My Notes — Important Concepts worth keeping

Hashing 雜湊

Today: / /

✱ Basics

1. Hash table：以陣列型態來存放資料
2. Hash function：透過 key 值計算出該放在 Hash table 哪個位置。

key → Hash → location

✱ Collision 碰撞

- Definition：同一個 Hash table 的位置放了 2 次。（放進去前發現裡面已經有資料了）

$$Prob\{N\ different\} = \frac{(table\ size-1)\times\cdots\times(table\ size+1-N)}{(table\ size)^{N-1}}$$

$$Prob\{collision\} = 1 - Prob\{N\ different\}$$

My Questions — Problems & Difficulties needing exploration

Fuctions

1. Digit selection：

109271**0**4 ⟶ 放進 [4]

2. Folding：

key：10927104，table size = 10

① 1+0+9+2+7+1+0+4 = 24

2+4 = 6 ⟶ 放進 [6]

② 109+27+10+4 = 150

1+5+0 = 6 ⟶ 放進 [6]

3. Modulo arithmetic 質數

✱ table size should be prime.

- 當數值很大的時候會使用②

key = 10927104，table size = 13

① 109+27+10+4 = 150

150 % 13 = 7 ⟶ 放進 [7]

or

② (109 % 13) + (27 % 13) + (10 % 13) + (4 % 13) = 7 ⟶ 放進 [7]

My Opinions — Thoughts, inspirations, and suggestions

4. Converting charater strings

key：10027104，table size = 13

① 49+48+48+50+55+49+48+52 = 399

399 % 13 = 9 ⟶ 放進 [9]

② (⋯(49 × 2^8 + 48) + ⋯ + 48) × 2^8 + 52 = 22260

22260 % 13 = 4 ⟶ 放進 [4]

char	ASCII
'0'	48
'1'	49
'2'	50
'3'	51
'4'	52
'5'	53
'6'	54
'7'	55
'8'	56
'9'	57

凡是值得做的事，都是上坡路。 —約翰·霍斯威爾

My Notes — Important Concepts worth keeping

Collision Resolution

Today: / /

a. Linear probing 線性探索

- 找到位置後假如有資料的話就放進下一個。

缺點：搜尋上可能有困難。

b. Quadratic probing 平方探索

- 和 a 類似，但不是 +1，是 $+1^2, +2^2, +3^2$，直到放進去。

優點：較 a 分散

缺點：可能會一直放到重覆的位置。

ex：

key：10027207 → Hash → [5]　Hash：sum % 7

5+1^2 = 6 % 7 = [6]　5+7^2 = 54 % 7 = [5]

5+2^2 = 9 % 7 = [2]　5+8^2 = 69 % 7 = [6]

5+3^2 = 14 % 7 = [0]　5+9^2 = 86 % 7 = [2]

5+4^2 = 21 % 7 = [0]　5+10^2 = 105 % 7 = [0]

5+5^2 = 30 % 7 = [2]

5+6^2 = 41 % 7 = [6]

[0]	10027210
[1]	
[2]	10027216
[3]	
[4]	
[5]	10027135
[6]	10027234

My Questions — Problems & Difficulties needing exploration

My learning weather report

c. Double hashing 双重碰撞

- 使用 2 個 hash functions. h_1 是初使位置，h_2 是 step size.

✱ $h_1 \neq h_2$ 且 $h_2(key) > 0$

✱ table size 最好是質數且 if 和 step size 互質就能探訪每個位置。

ex：

function：sum % 7

step size：5 - (sum % 5)

key：99271**4**4

9+9+2+7+1+4+4 = 36

36 % 7 = [1] (h_1)

h_2 = 5 - (36 % 5)

= 5 - 1

= 4

final location：

1+4 = 5 % 7 = 5

5+4 = 9 % 7 = 2

[0]	
[1]	98271**4**5
[2]	99271**4**4
[3]	
[4]	
[5]	99271**3**5
[6]	10027203

My Opinions — Thoughts, inspirations, and suggestions

Ex：

Hash function：sum % 7，step size：sum % 5 + 1

	sum	h_1	h_2	location
99271**0**4	32	[4]	×	[4]
10027101	12	[5]	×	[5]
10027161	18	[4]	4	[1]
10027249	25	[4]	1	[6]
98271**6**6	39	[4]	5	[2]
10027262	20	[6]	1	[0]

Number of probes：1+1+2+3+2 = 11

習慣就是習慣，誰也不能將其扔出窗外，只能一步一步地引下樓。 —馬克·吐溫

密碼 cipher key

My Notes
Important Concepts worth keeping

Collision Resolution (進階) Today: / /

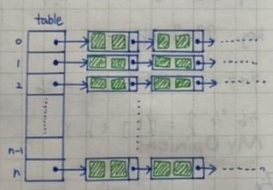
a. Buckets : (多容器)

* 每個 hash table 的位置都是 array，讓一個位置能存放很多資料。

b. Separate chaining : (分開鏈結)

* 每個 hash table 的位置都是一個 linked list (動態陣列)。

* Bucket + Separate chaining :
  - Memory management
  - Worst case : O(n)



Efficiency :

* Average-case Analysis

1. Load factor α
   - Current number of items in the table size.
   - Measure how full a hash table is.
2. Depends on whether the search is successful.
   - Unsuccessful searches generally require more time than successful searches.

You do not rise to the level of your goals. You

My Questions
Problems & Difficulties needing exploration

雜湊的應用

My learning weather report

1. Hash Join

Accidents

license
MXT-612
LCD-123
KIA-214
NCC-179
MXT-500
LKK-100

Maintenance

license
CBC-300
LCD-123
NCC-179
LKK-100
DHT-911

Nested-loop join: 6×5 = 30 comparisons.

先 build :

[0]	CBC-300 → LCD-123
[1]	LKK-100
[2]	NCC-179 → DHT-911

後 probe :

MXT-612	[0]
LCD-123	[0]
KIA-214	[1]
NCC-179	[2]
MXT-500	[2]
LKK-100	[1]

5 builds
6 probes
2+2+1+2+2+1 = 10 comparisons

2. Count-Min Sketch

	H1	H2
[0]	1	3
[1]	0	1
[2]	2	0
[3]	1	1
[4]	2	3
[5]	1	
[6]	1	

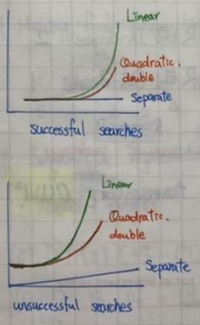
CAB → 3 1 2
ABE → 1 2 5
FG → 6 7

(key+3)%5
(key×2)%7

Counting :
- A = 2
- B = 2
- C = 1
- D = 0
- E = 1
- F = 1
- G = 1

* 選小的
* 可能不正確
  ex: 再加入3個I的話 B=5，但B只有2。

My Opinions
Thoughts, inspirations, and suggestions



Inefficient Operations on Hashing :

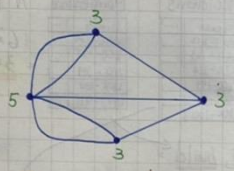
* Traversal
  - Visit all the data in sorted order.
* NN Search
  - Find the item that has the smallest or the largest search key.
* Range Query
  - Find the items between two searches keys.

決定你成功或失敗的，不是你的目標，而是你的系統。《原子習慣》

My Notes
Important Concepts worth keeping

Ch6. 圖形概論 Today: / /

* 七橋問題
  - G = {V, E}
  - V(G)：點的數量
  - E(G)：邊的數量
  - degree：點上邊的數量



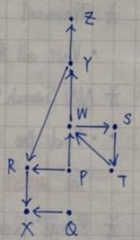
$\sum \text{degree}(V_i) = |E(G)| \times 2$

5+3+3+3 = 7×2

* 一筆畫
  - degree 皆為偶數
  - 0 or 2 個點的 degree 為奇數

* Adjacency Matrix :

	P	Q	R	S	T	W	X	Y	Z
P	0	0	1	0	0	1	0	0	0
Q	0	0	0	0	0	0	1	0	0
R	0	0	0	0	0	0	0	1	0
S	0	0	0	0	1	0	0	0	0
T	0	0	0	0	0	1	0	0	0
W	0	0	0	1	0	0	0	1	0
X	0	0	0	0	0	0	0	0	1
Y	0	0	1	0	0	0	0	0	0
Z	0	0	0	0	0	0	0	0	0



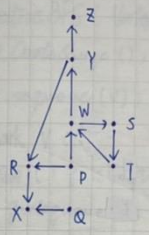
- R 的 In-degree : 2
- R 的 Out-degree : 1
- R ← P
  successor predecessor
- traverse(G) : $O(|V|^2)$

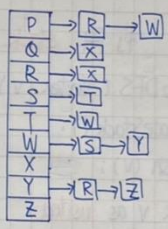
The amount of time you have been performing a habit is not as important as the number of the times you have performed it. 《Atomic Habits》

My Questions
Problems & Difficulties needing exploration

Adjacency List

My learning weather report





* 只能求 Out-degree
* traverse(G) : O(|V|+|E|)

My Opinions
Thoughts, inspirations, and suggestions

Other Graph Representations

* Mapping from vertex labels to array indices

P	Q	R	S	T	W	X	Y	Z
0	1	2	3	4	5	6	7	8

* Sequential representation
  - nodes + edges

[0]	[1]	[2]	[3]	…	[7]	[8]	[9]	[10]	[11]	[12]	…
10	12	13	14	…	19	20	20	> 5	6		…

Q→X

圖形回上

執行習慣的時間長短，習慣的次數多寡重要。《原子習慣》

My Notes
Important Concepts worth keeping

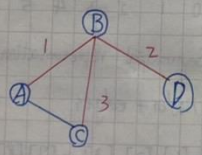
走訪

Today: / /

✳ DFS：利用 stack

```
Iterative DFS ( Vertex v ) {
  S. createStack ();
  S.push (v);
  Mark v as visited;
  while ( ! s. IsEmpty() ) {
    u = s.getTop();
    if (unvisited w is adjacent to u) {
      s.push (w);
      Mark w as visited;
    } // if
    else s.pop();
  } // while
} // DFS
```

DFS traversal sequence:
AB. BC. BD



暫停吸口氣，
靜思問自己，
閉鍵 10 秒鐘，
Ho 住十年功。

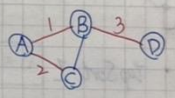
My Questions
Problems & Difficulties needing exploration

BFS：利用 queue

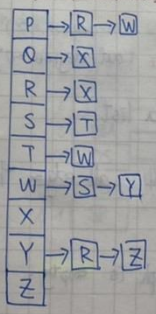
My learning weather report

```
Iterative BFS ( Vertex v ) {
  q. CreateQueue ();
  q. enqueue (v);
  Mark V as visited;
  while ( ! q. isEmpty () ) {
    q. dequeue (u);
    while (each unvisited w is adjacent to u) {
      Mark w as visited;
      q. dequeue (w);
    } // while
  } // while
} // BFS
```

BFS traversal sequence:
AB. AC. BD



My Opinions
Thoughts, inspirations, and suggestions



DFS：PR. RX. PW.
WS. ST. WY. YZ
PRXWSTYZ

BFS：PR. PW. RX.
WS. WY. ST. YZ
PRWXSYTZ

密碼 cipher key

慢比快還要快，
《三易則》

My Notes
Important Concepts worth keeping

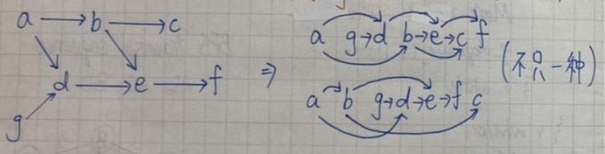
Ch7 圖形應用

Today: / /

✳ Topological Sort

- Definition：
  必須是 Directed Acylic Graph (DAG)
  具有方向性且沒有 cycles

- Example：



- topSort 1：

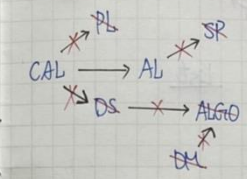
predecessor (in-degree ≠ 0)

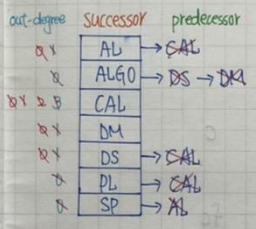
1. Find a vertex that has no successor (out-degree = 0).
2. Add the vertex to the beginning of a list.
   end
3. Remove that vertex from the graph, as well as all edges that lead to it.
4. Repeat the previous steps until the graph is empty.

My Questions
Problems & Difficulties needing exploration

out-degree = 0

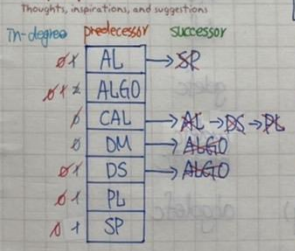
My learning weather report





① ALGO
② DM, ALGO
③ DS, DM, ALGO
④ PL, DS, DM, ALGO
⑤ SP, PL, DS, DM, ALGO
⑥ AL, SP, PL, DS, DM, ALGO
⑦ CAL, AL, SP, PL, DS, DM, ALGO

My Opinions
Thoughts, inspirations, and suggestions



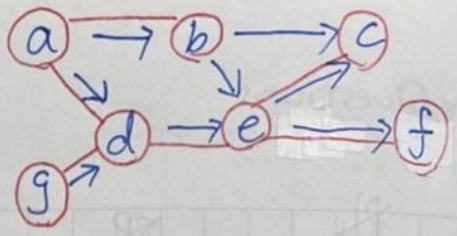
in-degree = 0

① CAL
② CAL, DM
③ CAL, DM, DS
④ CAL, DM, DS, PL
⑤ CAL, DM, DS, PL, AL
⑥ CAL, DM, DS, PL, AL, ALGO
⑦ CAL, DM, DS, PL, AL, ALGO, SP

My Notes

Important Concepts worth keeping

- topSort 2 (利用 DFS)



Action	Stack s	List
☆ push a	a	
☆ push g	ag	
push d	agd	
push e	agde	
push c	agdec	
pop c, add c to List	agde	c
push f	agdef	
pop f, add f to List	agde	fc
pop e, add e to List	agd	efc
pop d, add d to List	ag	defc
pop g, add g to List	a	gdefc
push b	ab	
pop b, add b to List	a	bgdefc
pop a, add a to List	(empty)	abgdefc

(從左→右，所以先將 in-degree = 0 的放入 stack。)

人生不應像孤島，
有時轉身分享，
有時求助，
有時轉身打氣，
有時互相提攜。
【轉身功】

My
Problems

My C
Thoughts,