My Notes
Important Concepts worth keeping

Today / /

1. Pointer to its parent is red
   It is a right child
   @ It sibling must point to its child by red
   sibling
   (a) No child : recolor
   (b) one or two child : rotate + recolor

2. Pointer to its parent is black
   It is a right child
   @ It sibling must point to its child by red
   sibling
   (a) No child : recolor + ?
   (b) one or two child : rotate + recolor

3. Pointer to its sibling is red
   It is a right child
   @ It sibling must have to black pointer
   @ LL rotation + recolor

sibling
(a) No child : ok
(b) one child : rotate
(d) Two child : rotate + recolor

My Questions
Problems & Difficulties needing exploration

L5 Hashing 雜湊

My learning weather report

Hash table 表格
Hash function (key → location mapping) 表格位置
key 資訊

Perfect has function 幾乎不可能做到
- Map each search key into a unique location
- Possible only if all the search keys are known

所以 ⇓ 會有此問題

Collision 碰撞

定義：兩個不同的 key，对應到相同位置

My Opinions
Thoughts, inspirations, and suggestions

Example：至少有多少人在一起，裡面有兩個同天生日的人會超過五成？

$Prob\{N \text{ different birthday}\} = 364 * \cdots * (366-N)/365^{N-1}$

$N=22 \Rightarrow 364 * \cdots * 344/365^{21} = 0.5243$

$N=23 \Rightarrow 364 * \cdots * 343/365^{22} = 0.4927$

此為不同天生日機率

$1-0.4927 = 0.5073 > 0.5$

同天生日機率

My Notes
Important Concepts worth keeping

Today / /

Simple hash function

1. Digit selection 數字選擇
2. Folding 折疊
3. Modulo arithmetic → 表格要是"質數"(prime)個，才不容易 Collision
4. Converting character strings
   - Use integers in the hash function instead of search string
     - positional sum

   若數字太大可以，除以表格數取"餘數"

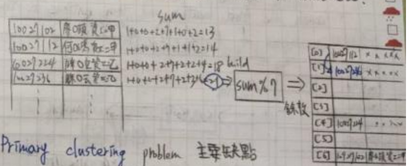
Collision Resolution

1. Open addressing → a.b. 缺點產生の probing sequnces 都跟key有關
   探索
   Probe for an empty location in the hash table
   As the hash table fills, collisions increase
   d. Increase of table size 把 table size 变大
   - Need to hash the items again
2. Restructuring the hash table
   Allow the hash table to accommodate more than one item in the location
   一個空格可放多個 item

My Questions
Problems & Difficulties needing exploration

My learning weather report

1. Open addressing
   a. Linear (線性) probing → 效率較低，但每點 item 都可放的進去
   - search the first location and then the next etc.



Primary clustering problem 主要缺點
- Item tend to clustering together and large clusters tend to get even larger
- Large clusters cause long probing

My Opinions
Thoughts, inspirations, and suggestions

Deletion problem 次要缺點
- Empty locations after deletions would incorrectly stop (錯誤停止) a probing sequence (探查).

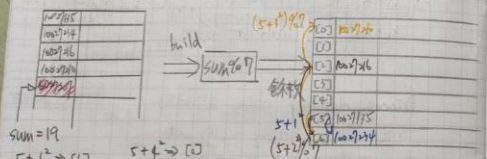
My Notes
Important Concepts worth keeping

Today: / /

1. Open addressing 開放

b. Quadratic probing → 效率較 a. 高，但有些 item 放進去

• Search the first location and then continue at the increments of $1^2, 2^2, 3^2, 4^2, 5^2 \cdots$ and so on



sum = 19

$5+1^2 \Rightarrow [6]$ $5+4^2 \Rightarrow [2]$

$5+2^2 \Rightarrow [2]$ $5+5^2 \Rightarrow [6]$

$5+3^2 \Rightarrow [0]$ $5+6^2 \Rightarrow [5]$

⋮ 慘，放不進去表格

Probing sequence may not visit every location in the hash table

Second clustering problem 次要原因

• different keys at same location create the same probing sequence

You do not rise to the level of your goals. You fall to the level of your systems. (Atomic Habits)

20

My Questions
Problems & Difficulties needing exploration

My warning weather report

c. Double hashing 分派

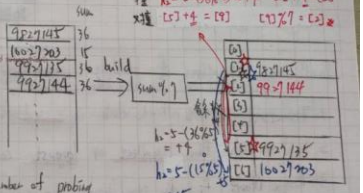
1. It create key dependent probing sequence
2. Use two hash function h_1, h_2 to reduce clustering problem.
3. h_1 for the first location and h_2 for the step size 特制步伐

$h_1 \neq h_2$, $h_2(key) > 0$

Example:

h_1: sum % 7

h_2: 5 − (sum % 5) → 自訂



Number of probing

$= 1+2+2+3 = 8$

Average number of probing $= \frac{8}{4} = 2$ 次

Load factor $= \frac{\text{Item}}{\text{hash table}} = \frac{4}{7} = 0.57\cdots$

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》

21

My Notes
Important Concepts worth keeping

Today: / /

2. Restructuring the hash table

a. Buckets 水桶

• Each location in the hash table is itself an array 這樣 collision 的發生，但 Buckets 滿之後，還是必需再叫前面的三種 open address 或 separate chaining 來處理！

b. Separate chaining 分開的鏈結串到 (pointer)

• Each hash table location keeps a linked list
• Successfully resolves collisions
• The size of the hash table is dynamic 動態的

Inefficient Operations of Hashing ~~Hash不適合做~~

1. Traversal: Visit all the data in sorted order
2. NN search: Find smallest or largest search key
3. Range Query: Find the item between two search key

應用：join, Sketch.

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)

22

My Questions
Problems & Difficulties needing exploration

My warning weather report

06.

Seven Bridge of konigsberg 七橋問題

$G_1 = \{V, E\}$

$V(G_1)$: vertex set 性地

$E(G_1)$: edge set 橋

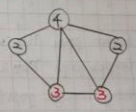
degree 每個點連到邊的個數

$\sum_{\text{vertex}} \text{degree}(V_i) = |E(G)| \times 2$ ⇒ $(5+3+3+3) = 7 \times 2$

@ How to every edge exactly once? (又走訪每個只一次?)

奇數的 degree 只會有 0 or 2 個

by Eulerian path [歐拉路徑]



My Opinions
Thoughts, inspirations, and suggestions

[沒有 Odd degree: being and end at the same vertex
  2個 Odd degree: being and end at odd degree

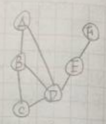
執行習慣的時間長短，不如執行該習慣的次數多寡重要。《原子習慣》

23

My Notes
Important Concepts worth keeping

Today: / /

Undirected graph 無向圖
Directed graph 有向圖
Adjacent vertices 相鄰的
Edge is incident to vertices 邊和哪些邊有關



Path: a sequence of edges 路徑
Cycle: being & end at same vertex 迴圈

6 degrees of separation from the small world experiment.

Simple path: a path that passes through any vertex only once
Simple cycle: a cycle that passes through the other vertices only once

---

Connect graph: There is path between any two vertices
Complete graph: There is an edge between any two vertices
Strong connected graph (有向圖): For any vertices on a digraph, there is a path from one vertex to the other.

Weighted graph: the edge have numeric labels

暫停吸口氣，
靜思問自己，
關鍵 10 秒鐘，
Ho 住十年功。
【且慢功】

My Questions
Problems & Difficulties needing exploration

My learning weather report

在程式中儲存分成矩陣，列表

[predecessor 先行者
[successor 後繼者

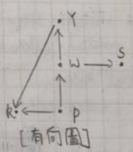
有向圖會有到.

Adjacency Matrix 矩陣 (缺点會浪費空間)

traverse(g): $O(|V|^2)$ 走所有的边

若有 weight: 可以將 (1,0) 改成 weight 数字

	P	R	S	W	Y
P	0	1	0	1	0
R	0	0	0	0	0
S	0	0	0	0	0
W	0	0	0	0	1
Y	0	1	0	0	0

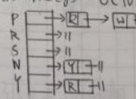


[有向圖]

My Opinions
Thoughts, inspirations, and suggestions

Adjacency list 列表

traverse(g): $O(|V|+|E|)$



(目前只找到 Out-degree)

Sequential 陣列 (缺点新增,刪除很麻煩)

P R S W Y
0 1 2 3 4

— node + edges

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]
6	11	11	8	10	11	1	3	2	4	1

空下

慢比快還要快。
《三余斜說》

My Notes
Important Concepts worth keeping

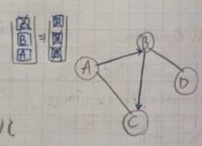
Today: / /

DFS and BFS

Depth-First Search (DFS) Traversal 深度優先

- stack: 後進先出
- 遞迴較好寫

```
① recursiveDFS(Vertex X) {
       Make v as visited;
       for(each unvisited vertex u adjacent to V)
           recursiveDFS(u);
   }
```




```
② iterativeDFS(Vertex V) {
       s.createStack();
       s.push(v);
       while(!s.isEmpty()){
           u=s.getTop();
           if (unvisited vertex w is adjacent to u) {
               s.push(w);
               Mark w as visited;
           }
           else s.pop();
   }
```

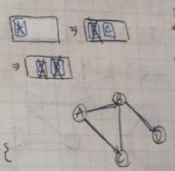
一步一步再一步，
山窮水盡疑無路，
一寸一寸又一寸，
柳暗花明又一村。
【一步功】

My Questions
Problems & Difficulties needing exploration

My learning weather report

Breadth-First Search (BFS) Traversal 寬度優先

— queue
— 迴圈較好寫




```
iterativeBFS(Vertex v) {
    q.createQueue();
    q.enqueue(v);
    Make v as visited;
    while(!s.Empty()){
        q.dequeue(u);
        for(each unvisited vertex W adjacent to u){
            Make W as visited;
            q.enqueue(w);
        }
} // iterativeBFS()
```

My Opinions
Thoughts, inspirations, and suggestions

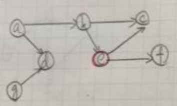
好還可以更好，
高還可以更高，
只要一步步向上跑。
【一步功】

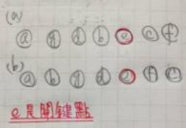
My Notes U7 Topological Sort 拓樸

Important Concepts worth keeping

Today / /

- Acyclic Digraph / Directed Acyclic Graph / DAG [無迴環的有向圖]
- 須要有向邊來定義順序
- 答案不唯一!





Activity-on-vertex (AOV) network

① toSort

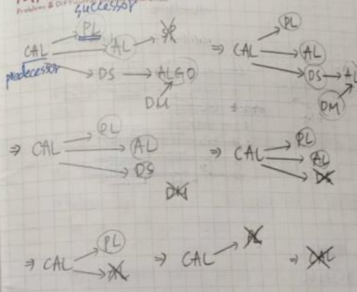
1. Find a vertex that has no successor (out-degree = 0)
   (可以) no predecessor (in-degree = 0)
2. Add the vertex to the beginning of a list
3. Remove the vertex from the graph, as well as all edges that lead to it
4. Repeat the previous steps until the graph is empty

- When the loop ends, the list of vertices well be in Topological order.

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 Questions

Problems & Difficulties needing exploration



My Opinions

Thoughts, inspirations, and suggestions

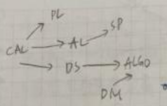
SP
ALGO . SP
DM . ALGO, SP
DS , DM, ALGO, SP
AL, DS, DM, ALGO, SP
PL, AL, DS, DM, ALGO, SP
CAL, PL, AL, DS, DM, ALGO, SP ※

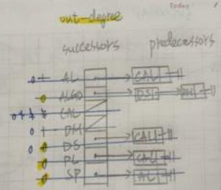
且慢功，停想動，
三思而後行，
謀定而後動！

My Notes

Important Concepts worth keeping

Today / /





ALGO
DM, DS, ALGO
PL, DM, DS, ALGO
SP, PL, DM, DS, ALGO
AL, SP, PL, DM, DS, ALGO
CAL, AL, SP, PL, DM, DS, ALGO ※

一步功，真輕鬆，
一步一腳印，
穩定中前進，
一點一滴滴，
日久見奇蹟。

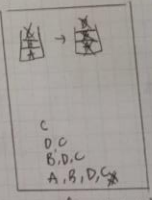
My Questions

Problems & Difficulties needing exploration

② topSort2

- reuse 之前寫的 iterative DFS
- Push all vertices that have no predecessor onto a stack
- Each time you pop a vertex from the stack, add it to the beginning of a list of vertices
  - When the traveral ends, the list of vertices will be in topological order.

```
iterativeDFS( Vertex v) {
    s.createStack();
    s.push(v);
    Mark v as visited;
    while (!s.Empty()) {
        u = s.getTop();
        if( unvisited vertex w is adjacent to u) {
            s.push(w);
            Mark w as visited;
        }
        else {
            //對pop前存值
            s.pop();
        }
    }
}
```



My Opinions

Thoughts, inspirations, and suggestions

My Notes U7 Spanning Tree 生成樹

Important Concepts worth keeping — Today / /

- A tree is an undirected connected graph without cycles [無迴環.無向圖.每個點都相連通]
- DFS, BFS 可形成生成樹
- A connected undirected graph that has n vertices must have at least n-1 edge. ex: 4個點, 3個邊

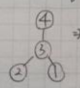
- How many different spanning tree?

2 nodes → $2^{2-2}=1$
3 nodes → $3^{3-2}=3$
4 nodes → $4^{4-2}=16$

- Prufer sequence 压缩的演算法, 生成一維陣列

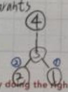
Each labeled tree with n vertices has a unique Prufer sequence of length n-2

- Leaf with the smallest label
- keep the label of its parants



⇒ length (3,3)

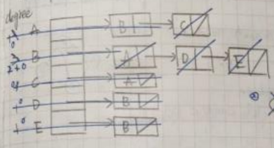
degree: (1,1,1,1) → 3 有2個+1上
(1,1,3,1)
① (0,1,2,1)
② (0,0,1,1)

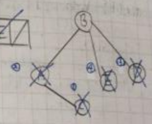


My Questions

Problems & Difficulties needing exploration

example:





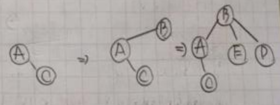
Prufer sequence: A B B [length: 5-2=3]

还原 [ABCDE] ⇒ [1,1,1,1,1] ⇒ 加回 ABB

[2,3,1,1,1] ⇒ [1,3,0,1,1] ⇒ [0,2,0,1,1]
⇒ [0,1,0,0,1]

My Opinions

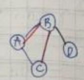
Thoughts, inspirations, and suggestions



My Notes

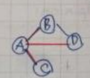
Important Concepts worth keeping — Today / /

利用 DFS 產生生成樹 — DFS traversal sequence: AB, BC, BD



```
iterativeDFS (Vertex v)
    s.createStack();
    s.push(v);
    Mark v as visited;
    while (!s.isEmpty() && count < |V|-1)
        u = s.getTop();
        if( unvisited vertex w is adjacent to u)
            s.push(w);     count++;
            Mark w as visited; // (u,w)
        else s.pop();
```

利用 BFS 產生生成樹 — BFS traversal sequence: AB, AC, AD



```
IterativeBFS ( Vertex v)
    q.createQueue();
    q.enqueue(v);
    Mark v as visited;
    while (!s.isEmpty() && count < |V|-1)
        q.dequeue(u);
        for (each unvisited vertex w adjacent to u)
            Mark w as visited; // (u,w)
            q.enqueue(w);
            count++;
```


My Questions (MST) Minimum Spanning Tree 最小生成樹

Problems & Difficulties needing exploration

- Sum of the edge has minimum weights on a spanning tree
- 應用常見 steiner tree, k-minimum spanning tree

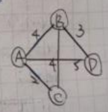
- Robert Prim
  1. Find the least-cost edge (v,u) from a visited vertex v to some unvisited vertex u.
  2. Make u as visited
  3. Add the vertex u and the edge (v,u) to the minimum spanning tree.
  4. Repeat the above steps until all vertices are visited.

◎ use Queue!

My Opinions

Thoughts, inspirations, and suggestions

MST: (A start) AC, AB, BD
(B start) BD, BC, AC



```
PrimAlgorithm( Vertex v)
    Mark v as visited;
    count = 0;
    while (count < |V|-1)
        (v,u) = the least-cost edge from visited to unvisited;
        Mark u as visited;
        Add(v,u) into MST;
        count++;
```

My Notes

Important Concepts worth keeping

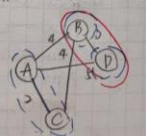
Today: / /

- Joseph Kruskal

1. Create a forest where each vertex is a tree
2. Find the least-cost edge (v, u) where vertex v and vertex u from two different trees
3. Merge the trees of vertex v and vertex u, and add the edge (v, u) to the minimum spanning tree.
4. Repeat the above steps until |V|-1 edges

KruskalAlgorithm( )   MST: AC, BD, AB/BC

```
Assign a unique label to each vertex;
count=0;
while (count<|V|-1)
    (v,u) = the least-cost edge of two vertices with different labels;
    Assign the label min(u,v) to all vertices with these two labels;
    Add (v,u) into MST;
    count++;
```



My Questions

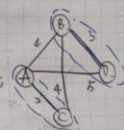
Problems & Difficulties needing exploration

- Otakar Boruvka [類平行處理]

1. Create a forest, where each vertex is a tree
2. For each tree T, do the following steps.
   - 2-1 Find the least edge (v, u) where vertex v is in T and vertex u is outside T
   - 2-2 Merge the trees of vertex v and vertex u, and add the edge (v, u) to the minimum spanning tree
3. Repeat step 2 until only one tree is left.

SollinAlgorithm( )   MST: AC, BD, AB

```
Assign a unique label to each vertex;
size = |V|;
while (size > 1)
    Initial Edges[1...size] as empty sets;
    for each vertex v
        L = v.label;
        (v,u) = the least-cost edge from v to u for any vertex with a different label;
        if( Edges[L].weight > (v,u).weight)
            Edges[L] = (v,u);
    for each edge (v,u) in Edges but not in MST
        Assign min(v.label, u.label) to vertices in the sets of v and u
        Add(v,u) to MST; size--;
```



My Notes   Shortest Paths 最短路徑

Important Concepts worth keeping

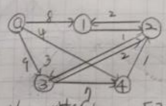
Today: / /

- Dijkstra's Algorithm

Problem definition: Find the shortest paths between a given origin of all other vertices.

Basic idea: A set vertexSet of selected vertices.
An array weight, where weight[v] is the cheapest weight of the shortest path from vertex 0 (origin) to vertex v that passes through only the vertices in vertexSet.

feature: 當想找 A-B-C, A→C 的最短路徑, 必定 A-B 是最短路徑, 也就是當要找一個origin to end 他們之間的子路徑必然也是最短路徑。



	V	vertexSet	[0]	[1]	[2]	[3]	[4]
V-1 Step	—	0	0	8	∞	9	4
	4	0,4	0	8	5	9	4
	2	0,4,2	0	7	5	8	4
	1	0,4,2,1	0	7	5	8	4

0→4→2→1 ※

My Questions

Problems & Difficulties needing exploration

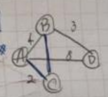
1. Initialize vertexSet (轉換) & weight (因); v = v₀;
2. Update weight for each vertex u not in vertexSet, which is adjacent to v.
   - weight[u] = min{ weight[u], weight[v] + edgeWeight[v,u] }
3. Find the shortest path from 0 to u among every path that starts from 0, passes vertices in vertexSet, and ends at a vertex not in vertexSet.
   - if(weight[u] is minimum), vertexSet = vertexSet + {u};
4. Repeat steps 2,3 until no more vertex can be added.

My Opinions

Thoughts, inspirations, and suggestions

A→B

A→C→B ※



```
DijkstraAlgorithm(Vertex v0)
    weight[0,..n] = {0, ∞ ..., ∞}
    vertexSet = ∅;
    do { Add v into vertex;
        for edge (v,u) where u is not in vertexSet
            weight[u] = min(weight[u], weight[u] + edgeWeight[v,u]);
        cheapest = ∞;
        for vertex u not in vertexSet
            if(weight[u] < cheapest) v=u; cheapest=weight[u];
    } //while(cheapest < ∞)
```

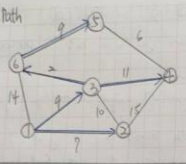
My Notes

Important Concepts worth keeping

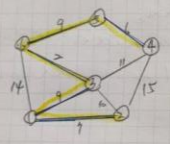
Shortest Path Tree V.s MST

Today:

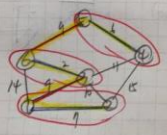
Shortest Path



MST



Prim algorithm



Kruscal's algorithm

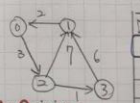
My Questions

Problems & Difficulties needing exploration

All Pairs Shortest Paths

- Floyd-Warshall algorithm

1. Initialize distance matrix D^0 = adjacency matrix;
2. for k=0 to |V|-1
   $D^k \leftarrow D^{k-1}$; // add vertex k into vertexSet
   For i=0 to |V|-1
   For j=0 to |V|-1
   $D^k[i,j] = \min\{D^{k-1}[i,j], D^{k-1}[i,k] + D^{k-1}[k,j]\}$;



D^{-1}	0	1	2	3
0	0	∞	3	∞
1	2	0	∞	∞
2	∞	7	6	1
3	6	∞	∞	0

D^0	0	1	2	3
0	0	∞	3	∞
1	2	0	5	∞
2	∞	7	0	1
3	6	∞	∞	0

D^1	0	1	2	3
0	0	∞	3	4
1	2	0	5	∞
2	9	7	0	1
3	6	∞	9	0

My Opinions

Thoughts, inspirations, and suggestions

A* algorithm

- Best-first search by keeping a priority queue and traversing a path of the lowest expected total cost. [起始 origin → end]

Dijkstra's : four vertices close to the origin

Greed best-first search : four vertices close to the goal

- Expected total cost : $f(v) = g(v) + h(v)$
  $g(v)$: exact cost of the path from the origin to vertex V
  $h(v)$: heuristic estimated cost from vertex v. to the goal.