「あの鹿に触らないで !!」

圖論

Graph Theory

御薬袋托托（@bufhdy）

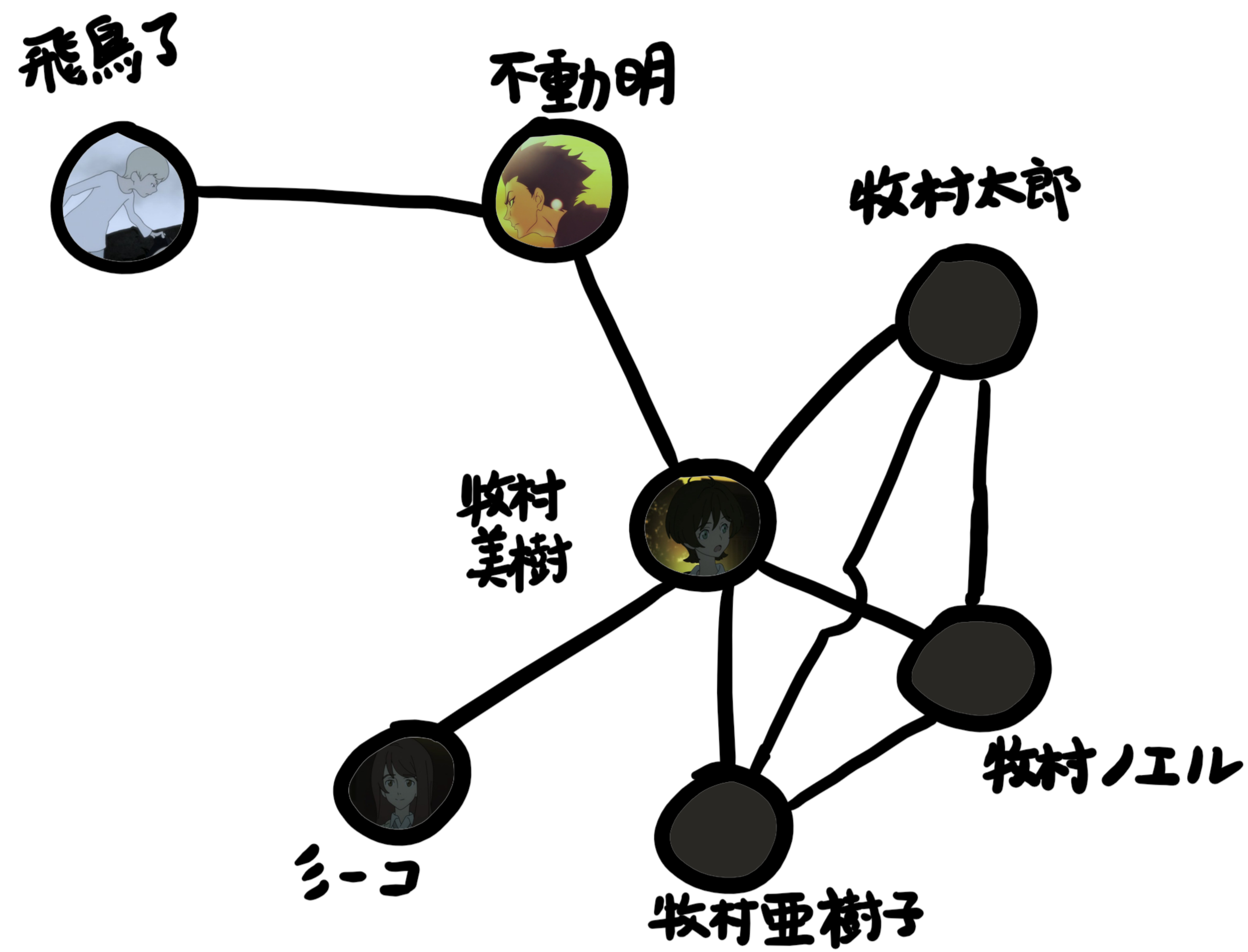
1. 定義 Definitions
2. 圖的表示 Storing a Graph
3. 歐拉迴路 Eulerian Path
4. 拓撲排序 Topological Sorting
5. 最短路徑問題：貝爾曼 – 福特算法
   Shortest Path Problem: Bellman–Ford Algorithm
6. 最小生成樹：Kruskal 算法
   Minimum Spanning Tree: Kruskal's Algorithm
7. 最近公共祖先 Lowest Common Ancestor

I. 定義

Difinitions

一個圖 G 是一個三元組，即 $G = (V, E, \Phi)$。其中，V 是點（vertex，簡為 Vtx）集，E 是邊集，Φ 係 E 到 V 的映射／關聯函數。

每一條邊（edge，或 arc）是一個點對：(v, w)，其中 $v, w \in V$，此時兩點鄰接（adjacent）。特別地，(v,v) 為一個環（loop）。當邊擁有權值（weight，簡作 Wgt，或稱作 cost），它就是賦權的（weighted）。圖的若點對是有序的，就稱圖是有向圖（digraph）。



＊　《Devilman Crybaby》人物關係圖　＊

1. 定義

Difinitions

圖中一條路徑（path）是一個點序 $w_1, w_2, \ldots, w_n$，其中 $(w_i, w_{i+1}) \in E, 1 \leq i < N$。路徑的長（length，簡為 Lgt）是點的總數。長度大於 1 且首尾相連（$w_l = w_n$）的路徑叫做圈（cycle）。一個有向無圈圖經常來進行研究，它的英文是 DAG。

無向圖中某點與其他點連邊的個數稱為它的度（degree），度數為奇數的稱為奇頂點，反之為偶頂點。在有向圖中，流出某點的度稱作出度（out degree），指向某點的度稱作入度（in degree）。

2. 圖的表示

Storing a Graph

關於圖的存儲一共有兩種做法，一是鄰接矩陣（adjacency matrix），一是鄰接表（adjacency list）。

對於鄰接矩陣，我們以 `Graph[MAXX][MAXY]` 二維數組形式儲存。有兩類情況：一是圖是否有向，對於有向圖，`Graph[u][v]` 表示從 *u* 到 *v* 的一條有向路徑；對於無向圖，`Graph[u][v]` 或者是 `Graph[v][u]` 都表示 *u* 和 *v* 之間有一條路徑。第一類情況說完了。

2. 圖的表示

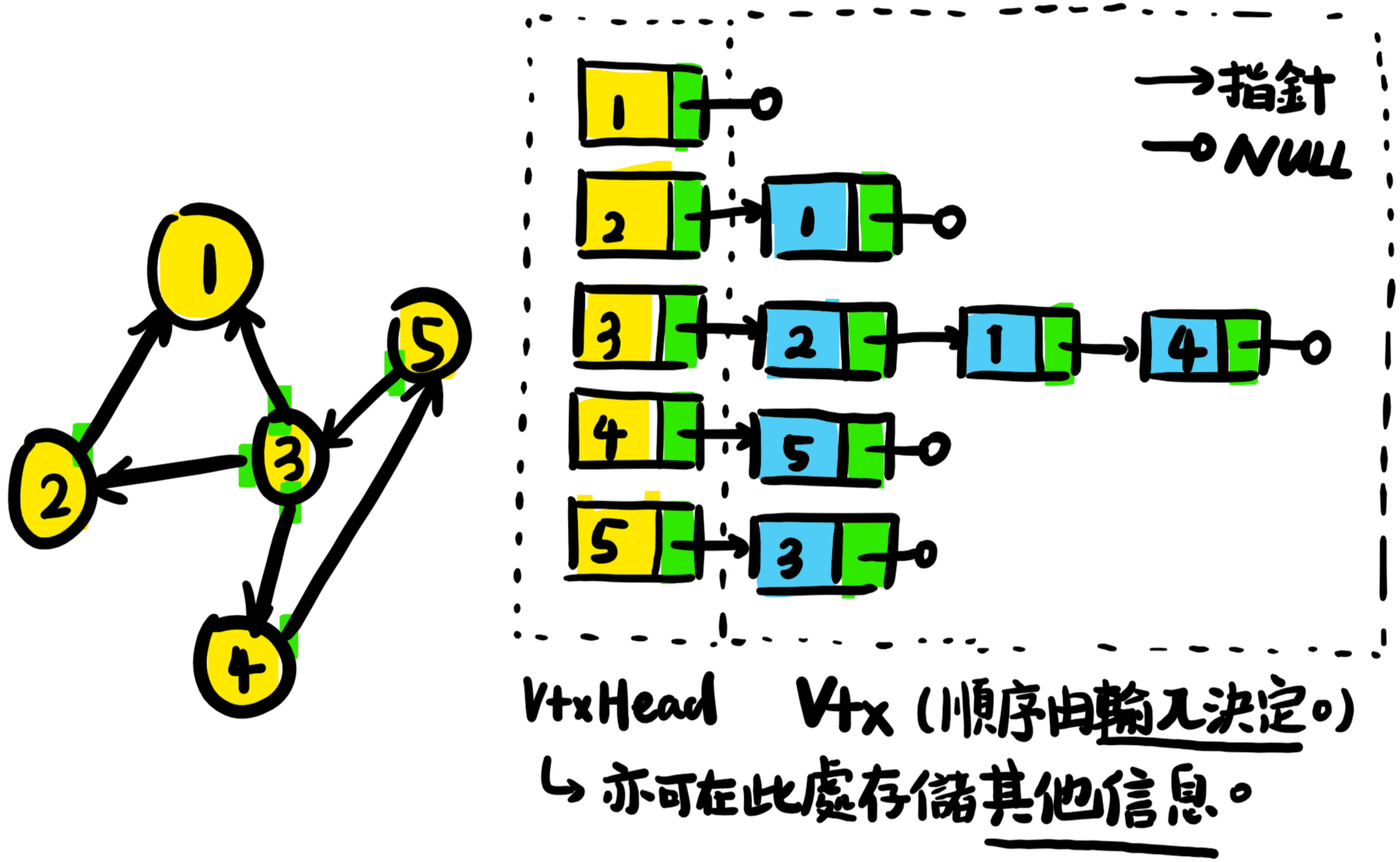
Storing a Graph

第二類是圖是否賦權，對於賦權圖，我們用 `int` 或 `long long` 數組儲存，將未連通的路賦值為 `INT_MAX`／`LLONG_MAX`／`0x7f7f7f7f`／`-1` 等；對於無權圖，使用 `bool` 即可。由於存儲空間較大（$\Theta(|V|^2)$），遍歷（traversal）較困難，因此如果不是稠密（dense）圖，就不常用鄰接矩陣。

2. 圖的表示

Storing a Graph

鄰接表是常用的圖的存儲方式，廣汎適用於稀疏（sparse）圖。在此介紹個人研究的方法。我們在 `VtxHead` 結構體中存儲每一個頂點的起始信息／其他信息，然後向下生長。`Vtx` 結構體成為了一個鏈表，這鏈表一保存 `VtxHead` 所有鄰接的點，生長的方式（`Grow( )`）與鏈表相同。



＊　鄰接表存儲圖圖解　＊

鄰接表存儲圖：托托的實現

Adjacency List for Storing a Graph: tot's Instance

```
struct Vertex {
    int To;
    Vertex *Next;

    Vertex(void) : To(NotAVertex),
Next(NULL) {}
};

struct VtxHead : Vertex {
    … // variables that every vertex
should store
    Vertex *Head;

    void operator += (int To)
    {
        if (Head == NULL) {
            Next = new Vertex();
            Next->To = To;
            Head = Next;
```

```
        } else {
            Next->Next = new Vertex();
            Next = Next->Next;
            Next->To = To;
        }
    }

    VtxHead(void) :
        Head(NULL),
        … ( … ) {}
} Graph[MAXN];
```

3. 歐拉迴路

Eulerian Path

若圖 G 中存在一條路徑，使得它恰好通過 G 中每條邊一次，則稱其為歐拉路徑（Eulerian path）。特別地，若該路徑是一個環路，則稱其為歐拉迴路。

具有歐拉迴路的圖為歐拉圖，只有歐拉路徑的圖為半歐拉圖。

3. 歐拉迴路

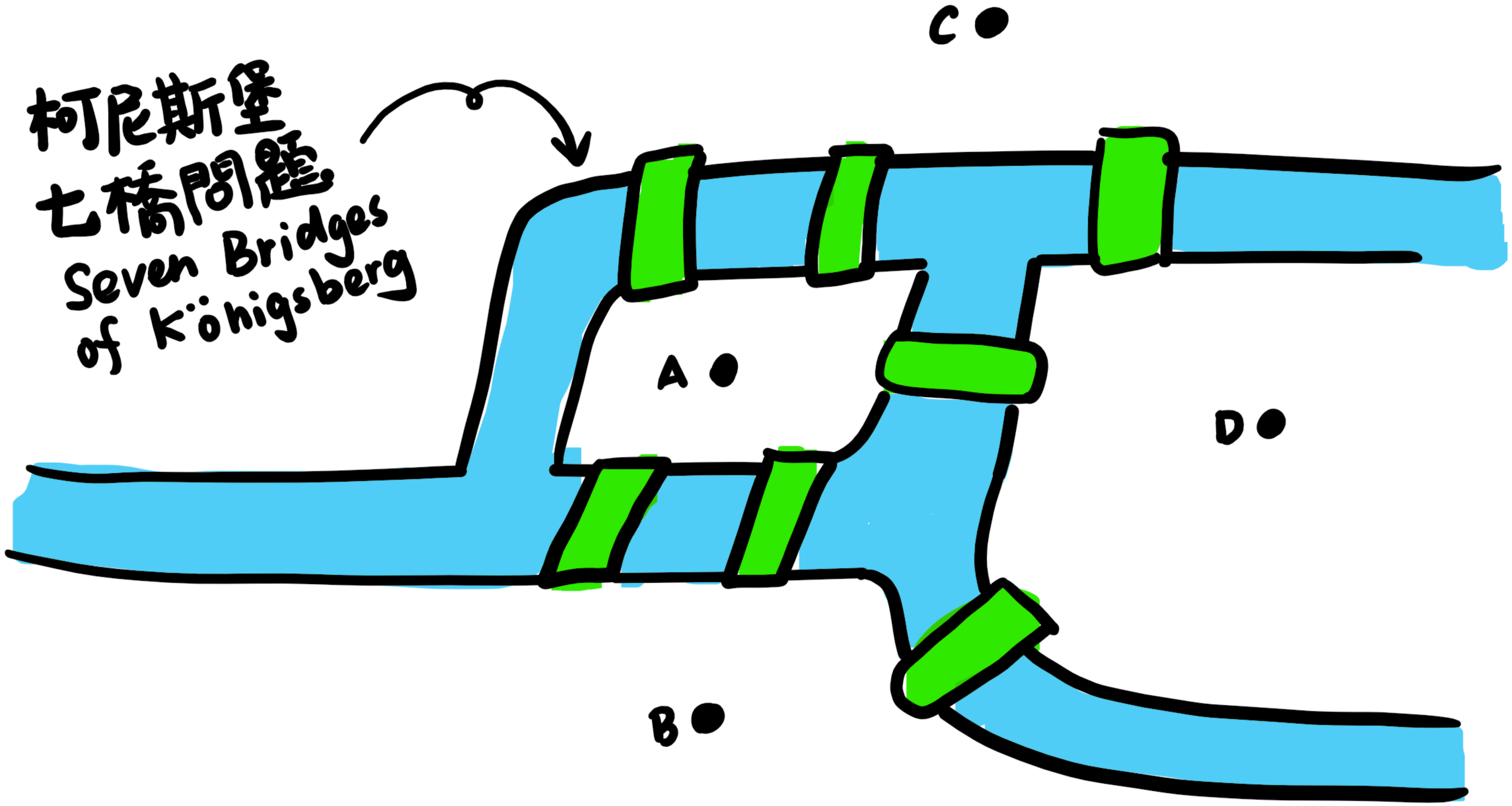
Eulerian Path

對於無向連通圖 G，有如下性質：

1. 存在歐拉路徑 ⇔ 圖 G 中有兩個奇頂點／無奇頂點；

2. 存在歐拉路徑且有兩個奇頂點，斯兩點為圖始終；

3. 存在歐拉路徑且無奇頂點，則為歐拉迴路。



✳　萊昂哈德·歐拉研究的七橋問題就是一個實例　✳

3. 歐拉迴路

Eulerian Path

對於有向連通圖 G，存在歐拉路逕必滿足如下任一條件：

1. 其餘頂點出入度相等，但有一個頂點出入度之差為 1，有另一頂點出入度之差為 −1，斯兩點為圖始終；

2. 所有頂點出入度相等，此時路徑係歐拉迴路。

習題 1：判斷歐拉迴路

Problem 1: Judge An Eulerian Path

請入：http://www.dsalgo.openjudge.cn/graph/7/

歐拉回路是指不令筆離開紙面，可畫過圖中每條邊僅一次，且可以回到起點的一條回路。

給定一個無向圖，請判斷該圖是否存在歐拉迴路。

習題 1：判斷歐拉迴路（程式填空）

Problem 1: Judge An Eulerian Path (Filling in the Blank)

```
bool IsVist[MAXN];
int Cnt;
void Search(int Start)
{

    queue<int> Travel;
    Travel.push(__________);
    IsVist[Start] = true;

    while (!Travel.empty()) {
        int From = Travel.front();
        Travel.pop();

        for (Vtx *i = Graph[From].Head;
            i; __________) {
            if (!__________) {
                IsVist[i->To] = true;
                ++Cnt;
                Travel.push(__________);
            }
        }
    }
}
```

習題 1：判斷歐拉迴路（程式填空）

Problem 1: Judge An Eulerian Path (Filling in the Blank)

```cpp
bool IsVist[MAXN];
int Cnt;
void Search(int Start)
{

    queue<int> Travel;
    Travel.push(Start);
    IsVist[Start] = true;

    while (!Travel.empty()) {
        int From = Travel.front();
        Travel.pop();

        for (Vtx *i = Graph[From].Head;
            i; i = i->Next) {
            if (!IsVist[i->To]) {
                IsVist[i->To] = true;
                ++Cnt;
```

```cpp
                Travel.push(i->To);
            }
        }
    }
}
```

習題 1：判斷歐拉迴路（程式填空）

Problem 1: Judge An Eulerian Path (Filling in the Blank)

```cpp
for (int i = 1; i <= m; ++i) {
    int u, v;
    cin >> u >> v;

    Graph[u] __________;
    Graph[v] += u;

    ++Graph[u].Degree;
    __________;
}

Search(1);

if (__________) {
    cout << "0" << endl;
    return;
}

for (int i = 1; i <= n; ++i)
```

```cpp
    if (Graph[i].Degree __________) {
        cout << "0" << endl;
        return;
    }

cout << __________ << endl;
```

習題 1：判斷歐拉迴路（程式填空）

Problem 1: Judge An Eulerian Path (Filling in the Blank)

```cpp
for (int i = 1; i <= m; ++i) {
    int u, v;
    cin >> u >> v;

    Graph[u] += v;
    Graph[v] += u;

    ++Graph[u].Degree;
    ++Graph[v].Degree;
}

Search(1);

if (Cnt != n) {
    cout << "0" << endl;
    return;
}

for (int i = 1; i <= n; ++i)
```

```cpp
    if (Graph[i].Degree % 2 == 1) {
        cout << "0" << endl;
        return;
    }

cout << "1" << endl;
```

習題 2：單詞拼接

Problem 2: Link Words

（測試樣例在下分文件中，請完成后上交統一測評。）

甲給了乙 n 個單詞，如果一個單詞的最後一個字母和另一個單詞的第一個字母相同，那麼兩個單詞就可以連接在一起組成一個新的單詞。現在甲想要乙計算一下，給定的 n 個單詞是否可以全部連接在一起。

習題 2：單詞拼接

Problem 2: Link Words

輸入格式

第一行輸入一個整數 n，代表一共有 n 個單詞（$1 \leq n \leq 10{,}0000$）。

接下來輸入 n 行，每行輸入一個單詞。單詞均由小寫字母組成，每個單詞長度不超過 20。

輸出格式

輸出一行，如果所有的單詞都可以連接在一起並且可以形成一個環，那麼輸出 `Euler loop`；如果所有單詞都可以連接在一起，但是不會形成環，輸出 `Euler path`；如果所有單詞不能連在一起，那麼輸出 `impossible`。

習題 2：單詞拼接（程式填空）

Problem 2: Link Words (Filling in the Blank)

```
// This is for adjacency matrix…

bool IsVisited[MAXN] = { false };
int Count = 1;
void Search(int Start)
{
    __________ Travel;
    Travel.push(Start);
    IsVisited[Start] = true;

    while (!__________) {
        int From = Travel.__________;
        Travel.pop();

        for (int i = 0; i < 26; ++i) {
            if (IsConnected[From][i] &&
                !IsVisited[i]) {
                IsVisited[i] = __________;
                ++Count;
```

```
                    Travel.push(__________);
                }
            }
        }
    }
```

習題 2：單詞拼接（程式填空）

Problem 2: Link Words (Filling in the Blank)

```
// This is for adjacency matrix…

bool IsVisited[MAXN] = { false };
int Count = 1;
void Search(int Start)
{
    queue<int> Travel;
    Travel.push(Start);
    IsVisited[Start] = true;

    while (!Travel.empty()) {
        int From = Travel.front();
        Travel.pop();

        for (int i = 0; i < 26; ++i) {
            if (IsConnected[From][i] &&
                !IsVisited[i]) {
                IsVisited[i] = true;
                ++Count;
```

```
                    Travel.push(i);
                }
            }
        }
    }
```

習題 2：單詞拼接（程式填空）

Problem 2: Link Words (Filling in the Blank)

```cpp
int a = 0, b = 0, c = 0;
for (int i = 0; i < 26; ++i)
    if (IsAppeared[i])
        if (Indegree[i] - __________ == -1)
            ++a;
        else if (Indegree[i] - Outdegree[i] == __________)
            ++b;
        else if (Indegree[i] == Outdegree[i])
            ++__________;

if (c == n) {
    cout << "Euler loop" << endl;
    return 0;
} else if (a == 1 && b == 1 && c == __________) {
    cout << __________ << endl;
    return 0;
}
```

習題 2：單詞拼接（程式填空）

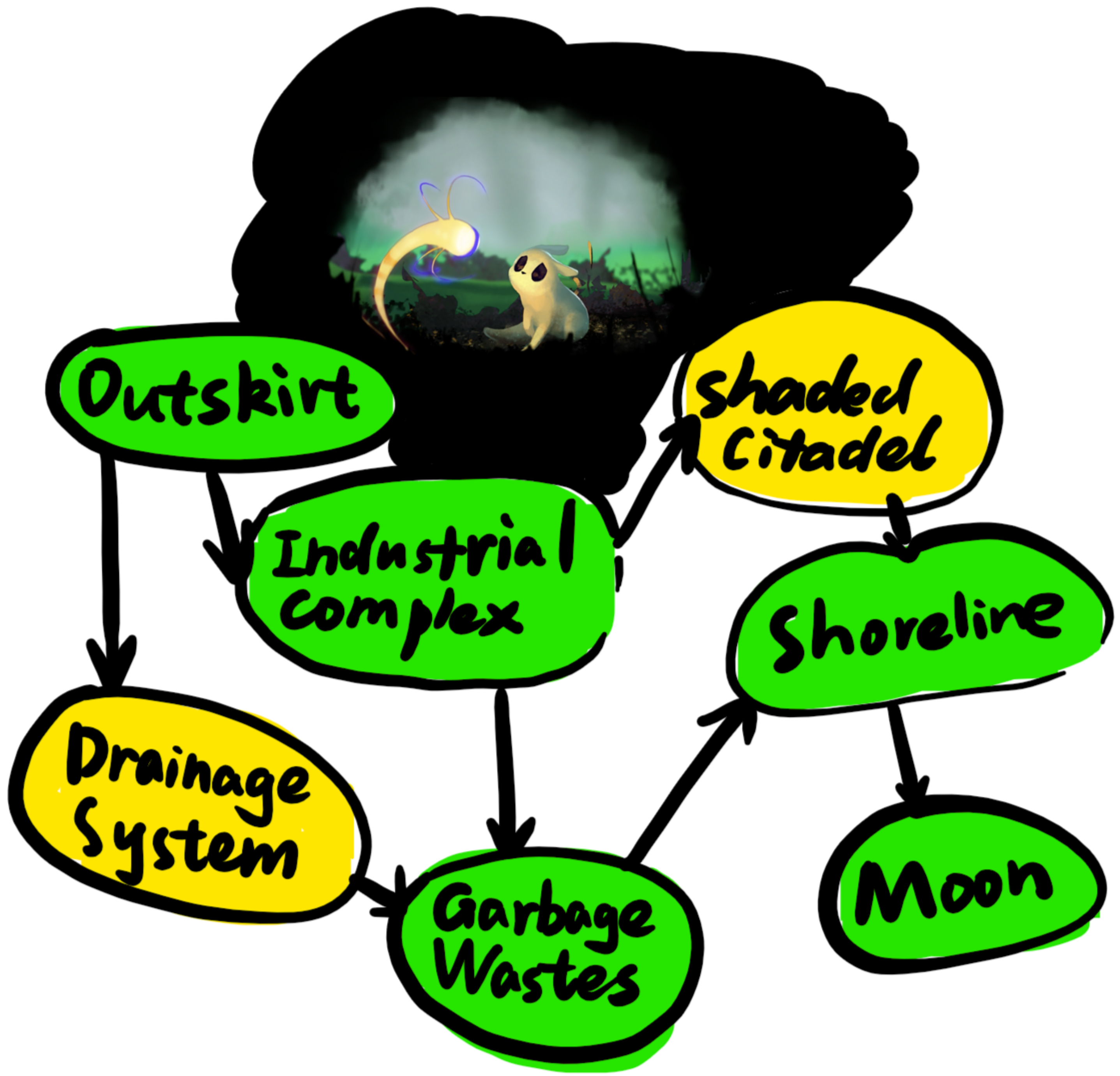
Problem 2: Link Words (Filling in the Blank)

```cpp
int a = 0, b = 0, c = 0;
for (int i = 0; i < 26; ++i)
    if (IsAppeared[i])
        if (Indegree[i] - Outdegree[i] == -1)
            ++a;
        else if (Indegree[i] - Outdegree[i] == 1)
            ++b;
        else if (Indegree[i] == Outdegree[i])
            ++c;

if (c == n) {
    cout << "Euler loop" << endl;
    return 0;
} else if (a == 1 && b == 1 && c == n - 2) {
    cout << "Euler path" << endl;
    return 0;
}
```

4. 拓撲排序

Topological Sorting

拓撲排序的對象是上文提到的有向無環圖（Directed Acyclic Graph，DAG）。它的意思是，對圖 G 中的頂點進行排序，排序的結果是做到：若有一條邊 $(u, v) \in E$，在排序中 u 在 v 之前。排序的結果叫做拓撲序（topological order）。以下的程式能夠達到 $O(|E| + |V|)$。



✳ 《Rain World》Session 1 通關流程——一個 DAG ✳

拓撲排序過程演示

Topological Sorting Process

```cpp
inline void TopSort(void)
{
    queue<int> Travel; int Idx = 1;

    for (int i = 1; i <= VtxAmt; ++i)
        if (InDegree[i] == 0)
            Travel.push(i);

    while (!Travel.empty()) {
        int CrtIdx = Travel.front();
        Travel.pop();
        TopIdx[CrtIdx] = Idx++;

        for (Vtx *i = Graph[CrtIdx].Head;
            i; i = i->Next)
            if (--InDegree[i->To] == 0)
                Travel.push(i->To);
    }
}
```

習題 3：拓撲排序

Problem 3: Topological Sorting

請入：http://bailian.openjudge.cn/practice/4084/

給出一個圖的結構，輸出其拓撲排序序列。要求在同等條件下，編號小的頂點輸出在前。

習題 3：拓撲排序（程式填空）

Problem 3: Topological Sorting (Filling in the Blank)

```
struct Unit { int x;
    bool operator < (const Unit &y) const
    { return x __________ y.x; } };

int v, a;

inline void TopSort(void)
{
    priority_queue<Unit> Travel;

    for (int i = 1; i <= v; ++i)
        if (Graph[i].InDegree == 0)
            Travel.push( Unit { i } );

    while (!Travel.empty()) {
        int CrtIdx = Travel.top().x;
        Travel.pop();
        printf("v%d ", __________);
```

```
          for (Vtx *i = Graph[CrtIdx].Head;
              i; i = i->Next)
              if (--Graph[i->To].__________
== 0)
                  Travel.push( Unit { _______
} );
    }
}
```

習題 3：拓撲排序（程式填空）

Problem 3: Topological Sorting (Filling in the Blank)

```cpp
struct Unit { int x;
    bool operator < (const Unit &y) const
    { return x > y.x; } };

int v, a;

inline void TopSort(void)
{
    priority_queue<Unit> Travel;

    for (int i = 1; i <= v; ++i)
        if (Graph[i].InDegree == 0)
            Travel.push( Unit { i } );

    while (!Travel.empty()) {
        int CrtIdx = Travel.top().x;
        Travel.pop();
        printf("v%d ", CrtIdx);
```

```cpp
        for (Vtx *i = Graph[CrtIdx].Head;
            i; i = i->Next)
            if (--Graph[i->To].InDegree ==
0)
                Travel.push( Unit { i->To
} );
    }
}
```

習題 4：奶牛競賽

Problem 4: Cow Contest

請入：https://www.luogu.org/problemnew/show/P2419

從 1 編號到 N 的 N（$1 \le N \le 100$）頭奶牛正在參加一場程式競賽。我們知道，有些奶牛的碼力總是要強一些。在競爭者中，每一頭奶牛都有一個唯一確定的碼力排名。

現在給出 M（$1 \le M \le 4500$）輪兩兩單挑的結果，它們不會自相矛盾。編號為 A 的奶牛的碼力強於編號為 B 的奶牛（$1 \le A \le N, 1 \le B \le N, A \ne B$）。在她們的單挑中，編號為 A 的奶牛總能獲勝。希望你可以推斷出盡可能多的奶牛的具體碼力排名。

習題 4：奶牛競賽（答案 1）

Problem 4: Cow Contest (Answer 1)

```
int IsVist[MAXN];
inline int Go(VtxHead *G, int x)
{
    int Sum = 0;

    for (Vtx *i = G[x].Head;
        i; i = i->Next)
        if (!IsVist[i->To]) {
            IsVist[i->To] = true;
            Sum += Go(G, i->To);
        }

    return Sum + 1;
}

// …
```

```
int Cnt = 0;
for (int i = 1; i <= n; ++i) {
    memset(IsVist, false, sizeof IsVist);
    int GCnt = Go(Graph, i);
    memset(IsVist, false, sizeof IsVist);
    int AntiGCnt = Go(AntiGraph, i);
    if (GCnt + AntiGCnt == n + 1)
        ++Cnt;
}
```

習題 4：奶牛競賽（答案 2）

Problem 4: Cow Contest (Answer 2)

```cpp
inline void Floyd(void)
{
    for (int i = 1; i <= n; ++i)
    for (int j = 1; j <= n; ++j)
    for (int k = 1; k <= n; ++k)
        if (Graph[i][j] == BLOCK)
            if (Graph[i][k] == GO &&
                Graph[k][j] == GO) {
                Graph[i][j] = GO;
                Graph[j][i] = BACK;
            } else if (Graph[i][k] == BACK
&&
                Graph[k][j] == BACK) {
                Graph[i][j] = BACK;
                Graph[j][i] = GO;
            }
}

// …
```

```cpp
int Cnt = 0;
for (int i = 1; i <= n; ++i) {
    int TmpCnt = 0;
    for (int j = 1; j <= n; ++j)
        if (i != j &&
            Graph[i][j] == BLOCK)
            ++TmpCnt;
    if (TmpCnt == 0)
        ++Cnt;
}
```

5. 最短路徑問題：貝爾曼 – 福特算法

Shortest Path Problem: Bellman–Ford Algorithm

最短路徑是常常涉及的一個問題。顧名思義，就是給定起點和終點，需要求出最短的無權路徑長（unweighted path length）或者是賦權路徑長（weighted path length，即 $\mathtt{Wgt}_1 + \mathtt{Wgt}_2 + \ldots + \mathtt{Wgt}_N$）。

以後者為例，我們跳過經典的 Dijkstra 算法，直接介紹它的變體 SPFA 算法（Bellman–Ford Algorithm）。他們都是貪婪算法（greedy algorithm）。SPFA 算法不難~~，在筆者沒有接觸到這個概念時，愚以為此算法是自己發明的~~。

貝爾曼 – 福特算法演示

Bellman–Ford Algorithm Process

```
void Weighted(int Start)
{
    Graph[Start].Dist = 0;
    queue<int> Travel;
    Travel.push(Start);

    while (!Travel.empty()) {
        int From = Travel.front();
        Travel.pop();

        for (Vtx *i = Graph[From].Head;
            i; i = i->Next)
            if (Graph[From].Dist + i->Wgt <
                Graph[i->To].Dist) {
                Graph[i->To].Dist =
                    Graph[From].Dist + i->Wgt;

                Travel.push(i->To);
            } } }
```

習題 5：最短路計數

Problem 5: Count the Shortest Paths

請入：https://www.luogu.org/problemnew/show/P1144

給出一個 N 個頂點、M 條邊的無向無權圖，頂點編號為 1 - N。問從頂點 1 開始，到其他每個點的最短路有幾條。

習題 5：最短路計數（程式填空）

Problem 5: Count the Shortest Paths (Filling in the Blank)

```cpp
struct VtxHead : Vtx {
    // …
    void operator += (int To)
    {
        if (!Head) {
            Next = new Vtx();
            Next->To = To;
            Head = Next;
        } else {
            Next->Next = new Vtx();
            Next = Next->Next;
            Next->To = To;
        }
    }

    VtxHead(void) :  Head(NULL),
        Dist(INT_MAX), Cnt(0), IsVist(false)
{}
} Graph[MAXN];
```

```cpp
for (Vtx *i = Graph[From].Head;
    i; i = i->Next) {
    if (Graph[From].Dist + 1 <
        Graph[i->To].Dist) {
        Graph[i->To].Dist =
            Graph[From].Dist + 1;
        Graph[i->To].Cnt =
            Graph[From].Cnt;

        if (!Graph[i->To].IsVist) {
            Graph[i->To].IsVist = true;
            Travel.push(i->To);
        }
    } else if (Graph[From].Dist + 1 ==
        Graph[i->To].Dist)
        Graph[i->To].Cnt = (Graph[i->To].Cnt
+
            Graph[From].Cnt) % 100003;
}
```

習題 5：最短路計數（程式填空）

Problem 5: Count the Shortest Paths (Filling in the Blank)

```
struct VtxHead : Vtx {
    // …
    void operator += (int To)
    {
        if (!Head) {
            Next = new Vtx();
            __________ = To;
            Head = Next;
        } else {
            Next->Next = new Vtx();
            Next = __________;
            Next->To = __________;
        }
    }

    VtxHead(void) :  Head(NULL),
        Dist(__________), Cnt(0),
IsVist(false) {}
} Graph[MAXN];
```

```
for (Vtx *i = Graph[From].__________;
    i; i = i->Next) {
    if (Graph[From].Dist + 1 <
        Graph[i->To].Dist) {
        Graph[i->To].Dist =
            __________;
        __________ =
            Graph[From].Cnt;

        if (!Graph[i->To].IsVist) {
            Graph[i->To].IsVist = true;
            Travel.push(i->To);
        }
    } else if (Graph[From].Dist + 1 ==
        Graph[i->To].Dist)
        Graph[i->To].Cnt = (Graph[i->To].Cnt
+
            Graph[From].Cnt) % __________;
}
```

習題 6：奶酪

Problem 6: Cheese

此雖不是一最短路題目，但藉此訓練隊列優化這一技巧。

請入：https://vijos.org/p/2031

一塊奶酪可以視作一個三維坐標系，其中有許多球形的中空。老鼠傑瑞想穿過這些中空從一端來到另一端。希望你能夠告訴她是否能到終點。

習題 6：奶酪（答案）

Problem 6: Cheese (Answer)

```
for (int i = 1; i <= n; ++i) {
    scanf("%lld %lld %lld",
        &Cheese[i].x, &Cheese[i].y,
&Cheese[i].z);
    if (Cheese[i].z + r >= h)
        G[i] += n + 1;
    if (Cheese[i].z <= r)
        G[0] += i;
}

for (int i = 1; i <= n - 1; ++i)
    for (int j = i + 1; j <= n; ++j)
        if (IsConnected(Cheese[i],
Cheese[j])) {
            G[i] += j;
            G[j] += i;
        }
```

```
inline long long Calc(
    const Coord &c_1, const Coord &c_2)
{
    long long xSpan = c_1.x - c_2.x,
        ySpan = c_1.y - c_2.y,
        zSpan = c_1.z - c_2.z;

    return xSpan * xSpan +
        ySpan * ySpan +
        zSpan * zSpan;
}

bool IsConnected(
    const Coord &c_1, const Coord &c_2)
{
    return Calc(c_1, c_2) <= (long long)4 *
r * r;
}
```

習題 6：奶酪（答案）

Problem 6: Cheese (Answer)

```cpp
bool IsVist[MAXN];
void Search(void)
{
    queue<int> Travel;
    Travel.push(0);
    IsVist[0] = true;

    while (!Travel.empty()) {
        int From = Travel.front();
        Travel.pop();

        for (Vtx *i = G[From].Head;
            i; i = i->Next)
            if (!IsVist[i->To]) {
                IsVist[i->To] = true;

                if (i->To == n + 1) return;

                Travel.push(i->To); } } }
```

6. 最小生成樹：Kruskal 算法

Minimum Spanning Tree: Kruskal’s Algorithm

關於樹有兩個定理，一是定點數 N、變數 E 滿足 $E = N - 1$；而是關於有 N 頂點，E 條邊的圖 T 的三個等價命題，在<u>概念辨析題</u>中曾有所涉及：

1. 圖 T 是樹；

2. 圖 T 無圈，且 $E = N - 1$；

3. 圖 T 連通，且 $E = N - 1$。

生成樹是圖去掉若干邊的<u>最小連通子圖</u>。

6. 最小生成樹：Kruskal 算法

Minimum Spanning Tree: Kruskal's Algorithm

對於一個賦權圖，找到權值最小的生成樹就是去求取最小生成樹（Minimum Spanning Tree，MST），我們使用同樣是貪婪算法的 Kruskal 為例。給定圖的邊集 E 和生成樹集 T，且每個點屬於一個獨立集合（$S_1, S_2, \ldots, S_N$），它的算法的過程是：

1. 從小至大排序 E；

2. 從小至大遍歷邊，對於 $(u, v) \in E$，若 u、v 不在同一集合，就合并它們的集合，使用並查集完成；

3. 重複步驟 2，直到僅剩一個集合或 E 遍歷完成。

並查集（不相交集）例程

Disjoint Set Process

```cpp
int Ast[MAXN];
int GetAst(int x)
{
    if (Ast[x] == x) return x;
    return Ast[x] = GetAst(Ast[x]);
}

inline bool Query(int x, int y)
{
    return GetAst(x) == GetAst(y);
}

inline void Merge(int x, int y)
{
    int xAst = GetAst(x),
        yAst = GetAst(y);

    if (xAst == yAst)
        return;
    Ast[yAst] = xAst;
}

inline void Init(int Amount)
{
    for (int i = 0; i < Amount; ++i)
        Ast[i] = i;
}

// more at https://github.com/bufhdy/tot-problem/tree/master/materials/ADT/disjoint
```

最小生成樹：Kruskal 算法演示

Minimum Spanning Tree: Kruskal's Algorithm Process

```cpp
struct Arc {
    int u, v;
    long long Dist;

    bool operator < (const Arc &x) const
    {
        return Dist < x.Dist;
    }
} Graph[MAXN];

// ...
```

```cpp
sort(Graph, Graph + ArcAmt);

long long Sum = 0;
int Set = n;
// the amount of vertices, finally Set ==
1;
for (int i = 0; i < ArcAmt && Set > 1;
++i) {
    int x = Graph[i].u,
        y = Graph[i].v,
        xAst = GetAst(x),
        yAst = GetAst(y);

    if (xAst != yAst) {
        Ast[xAst] = yAst;
        --Set; // Attention!
        Sum += Graph[i].Dist;
    }
}
```

7. 最近公共祖先

Lowest Common Ancestor

最近公共祖先（Lowest Common Ancestor，LCA）是指在一個樹或者有向無環圖中同時擁有 *v* 和 *w* 作為後代的最深的節點。

我們采用倍增的方法對有根樹求取 LCA。

7. 最近公共祖先

Lowest Common Ancestor

首先對數組進行概念辨析。我們用到的數組有 Ast[MAXN][LMT] 和 Dpt[MAXN]。Ast[x][k] 表示節點 x 的 2^k 倍祖先，Dpt[x] 表示節點 x 在有根樹中的深度。

分兩步走。第一步是進行<u>初始化</u>：

1. 設置根節點的深度為 1；
2. 遞歸求得所有節點的深度和父節點（Ast[x][0]）；
3. 逐層遞推，倍增求得所有祖先。

7. 最近公共祖先

Lowest Common Ancestor

第二步是具體地去求 Lca(x, y)：

1. 有正整數 k，使得 $2^k \geq$ Dpt[x]；

2. 讓 x 不斷靠近 y（Dpt[x] - Dpt[y] $\geq 2^i$, i $\in [0, k]$，由大到小遞減）；

3. 若此時 x、y 不在同處，就讓 x、y 手拉手向上走（條件是 Ast[x][i] $\neq$ Ast[y][i], i $\in [0, k]$，由大到小遞減）；

4. 最後返回 Ast[x][0]。

最近公共祖先過程演示

Lowest Common Ancestor Process

```
int Dpt[MAXN], Ast[MAXN][LMT];
void Search(int x)
{
    for (Vtx *i = G[x].Head;
        i; i = i->Next)
        if (!Dpt[i->To]) {
            Dpt[i->To] = Dpt[x] + 1;
            Ast[i->To][0] = x;

            Search(i->To);
        }
}

inline void Init(int x)
{
    Dpt[s] = 1;
    Search(x);

    for (int l = 1; (1 << l) <= n; ++l)
        for (int i = 1; i <= n; ++i)
            Ast[i][l] = Ast[
                Ast[i][l - 1]
            ][l - 1];
}
```

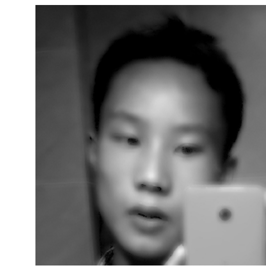
最近公共祖先過程演示

Lowest Common Ancestor Process

```cpp
int Lca(int x, int y)
{
    if (Dpt[x] < Dpt[y])
        swap(x, y);
    int k = 0;
    while ((1 << k) < Dpt[x]) ++k;

    for (int i = k; i >= 0; --i)
        if (Dpt[x] - Dpt[y] >= (1 << i))
            x = Ast[x][i];

    if (x == y) return x;

    for (int i = k; i >= 0; --i)
        if (Ast[x][i] != Ast[y][i]) {
            x = Ast[x][i];
            y = Ast[y][i];
        }
    return Ast[x][0];
}
```

講完了，謝謝大家。

That’s all, thanks.



御薬袋托托

就讀於重慶市第十一中學校，高 2019 級學生
《NOIP 竞赛指北（noip.guide）》作者，其相關的代碼倉庫（code.noip.guide）
〇〇年代折衷媒體《托托的故事（totstories.pw）》作者
《忽冷忽熱怪物 Lukewarm Beast》主播

啁啾會館：@bufhdy　剎那圖鑒：@bufhdy
如來樞紐：@bufhdy　電報：@bufhdy　電綫：@bufhdy
郵箱：bufhdy@hotmail.com