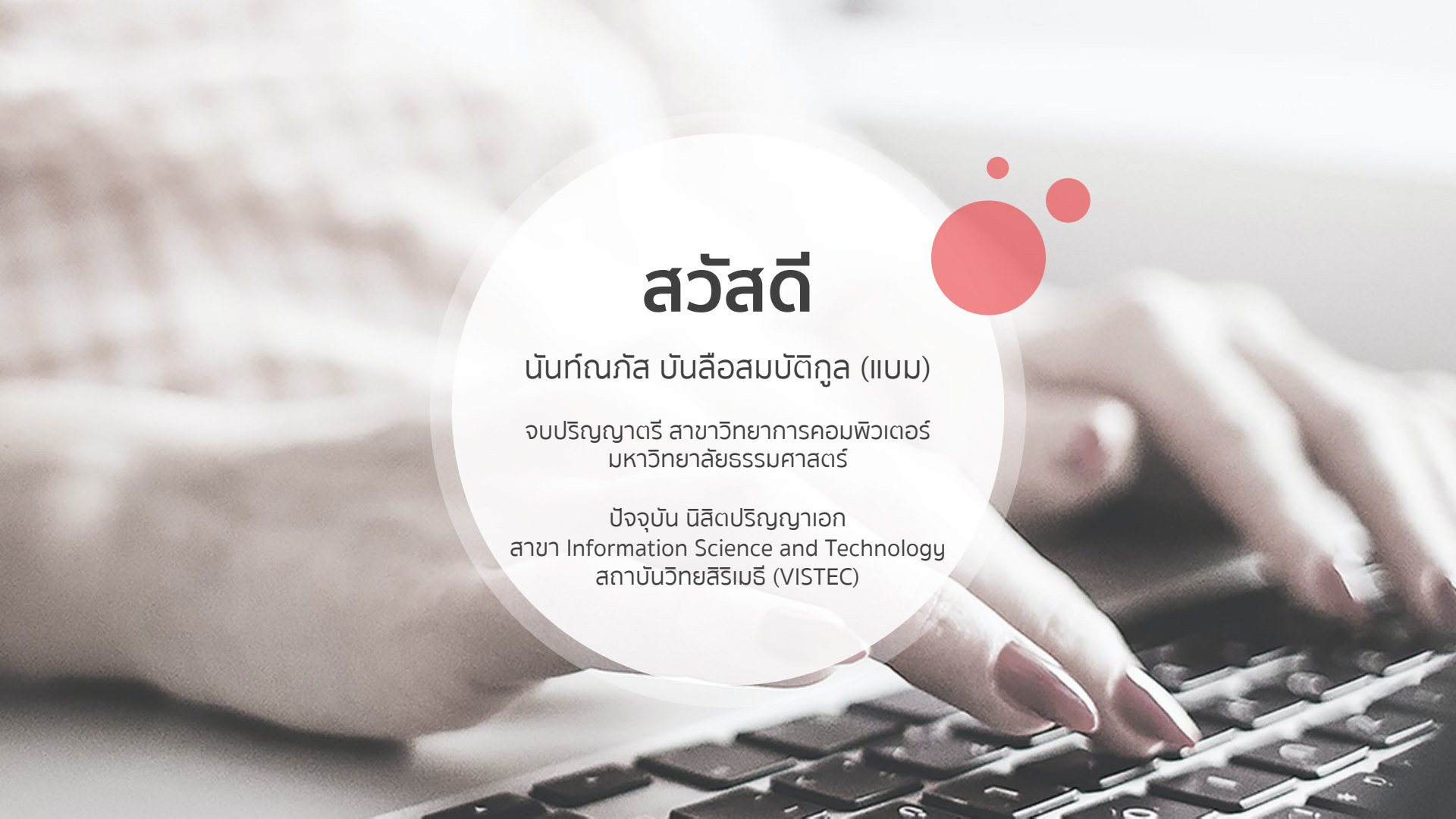
Graph Algorithm

สื่อการสอนสำหรับค่ายคอมพิวเตอร์โอลิมปิก สอวน. ค่าย 2/2562

โดย นางสาว นันท์ณภัส บันลือสมบัติกูล, สถาบันวิทยสิริเมธี (VISTEC)



สวัสดี

นันท์ณภัส บันลือสมบัติกูล (แบม)

จบปริญญาตรี สาขาวิทยาการคอมพิวเตอร์
มหาวิทยาลัยธรรมศาสตร์

ปัจจุบัน นิสิตปริญญาเอก
สาขา Information Science and Technology
สถาบันวิทยสิริเมธี (VISTEC)

All-pair Shortest path

คืออะไร, Algorithm

เช็คชื่อกันเถอะ

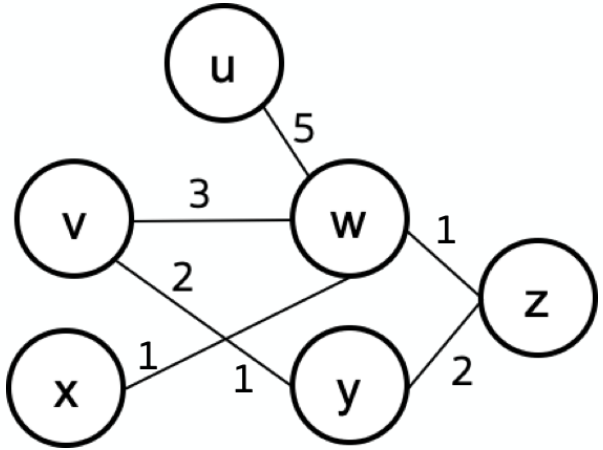
https://forms.gle/xRWnwbJgBxVAXLMy9

All-pair Shortest path of Graph

คืออะไร

- **Shortest Path (แบบแรก)**
  - **มีจุดเริ่มต้น (source) 1 จุด ->** เส้นทางใดสั้นที่สุด

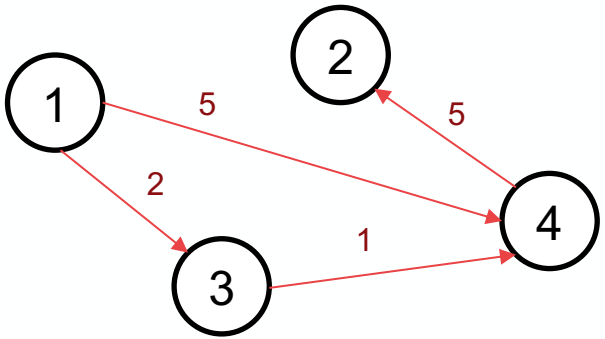
- **All-pair Shortest Path**
  - **ทุก ๆ Node u, v ที่ไปถึงกันได้** -> หาเส้นทางที่ใกล้ที่สุด จาก u ไป v



All-pair Shortest path of Graph

Floyd-Warshall Algorithm

- **Floyd-Warshall Algorithm**
  - ทุก ๆ Node u, v ที่ไปถึงกันได้ -> ระยะทางที่ใกล้ที่สุด จาก u ไป v เป็นเท่าไหร่บ้าง



Node	distance
1, 2	
1, 3	
1, 4	
3, 2	
3, 4	
4, 2	

All-pair Shortest path of Graph

Floyd-Warshall Algorithm

- **ขั้นตอนของ Floyd-Warshall Algorithm**

  “ ในการหาเส้นทางจาก i -> j ถ้ามีเส้นทางที่อ้อมไป (i -> k -> j) แล้วสั้นกว่า จะเก็บไว้เรื่อย ๆ ”

กำหนดให้ input graph G = (V, E)

1. Initialize array **dist** ขนาด V x V

2. **dist** = **G**
   dist[i][j] = 0 if i = j และ dist[i][j] = INF if no (i, j) in E

3. for (k=0; k<V; k++):
       for (i=0; i<V; i++):
           for (j=0; j<V; j++):
               if (**dist[i][k] + dist[k][j] < dist[i][j]**)
                   dist[i][j]) = dist[i][k] + dist[k][j]

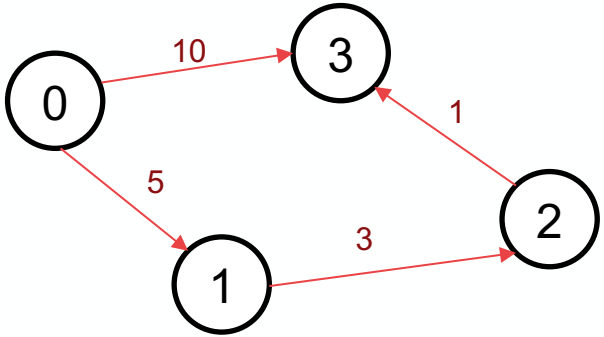
All-pair Shortest path of Graph

Floyd-Warshall Algorithm

กำหนดให้ input graph G = (V, E)

Initialize array **dist** ขนาด V x V

dist = **G**
dist[i][j] = 0 if i = j
dist[i][j] = INF if no (i, j) in E




```
for (k=0; k<V; k++):
    for (i=0; i<V; i++):
        for (j=0; j<V; j++):
            if (dist[i][k] + dist[k][j] < dist[i][j])
                dist[i][j]) = dist[i][k] + dist[k][j]
```

All-pair Shortest path of Graph

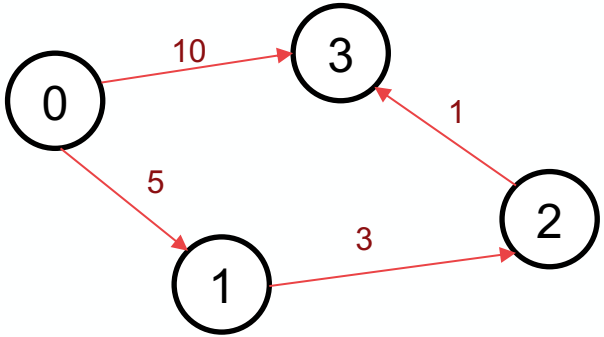
Floyd-Warshall Algorithm

กำหนดให้ input graph G = (V, E)

Initialize array **dist** ขนาด V x V

dist = **G**
dist[i][j] = 0 if i = j
dist[i][j] = INF if no (i, j) in E

```
for (k=0; k<V; k++):
    for (i=0; i<V; i++):
        for (j=0; j<V; j++):
            if (dist[i][k] + dist[k][j] < dist[i][j])
                dist[i][j]) = dist[i][k] + dist[k][j]
```



dist

	0	1	2	3
0				
1				
2				
3				

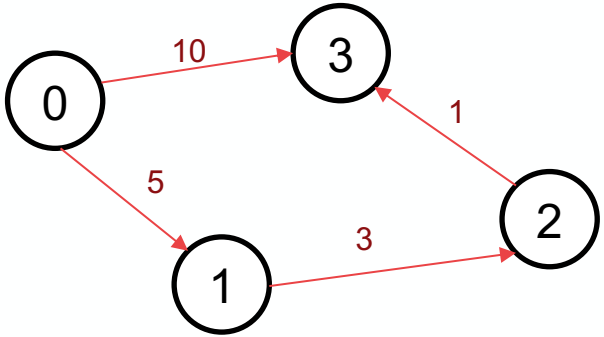
All-pair Shortest path of Graph

Floyd-Warshall Algorithm

กำหนดให้ input graph G = (V, E)

Initialize array **dist** ขนาด V x V

dist = **G**
dist[i][j] = 0 if i = j
dist[i][j] = INF if no (i, j) in E



dist

	0	1	2	3
0				
1				
2				
3				

```
for (k=0; k<V; k++):
    for (i=0; i<V; i++):
        for (j=0; j<V; j++):
            if (dist[i][k] + dist[k][j] < dist[i][j])
                dist[i][j]) = dist[i][k] + dist[k][j]
```

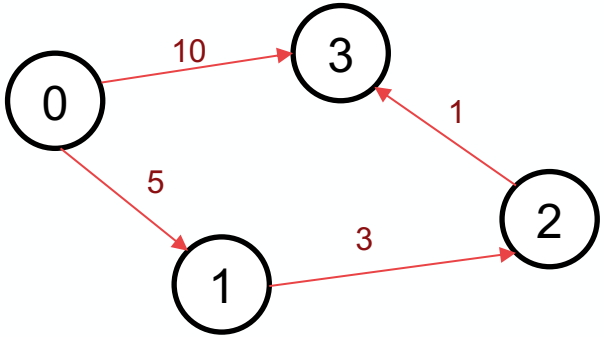
All-pair Shortest path of Graph

Floyd-Warshall Algorithm

กำหนดให้ input graph G = (V, E)

Initialize array **dist** ขนาด V x V

dist = **G**
dist[i][j] = 0 if i = j
dist[i][j] = INF if no (i, j) in E



dist

	0	1	2	3
0	**0**	**5**	**INF**	**10**
1	**INF**	**0**	**3**	**INF**
2	**INF**	**INF**	**0**	**1**
3	**INF**	**INF**	**INF**	**0**

```
for (k=0; k<V; k++):
    for (i=0; i<V; i++):
        for (j=0; j<V; j++):
            if (dist[i][k] + dist[k][j] < dist[i][j])
                dist[i][j]) = dist[i][k] + dist[k][j]
```

All-pair Shortest path of Graph

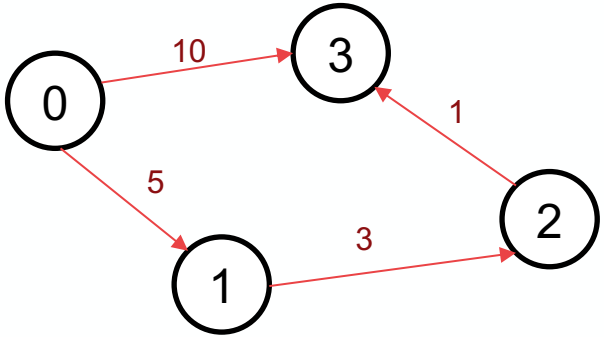
Floyd-Warshall Algorithm

กำหนดให้ input graph G = (V, E)

Initialize array **dist** ขนาด V x V

dist = **G**
dist[i][j] = 0 if i = j
dist[i][j] = INF if no (i, j) in E

```
for (k=0; k<V; k++):
    for (i=0; i<V; i++):
        for (j=0; j<V; j++):
            if (dist[i][k] + dist[k][j] < dist[i][j])
                dist[i][j]) = dist[i][k] + dist[k][j]
```



dist

	0	1	2	3
0	**0**	**5**	**INF**	**10**
1	**INF**	**0**	**3**	**INF**
2	**INF**	**INF**	**0**	**1**
3	**INF**	**INF**	**INF**	**0**

i = 0 **j = 0** **k = 0** →

All-pair Shortest path of Graph

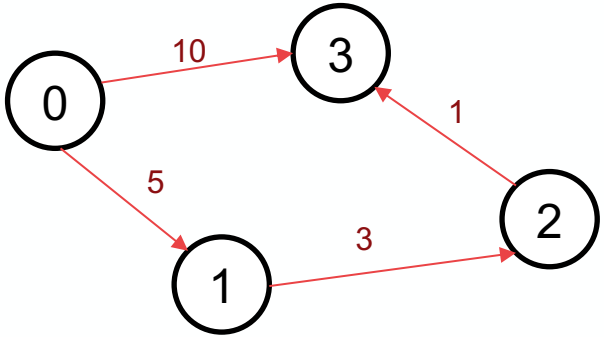
Floyd-Warshall Algorithm

กำหนดให้ input graph G = (V, E)

Initialize array **dist** ขนาด V x V

dist = **G**
dist[i][j] = 0 if i = j
dist[i][j] = INF if no (i, j) in E

```
for (k=0; k<V; k++):
    for (i=0; i<V; i++):
        for (j=0; j<V; j++):
            if (dist[i][k] + dist[k][j] < dist[i][j])
                dist[i][j]) = dist[i][k] + dist[k][j]
```



dist

	0	1	2	3
0	**0**	**5**	**INF**	**10**
1	**INF**	**0**	**3**	**INF**
2	**INF**	**INF**	**0**	**1**
3	**INF**	**INF**	**INF**	**0**

i = 0 **j = 0** **k = 0** → **False**

All-pair Shortest path of Graph

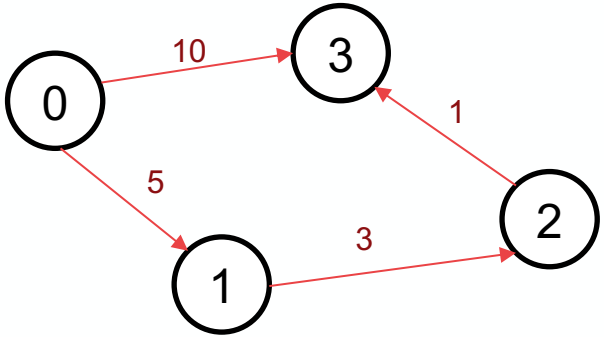
Floyd-Warshall Algorithm

กำหนดให้ input graph G = (V, E)

Initialize array **dist** ขนาด V x V

dist = **G**
dist[i][j] = 0 if i = j
dist[i][j] = INF if no (i, j) in E

```
for (k=0; k<V; k++):
    for (i=0; i<V; i++):
        for (j=0; j<V; j++):
            if (dist[i][k] + dist[k][j] < dist[i][j])
                dist[i][j]) = dist[i][k] + dist[k][j]
```



dist

	0	1	2	3
0	**0**	**5**	**INF**	**10**
1	**INF**	**0**	**3**	**INF**
2	**INF**	**INF**	**0**	**1**
3	**INF**	**INF**	**INF**	**0**

i = 0 **j = 1** **k = 0** →

All-pair Shortest path of Graph

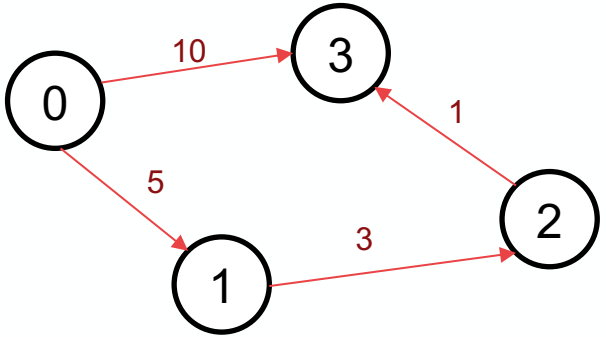
Floyd-Warshall Algorithm

กำหนดให้ input graph G = (V, E)

Initialize array **dist** ขนาด V x V

dist = **G**
dist[i][j] = 0 if i = j
dist[i][j] = INF if no (i, j) in E

```
for (k=0; k<V; k++):
    for (i=0; i<V; i++):
        for (j=0; j<V; j++):
            if (dist[i][k] + dist[k][j] < dist[i][j])
                dist[i][j]) = dist[i][k] + dist[k][j]
```



dist

	0	1	2	3
0	**0**	**5**	**INF**	**10**
1	**INF**	**0**	**3**	**INF**
2	**INF**	**INF**	**0**	**1**
3	**INF**	**INF**	**INF**	**0**

i = 0 **j = 2** **k = 0** →

All-pair Shortest path of Graph

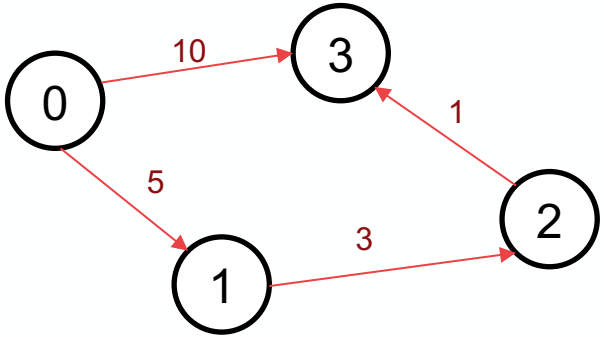
Floyd-Warshall Algorithm

กำหนดให้ input graph G = (V, E)

Initialize array **dist** ขนาด V x V

dist = **G**
dist[i][j] = 0 if i = j
dist[i][j] = INF if no (i, j) in E

```
for (k=0; k<V; k++):
    for (i=0; i<V; i++):
        for (j=0; j<V; j++):
            if (dist[i][k] + dist[k][j] < dist[i][j])
                dist[i][j]) = dist[i][k] + dist[k][j]
```



dist

	0	1	2	3
0	**0**	**5**	**INF**	**10**
1	**INF**	**0**	**3**	**INF**
2	**INF**	**INF**	**0**	**1**
3	**INF**	**INF**	**INF**	**0**

i = 0 **j = 3** **k = 0** →

All-pair Shortest path of Graph

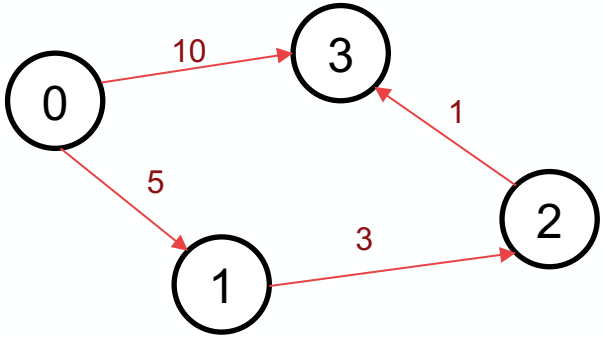
Floyd-Warshall Algorithm

กำหนดให้ input graph G = (V, E)

Initialize array **dist** ขนาด V x V

dist = **G**
dist[i][j] = 0 if i = j
dist[i][j] = INF if no (i, j) in E

```
for (k=0; k<V; k++):
    for (i=0; i<V; i++):
        for (j=0; j<V; j++):
            if (dist[i][k] + dist[k][j] < dist[i][j])
                dist[i][j]) = dist[i][k] + dist[k][j]
```



dist

	0	1	2	3
0	**0**	**5**	**INF**	**10**
1	**INF**	**0**	**3**	**INF**
2	**INF**	**INF**	**0**	**1**
3	**INF**	**INF**	**INF**	**0**

i = 1 **j = 0** **k = 0** →

All-pair Shortest path of Graph

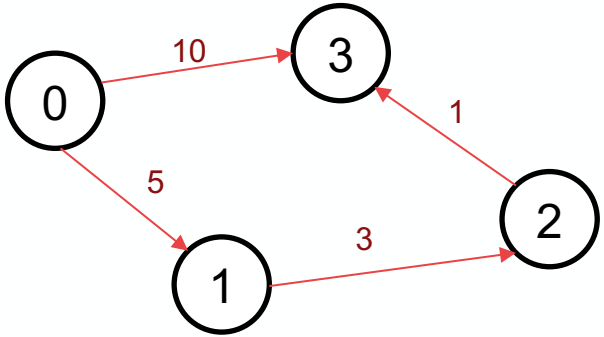
Floyd-Warshall Algorithm

กำหนดให้ input graph G = (V, E)

Initialize array **dist** ขนาด V x V

dist = **G**
dist[i][j] = 0 if i = j
dist[i][j] = INF if no (i, j) in E

```
for (k=0; k<V; k++):
    for (i=0; i<V; i++):
        for (j=0; j<V; j++):
            if (dist[i][k] + dist[k][j] < dist[i][j])
                dist[i][j]) = dist[i][k] + dist[k][j]
```



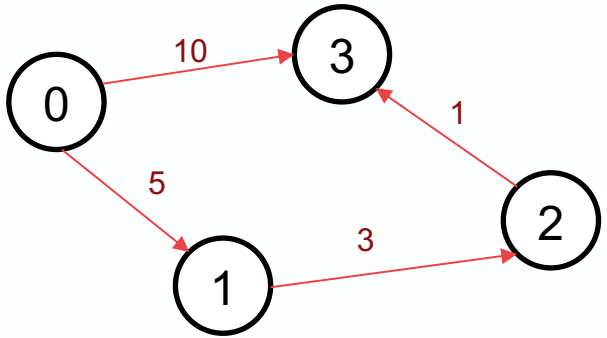
dist

	0	1	2	3
0	**0**	**5**	**INF**	**10**
1	**INF**	**0**	**3**	**INF**
2	**INF**	**INF**	**0**	**1**
3	**INF**	**INF**	**INF**	**0**

i = **j =** **k =** →

All-pair Shortest path of Graph

Floyd-Warshall Algorithm



Let’s code !!

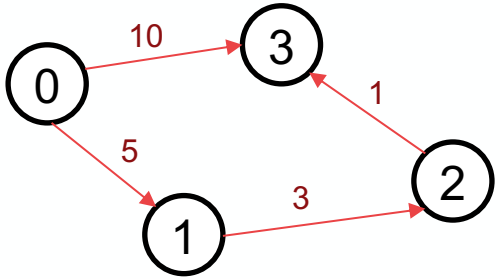
[07-Floyd-Warshall.cpp]

All-pair Shortest path of Graph

Floyd-Warshall Algorithm

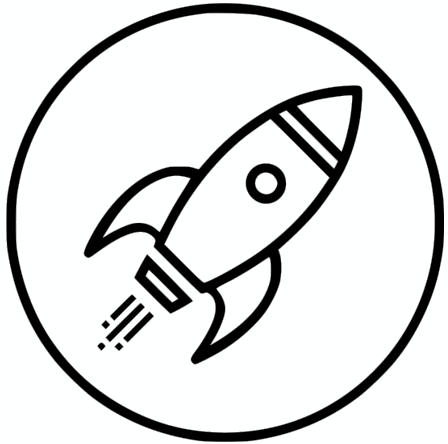
Quiz

1. จาก Floyd-Warshall Algorithm ที่อธิบายเมื่อสักครู่ นักเรียนมีวิธีการในการ**เพิ่มประสิทธิภาพ**ของวิธีนี้อย่างไรบ้าง
2. ต้องแก้ไขโค้ดอย่างไร เพื่อให้สามารถบอก Shortest Path ของแต่ละ Pair ได้



0, 1	0 -> 1	(dist = 5)
0, 2	0 -> 1 -> 2	(dist = 8)
0, 3	0 -> 1 ->2 -> 3	(dist = 9)
1, 2	1 -> 2	(dist = 3)
1, 3	1 -> 2 -> 3	(dist = 4)
2, 3	2 -> 3	(dist = 1)

https://forms.gle/wfyqBJ3gPUNvEgp58



All-pair Shortest path of Graph

Floyd-Warshall Algorithm

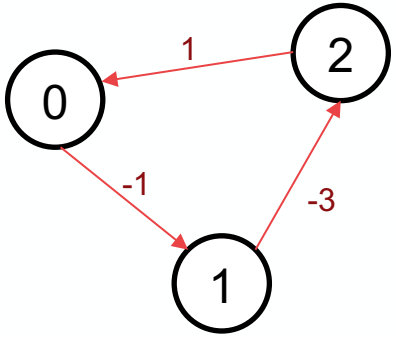
- **Dijkstra's Algorithm vs. Floyd-Warshall Algorithm**

Dijkstra's	Floyd-Warshall
Shortest Path (Single source)	All-pair Shortest Path
O(ElogV) → all nodes = $O(V^3 logV)$	$O(V^3)$
Not Support negative weights	Support negative weights (if no negative weight cycle detected)

All-pair Shortest path of Graph

Floyd-Warshall Algorithm

- How to detect **negative weight cycle** using **Floyd-Warshall Algorithm**
  - ปกติแล้ว dist[i][j] = 0 เสมอ เมื่อ i == j
  - ถ้าพบว่า **dist[i][j] < 0** แสดงว่าพบ **negative weight cycle**



Sum of edges in this cycle = -1 + -3 + 1 = -3
(จึงเป็น Negative Weight Cycle)

dist

	0	1	2
0	**0**	**-1**	**INF**
1	**INF**	**0**	**-3**
2	**1**	**INF**	**0**

i = **j =** **k =**

if (**dist[i][k] + dist[k][j] < dist[i][j]**)
dist[i][j]) = dist[i][k] + dist[k][j]

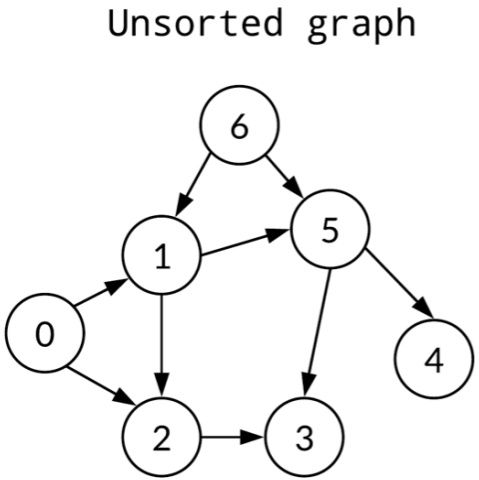
Topological sort

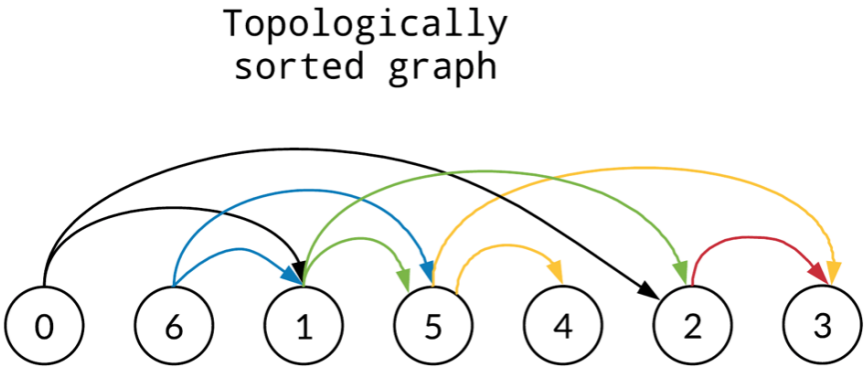
คืออะไร, Algorithm

Topological sort of Graph

คืออะไร, Algorithm

- จาก Directed Acyclic Graph (DAG) G = (V, E)
- Directed Acyclic Graph คือ กราฟแบบมีทิศทาง ที่ไม่มี cycle อยู่ข้างใน
- **Topological sort of G คือ ลำดับของ Vertex ที่ทุก ๆ edge(u, v) u จะมาก่อน v เสมอ**

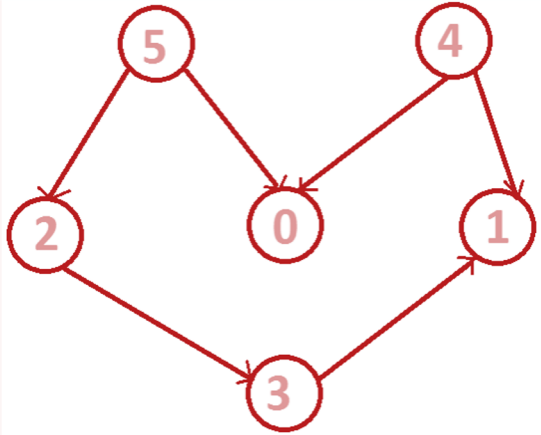




Topological sort of Graph

คืออะไร, Algorithm

- ข้อใดเป็น **Topological sort** ของกราฟด้านล่างบ้าง?

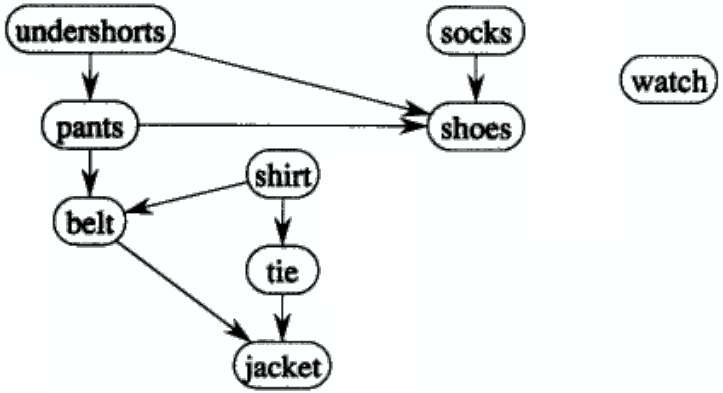


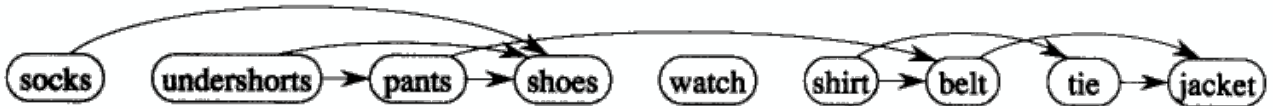
- ☐ 5 4 2 3 1 0
- ☐ 4 5 2 3 1 0
- ☐ 5 2 3 1 4 0
- ☐ 4 2 3 1 5 0

Topological sort of Graph

คืออะไร, Algorithm

- ประโยชน์

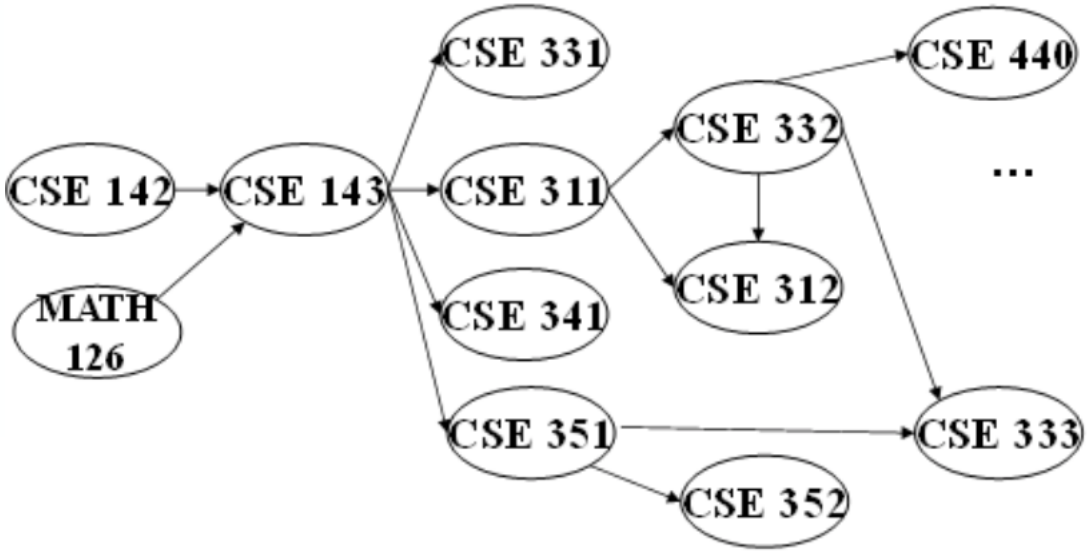




Topological sort of Graph

คืออะไร, Algorithm

- ประโยชน์



Topological sort of Graph

คืออะไร, Algorithm

- วิธีหา **Topological sort** “ประยุกต์จาก DFS แต่เพิ่ม stack เพื่อเก็บ node ที่เป็นลำดับก่อนหน้าให้ออกมาก่อน ”

DFS:

```
initialize all in visited[V] = false

//call recursive function
DFS_util(source, visited);
```

DFS_util(v, visited):

```
show v;
mark visited[v] = true;
For each node in adjacency(v):
    if visited[node] == false:
        DFS(node, visited);
```

TopologicalSort:

```
initialize all in visited[V] = false
empty stack S

//call recursive function for each Vertex v
for each v:
    if visited[v] != true
        TopologicalSortUtil(v, visited, S)

// print from stack
```

TopologicalSortUtil(v, visited):

```
visited[v] = true // ** Not print yet
For each node in adjacency(v):
    if visited[node] == false:
        TopologicalSortUtil(node, visited, S)

push v to stack S
```

Topological sort of Graph

คืออะไร, Algorithm

TopologicalSort:

initialize all in **visited[V]** = false
empty stack **S**

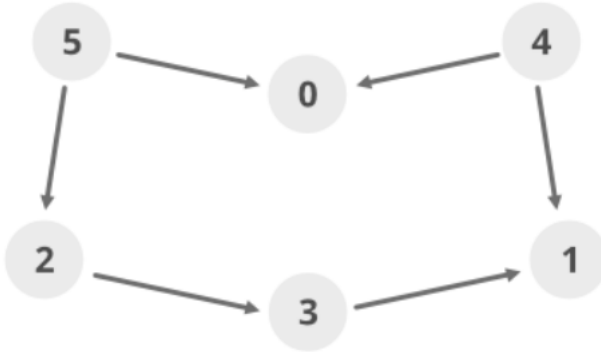
//call recursive function for each Vertex v
for each v:
    if visited[v] != true
        TopologicalSortUtil(v, visited, S)

// print from stack

TopologicalSortUtil(v, visited):

visited[v] = true
for each **node** in adjacency(**v**):
    if visited[**node**] == false:
        TopologicalSortUtil(**node**, **visited, S**)

push **v** to stack **S**



Adja cent list (G)

0 →
1 →
2 → 3
3 → 1
4 → 0, 1
5 → 2, 0

	0	1	2	3	4	5
visited =	F	F	F	F	F	F

stack S = [ ]


Thank you
สู้ๆ นะคะทุกคน