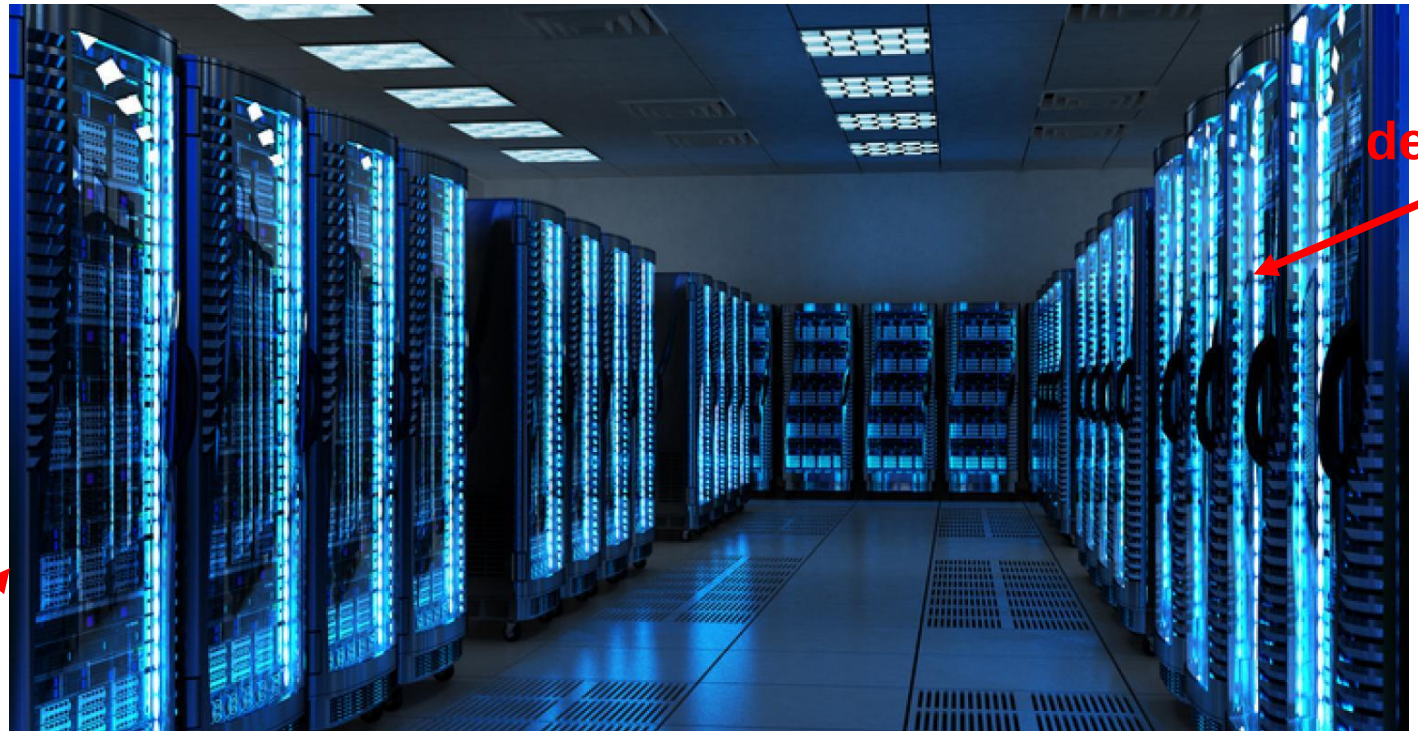
Object-Oriented Programming
Programming Project #1

郭建志

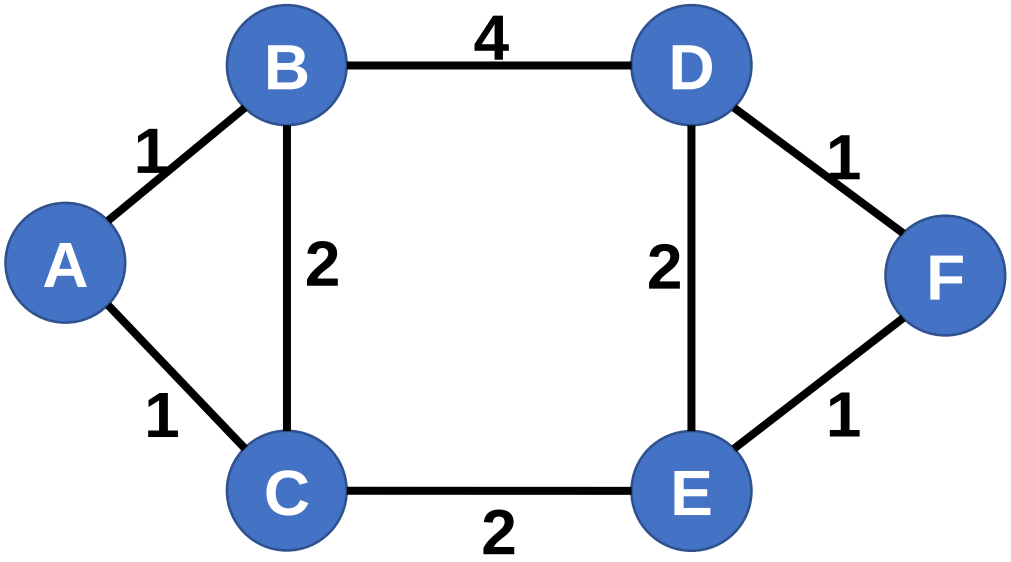
Background

- Local area network within a data center
- Routing efficiency is critical to system performance and quality of service



The Routing Problem

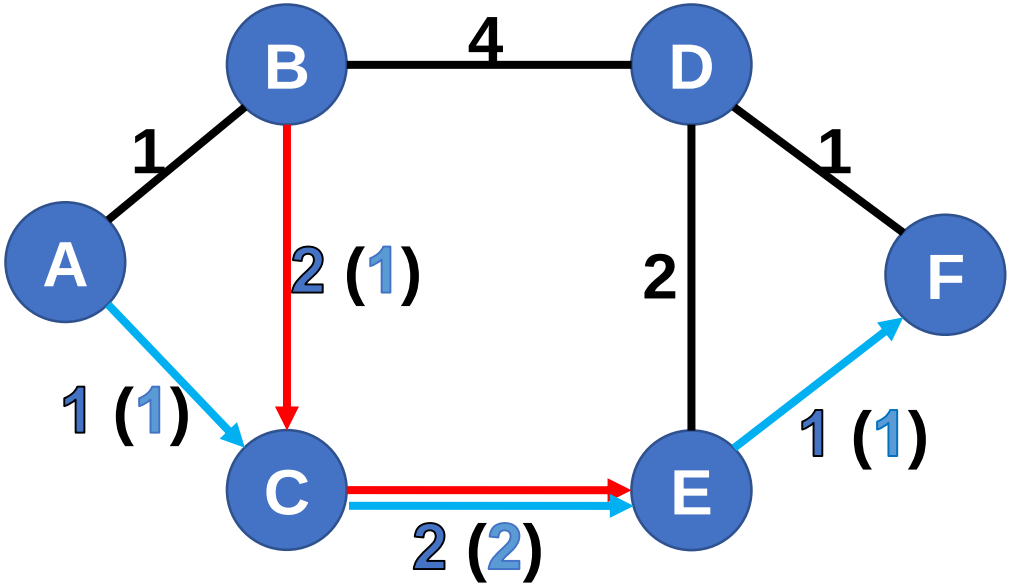
- Consider a scenario, where each request has a different size:



Requests
(src, dest; demand):

(A, F; 1)
(A, F; 2)
(B, E; 1)
(B, E; 3)
(B, E; 4)
(B, C; 1)
(B, D; 3)
(A, B; 4)
(A, C; 2)
(D, F; 4)
(E, F; 5)
(B, C; 6)
(B, D; 2)
(B, D; 1)
(B, D; 2)
(C, E; 3)
(C, E; 1)

One Routing Solution

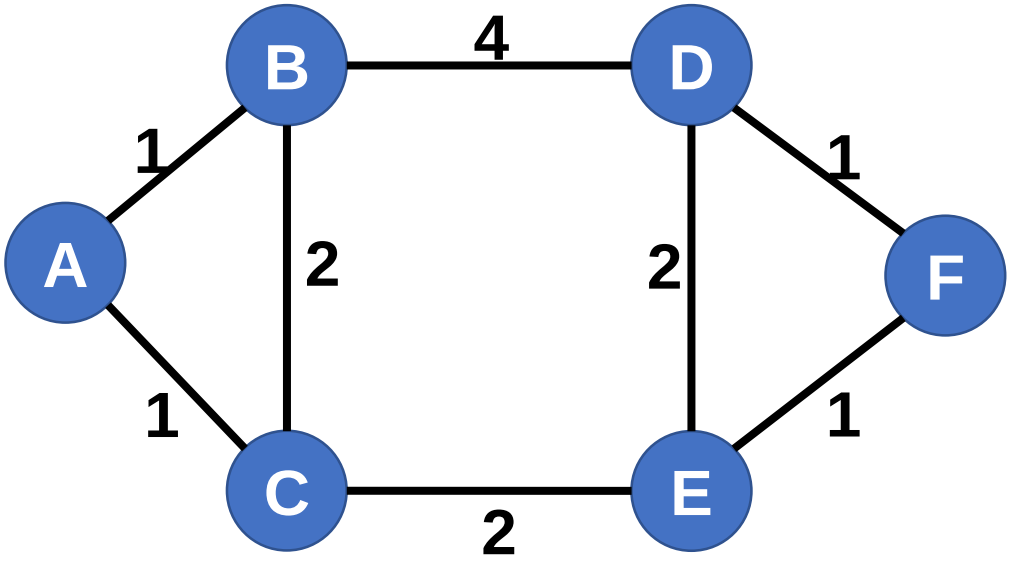


Requests
(src, dest; demand):

- **(A, F; 1)**
- **(A, F; 2)**
- **(B, E; 1)**
- **(B, E; 3)**
- **(B, E; 4)**
- **(B, C; 1)**
- **(B, D; 3)**
- **(A, B; 4)**
- **(A, C; 2)**
- **(D, F; 4)**
- **(E, F; 5)**
- **(B, C; 6)**
- **(B, D; 2)**
- **(B, D; 1)**
- **(B, D; 2)**
- **(C, E; 3)**
- **(C, E; 1)**

The Online Routing Problem

- The requests are arrived in an online fashion.



Requests
(src, dest; demand):

(A, F; 1)
(A, F; 2)
(B, E; 1)
(B, E; 3)
(B, E; 4)
(B, C; 1)
(B, D; 3)
(A, B; 4)
(A, C; 2)
(D, F; 4)
(E, F; 5)
(B, C; 6)
(B, D; 2)
(B, D; 1)
(B, D; 2)
(C, E; 3)
(C, E; 1)

Programming Project #1: Accept or reject a flow

- Input:
  - Numbers of nodes and undirected links
  - Undirected links with non-negative link capacities
  - Requested flows with their sources, destinations, and sizes
- Procedure:
  - Accept or reject the flow one by one
  - If accepted, then assign a path
- Output:
  - Number of accepted flows
  - Total throughput
  - The assigned paths of each accepted flow
- The grade is proportional to the total throughput

Input file:

```
5    10
0    0    1    14
1    0    2    12
2    0    3    9
…
0    0    1    8
1    2    1    6
…
```

Output file:

```
15        124
0         0         1
1         2         0         1
…
```

Programming Project #1: Accept or reject a flow

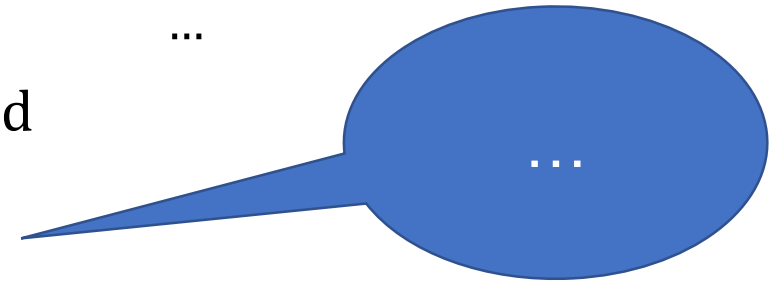
- Input:
  - Numbers of nodes and undirected links
  - Undirected links with non-negative link capacities
  - Requested flows with their sources, destinations, and sizes
- Procedure:
  - Accept or reject the flow one by one
  - If accepted, then assign a path
- Output:
  - Number of accepted flows
  - Total throughput
  - The assigned paths of each accepted flow
- The grade is proportional to the total throughput

Input file:

```
5    10
0    0    1    14
1    0    2    12
2    0    3    9
…
0    0    1    8
1    2    1    6
…
```

Output file:

```
15        124
0         0         1
1         2         0         1
…
```



Programming Project #1: Accept or reject a flow

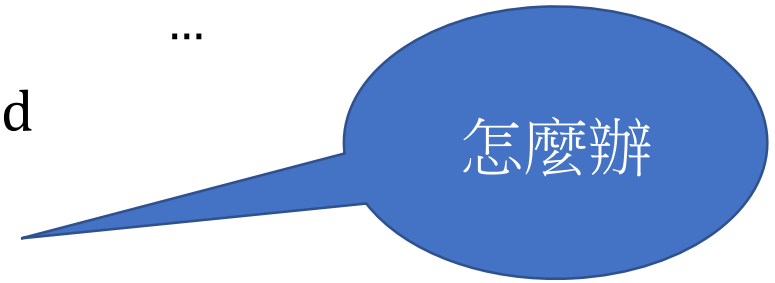
- Input:
  - Numbers of nodes and undirected links
  - Undirected links with non-negative link capacities
  - Requested flows with their sources, destinations, and sizes
- Procedure:
  - Accept or reject the flow one by one
  - If accepted, then assign a path
- Output:
  - Number of accepted flows
  - Total throughput
  - The assigned paths of each accepted flow
- The grade is proportional to the total throughput

Input file:

```
5    10
0    0    1    14
1    0    2    12
2    0    3    9
…
0    0    1    8
1    2    1    6
…
```

Output file:

```
15        124
0         0         1
1         2         0         1
…
```



Programming Project #1: Accept or reject a flow

- Input:
  - Numbers of nodes and undirected links
  - Undirected links with non-negative link capacities
  - Requested flows with their sources, destinations, and sizes
- Procedure:
  - Accept or reject the flow one by one
  - If accepted, then assign a path
- Output:
  - Number of accepted flows
  - Total throughput
  - The assigned paths of each accepted flow
- The grade is proportional to the total throughput

Input file:

```
5    10
0    0    1    14
1    0    2    12
2    0    3    9
…
0    0    1    8
1    2    1    6
…
```

Output file:

```
15       124
0        0        1
1        2        0        1
…
```

Programming Project #1: Accept or reject a flow

- Input:
  - Numbers of nodes and undirected links
  - Undirected links with non-negative link capacities
  - Requested flows with their sources, destinations, and sizes
- Procedure:
  - Accept or reject the flow one by one
  - If accepted, then assign a path
- Output:
  - Number of accepted flows
  - Total throughput
  - The assigned paths of each accepted flow
- Implement a designated algorithm to select the flows

Input file:

```
5    10
0    0    1    14
1    0    2    12
2    0    3    9
…
0    0    1    8
1    2    1    6
…
```

Output file:

```
15        124
0         0         1
1         2         0         1
…
```

Programming Project #1: Accept or reject a flow

- **Designated Algorithm:**
  1. Initially, $cost(e) =$ the minimum of double for each edge e
  2. Examine the current requested flow f with size $size(f)$ and find the shortest path P (i.e., the path with the smallest sum of edge costs)
  3. If all edges on the path P can accommodate this flow (i.e., $load(e) + size(f) \leq capacity(e)$), then accept the flow f; otherwise, reject it and go to step 5
  4. After accepting a flow, update $load(e)\ +\!= size(f)$ and then $cost(e) = \frac{load(e)}{capacity(e) - load(e)}$ for each edge $e \in P$
  5. Go to step 2 and examine the next requested flow until no next one exists

Input file:

```
5    10
0    0    1    14
1    0    2    12
2    0    3    9
...
0    0    1    8
1    2    1    6
...
```

Output file:

```
15        124
0         0         1
1         2         0         1
...
```

Discussion

- Why does the algorithm work?
- It chooses the path that avoids the link with a heavy load

- What can be added to improve the performance?
- Discussion & bonus

Discussion

- Why does the algorithm work?
- It chooses the path that avoids the link with a heavy load

- What can be added to improve the performance?
- Discussion & bonus
- Design a new edge cost formula?
- Add an acceptance condition?

Further Reading

- Section 9.1 in “The Design of Competitive Online Algorithm via a Primal-Dual Approach”, Foundations and Trends in Theoretical Computer Science, 2009.
- Dynamic Routing for Network Throughput Maximization in Software-Defined Networks, in INFOCOM 2016
- …
- 有興趣可以找我要paper
- 我猜沒有人XD

Input Sample: request.txt

Format:

#nodes #undirectedLinks
#linkID #firstNode #secondNode #linkCapacity
…
#requestFlows
#flowID #sourceID #destinationID #flowSize
…

下次揭曉，先自己設計input

Output Sample: result.txt

Format:

#acceptedFlows totalThroughput
#flowID #firstNode #secondNode … #lastNode

下次揭曉，用自己設計input產生答案

Input Sample: request.txt

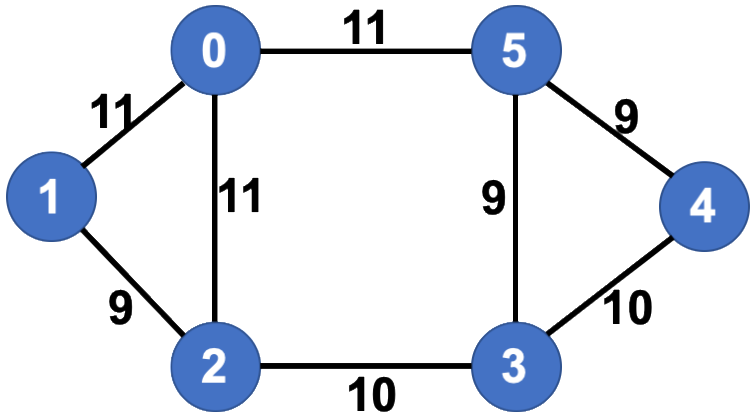
Format:

#nodes #undirectedLinks
#linkID #firstNode #secondNode #linkCapacity
...
#requestFlows
#flowID #sourceID #destinationID #flowSize
...

Format:

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

10			
0	2	4	4
1	2	3	5
2	4	5	6
3	4	1	4
4	0	3	4
5	4	2	7
6	5	0	4
7	3	0	5
8	2	1	4
9	5	2	4



Output Sample: result.txt

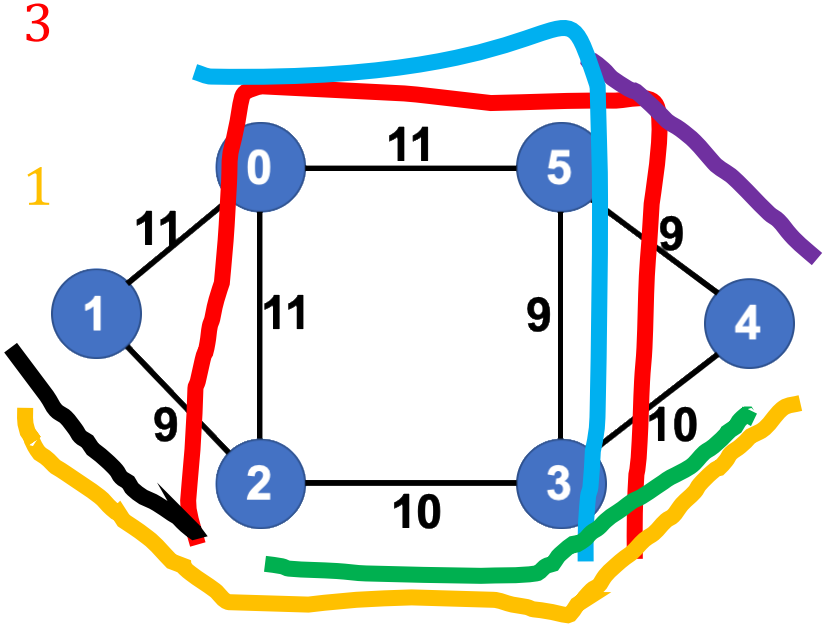
Format:

```
6    27
0    2    3    4
1    2    0    5    3
2    4    5
3    4    3    2    1
4    0    5    3
8    2    1
```

Format:

#acceptedFlows totalThroughput
#flowID #firstNode #secondNode ... #lastNode

```
10
0    2    4    4
1    2    3    5
2    4    5    6
3    4    1    4
4    0    3    4
5    4    2    7
6    5    0    4
7    3    0    5
8    2    1    4
9    5    2    4
```



Note

- Deadline:
  3/26 Tue

- E-course

- **C++ Source code**

- Show a good
  programming style