) (學習) Level

1. 除法 (Mod / Division)

Data % x.

2. 中間平方法 (Middle square)

Data = 235

$235^2 = 55225 \rightarrow$ 取中 3 = 522.

3. 折疊相加法 (Folding Addition)

又可以分成兩類

3-1 Shift (移位)

ex: 987586265 $\Rightarrow 987 + 586 + 265 = 1838$

3-2 Boundary (邊界)

ex: 987586265 $\Rightarrow 987 + 685 + 265 = 1737$

可以是偶/奇數段

4 數位分析法 (Digits Analysis)

電話號碼除去頭的 09，並假設只要 2 位數

那就分析一下剩的 8 位數哪個分佈比較平均

* 方法要求已知 data 內容

Overflow 處理方法

1. 線性探測法 (Linear Probing) \Rightarrow 直接找 (由上往下)

1. divide Method

→ 10^4 的會比較好。

2. Mid square Method

ex

$$h(40) = 60$$
$$\downarrow$$
$$40^2 = 1600$$

3. Digit Folding Method.

ex

$$k = 12345$$

$$\Rightarrow k_1 = 12, k_2 = 34, k_3 = 5$$

$$h(k) = k_1 + k_2 + k_3 = 12 + 34 + 5 = 51$$

perfect hash function

⇒ 沒有發生 collision

性質

1. 計算 hash 值很快

2. key 分佈很平均 (hash value 分散)

FNV-1 & FNV-1a. hard to tell

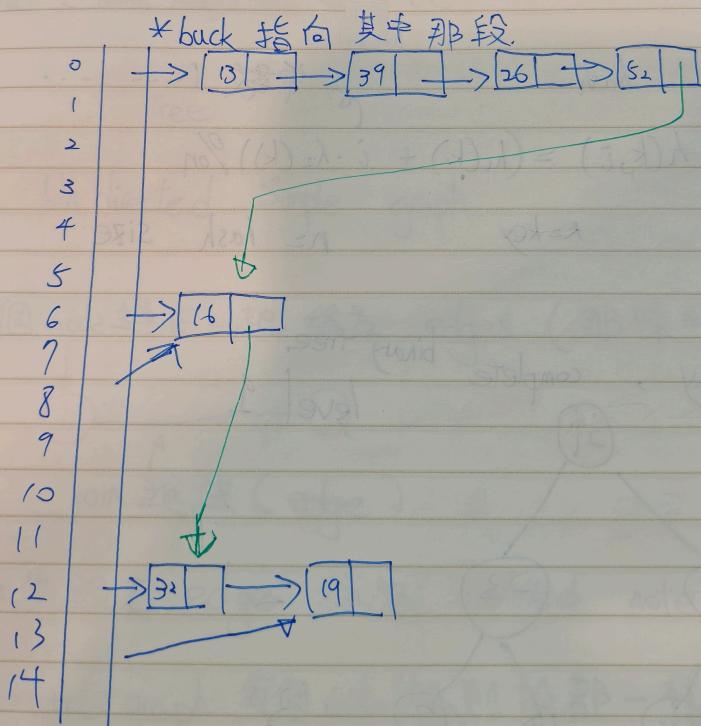
這還可以分成 32 bit - 64 - 128 - ... - 1024

$\text{hash} = \text{offset_basis}$
 for each octet_of_data to be hashed
 $\text{hash} = \text{hash} \times \text{FNV_prime}$
 $\text{hash} = \text{hash} \text{ xor } \text{octet_of_data}$

- 1a

collision 處理方法

用 linklist (會把所有串在一起)



Open Addressing

如果 buck 已有 就往下找到第一個空格

Linear 尋找 $(h(k) + i) \% b$ $(h(k) + (b-1)) \% b$.

$b = \text{格子數量}$

Quadratic

$(h(k) + i) \% b$, $(h(k) + i^2) \% b$... $(h(k) + (b-1)^2) \% b$

Double hash

常數 1, 2, ...

$$h(k, i) = (h_1(k) + i \cdot h_2(k)) \% n$$

$k = \text{key}$

$n = \text{hash size}$

heap : array, complete binary tree, level 1

