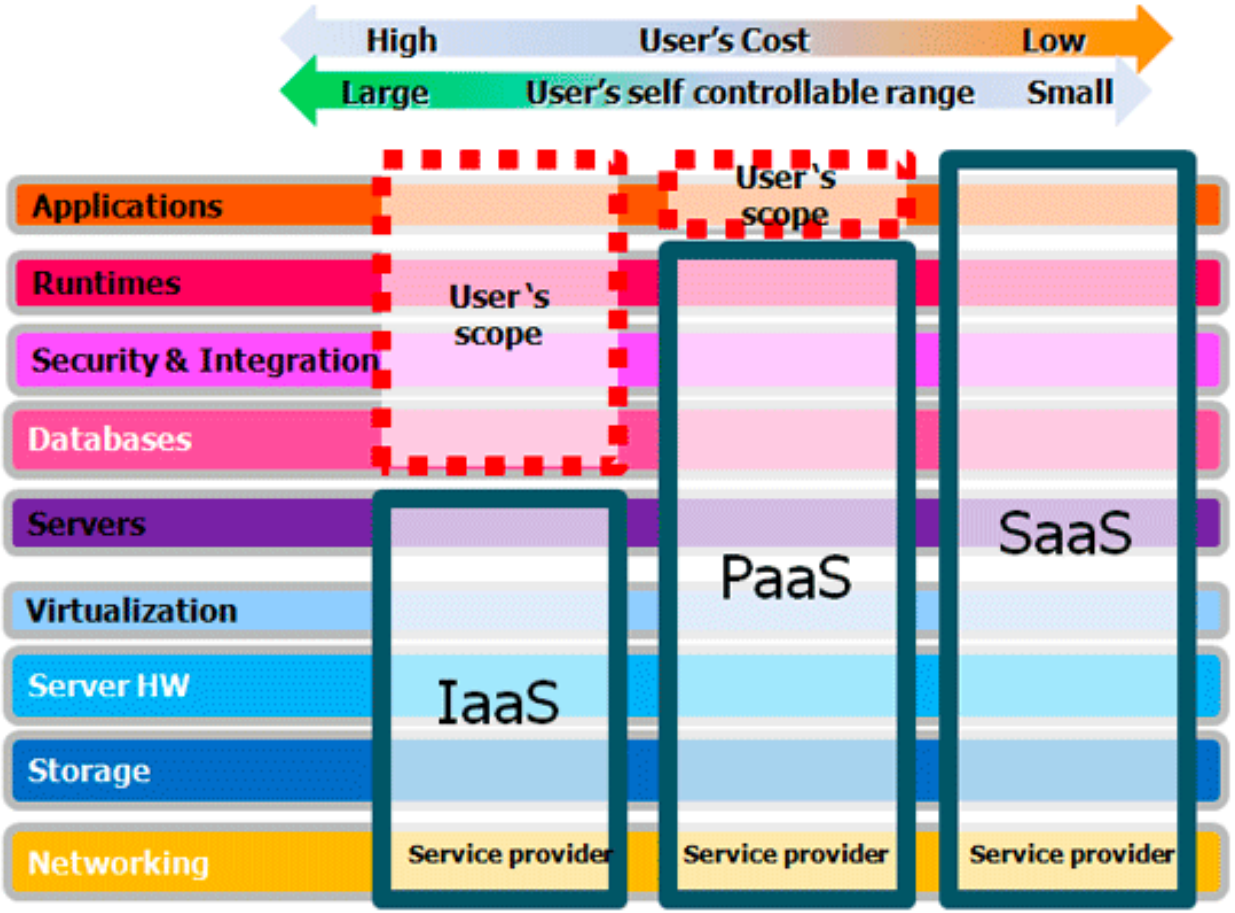
# Object-Oriented Programming
# Programming Project #2

郭建志

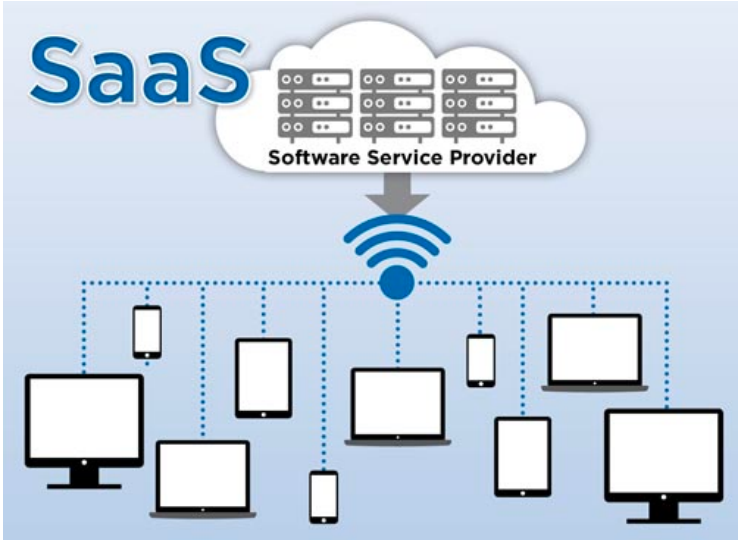
# Background

- Cloud services can be broadly divided into main three models
- Infrastructure-as-a-Service (IaaS)
- Platform-as-a-Service (PaaS)
- Software-as-a-Service (SaaS)



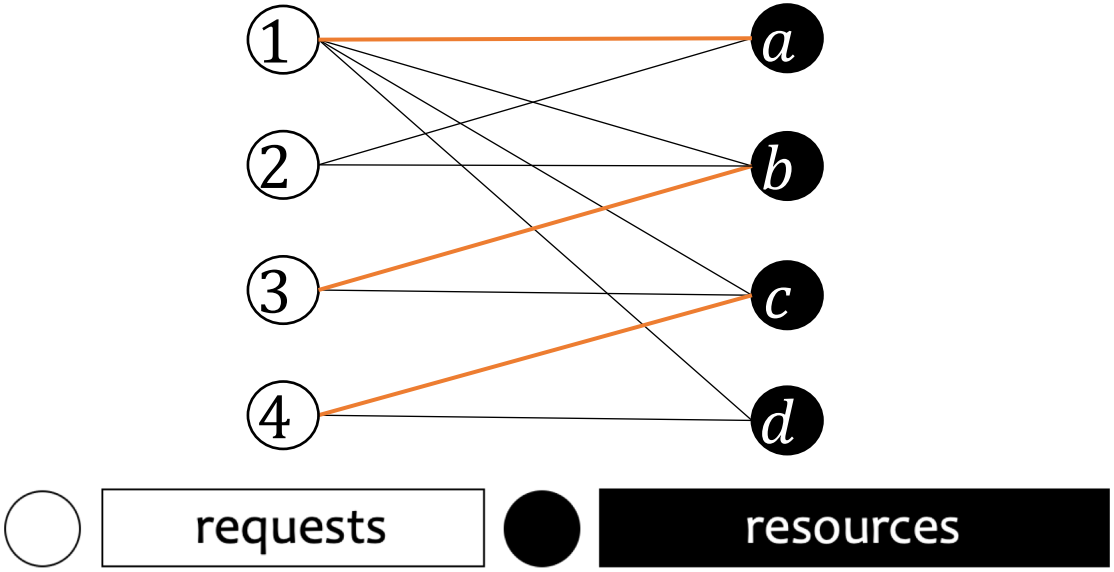
# Background

- Service providers have resources (such as virtual infrastructures, virtual platform, service software) and users request the resources for services
- For example, Google constructs data centers and maintain the hardware and software to provide the mail service for users (i.e., Gmail)



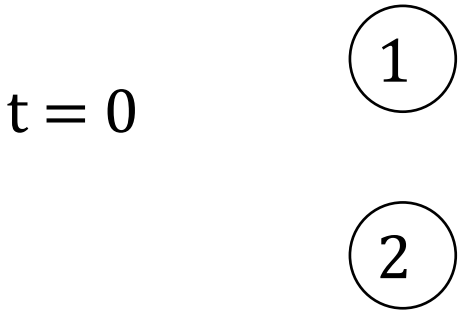
# Background

- Resource allocation can be modeled as a bipartite matching problem
- But resource allocation is usually dynamic
- Some request arrives and some resources are released at a time



# Online Bipartite Matching Problem (BMP)

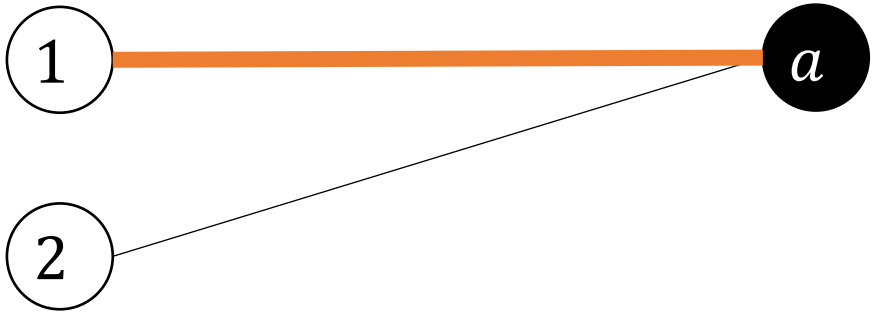
- Consider a scenario, where some request arrives and some resources are released at a time:




# Online Bipartite Matching Problem (BMP)

- Consider a scenario, where some request arrives and some resources are released at a time:

t = 1



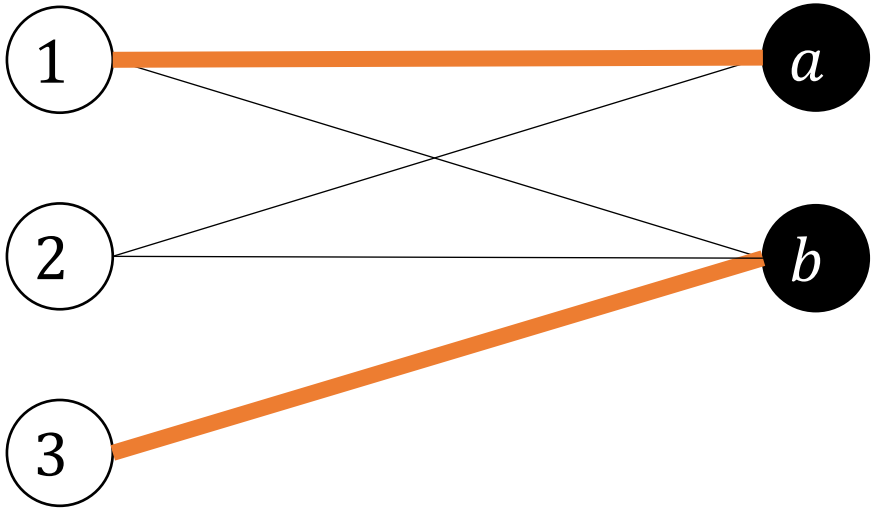

requests

resources

# Online Bipartite Matching Problem (BMP)

- Consider a scenario, where some request arrives and some resources are released at a time:

t = 2



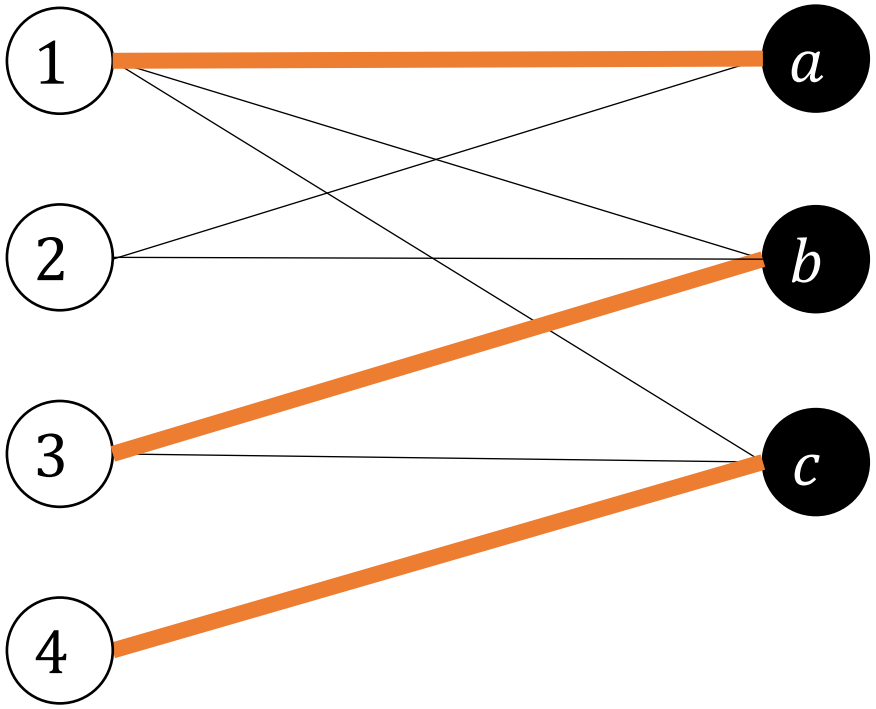

requests

resources

# Online Bipartite Matching Problem (BMP)

- Consider a scenario, where some request arrives and some resources are released at a time:

t = 3

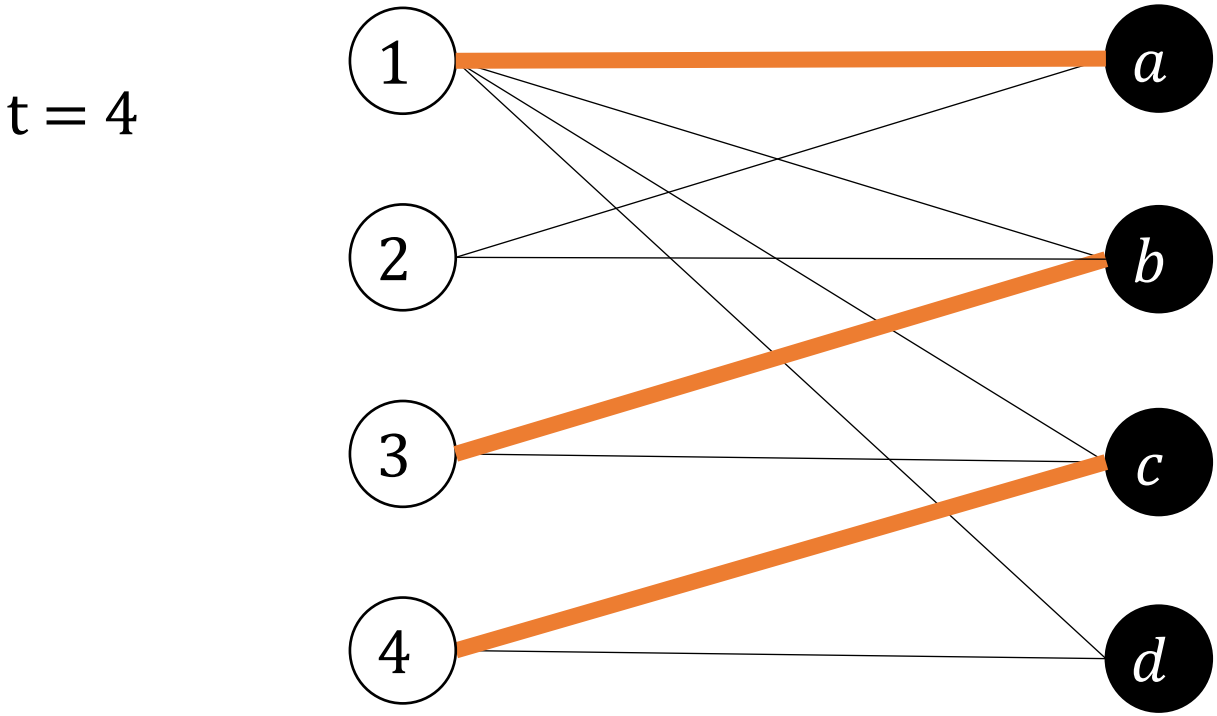



requests

resources

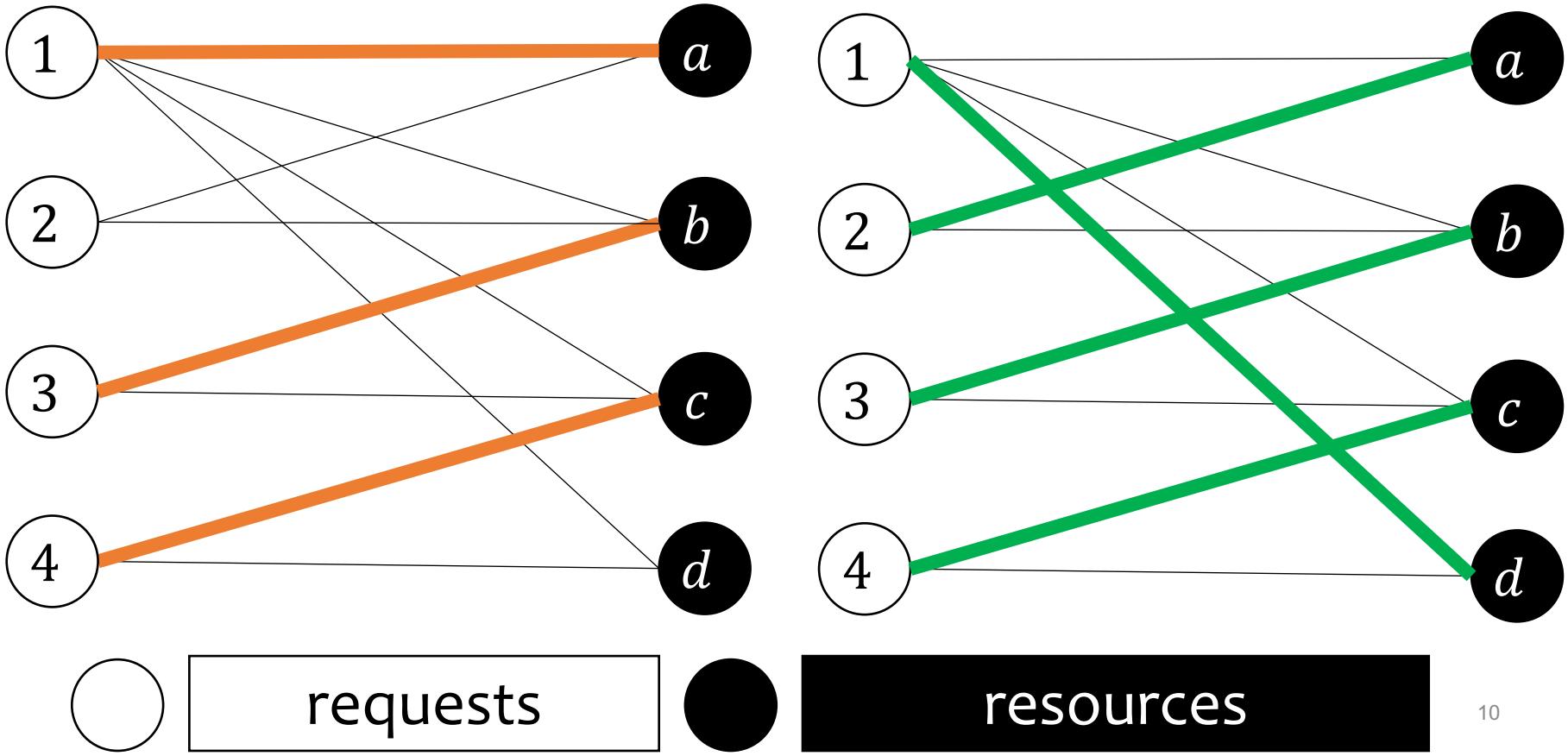
# Online Bipartite Matching Problem (BMP)

- Consider a scenario, where some request arrives and some resources are released at a time:



# Online Bipartite Matching Problem (BMP)

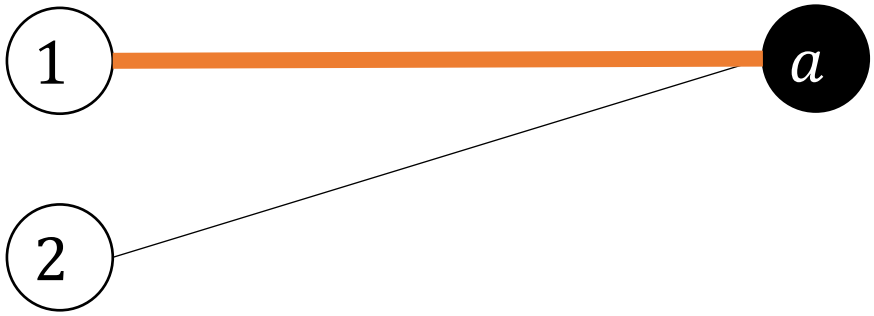
- Compared with the offline optimal solution



# Hardness of the Online BMP

- If your algorithm is deterministic

t = 1



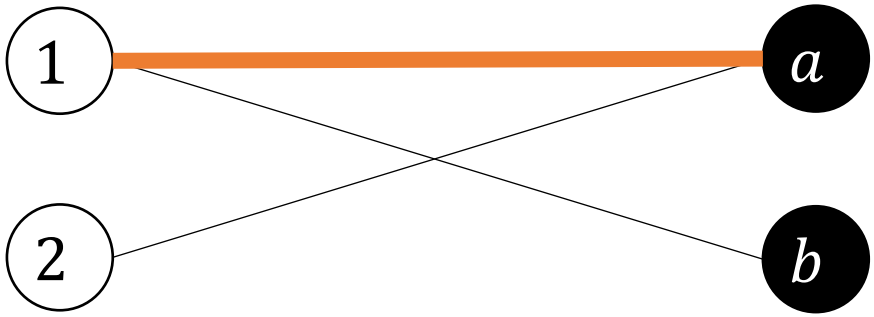

requests

resources

# Hardness of the Online BMP

- If your algorithm is deterministic

$t = 2$

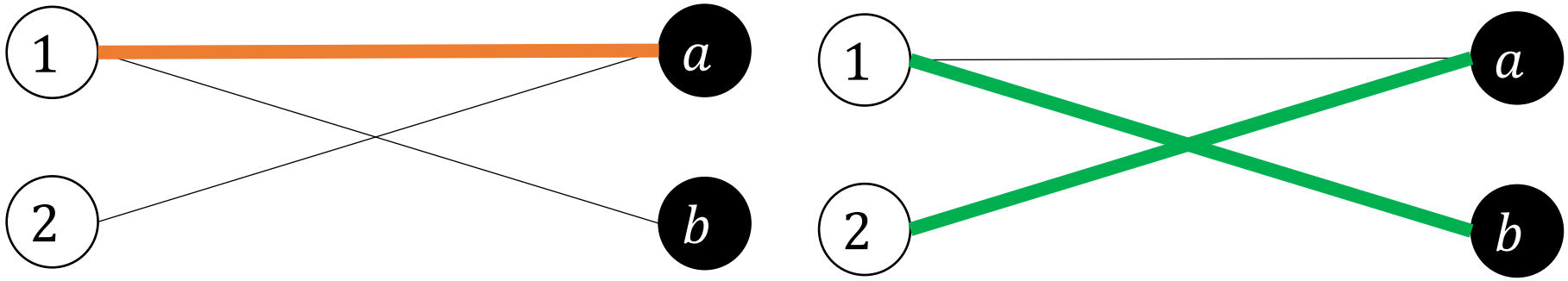



requests

resources

# Hardness of the Online BMP

- Compared with the offline optimal solution




- No deterministic algorithm can bound the ratio larger than 1/2

requests

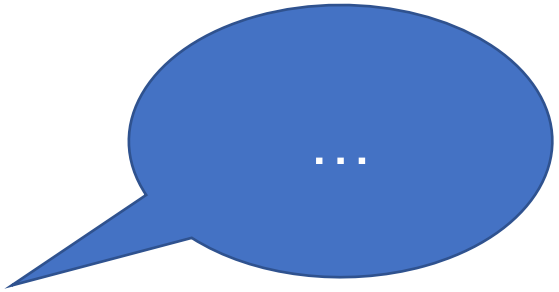
resources

# Programming Project #2: Online resource allocation

- Input:
  - # time slots
  - Time slot ID, # requests, # resources, request IDs, and resource IDs at the time slot
  - Number of edges and edges between a request ID and a resource ID at the time slot
- Procedure:
  - Assign the resource to a request at the time slot when the resource is released
- Output:
  - # assigned pairs
  - The time slot
- The grade is proportional to # assigned pairs

# Programming Project #2: Online resource allocation

- Input:
  - # time slots
  - Time slot ID, # requests, # resources, request IDs, and resource IDs at the time slot
  - Number of edges and edges between a request ID and a resource ID at the time slot
- Procedure:
  - Assign the resource to a request at the time slot when the resource is released
- Output:
  - # assigned pairs
  - The time slot
- The grade is proportional to # assigned pairs



# Programming Project #2: Online resource allocation

- Input:
  - # time slots
  - Time slot ID, # requests, # resources, request IDs, and resource IDs at the time slot
  - Number of edges and edges between a request ID and a resource ID at the time slot
- Procedure:
  - Assign the resource to a request at the time slot when the resource is released
- Output:
  - # assigned pairs
  - The time slot
- The grade is proportional to # assigned pairs

# Programming Project #2: Online resource allocation

- Input:
  - # time slots
  - Time slot ID, # requests, # resources, request IDs, and resource IDs at the time slot
  - Number of edges and edges between a request ID and a resource ID at the time slot
- Procedure:
  - Assign the resource to a request at the time slot when the resource is released
- Output:
  - # assigned pairs
  - The time slot
- The grade is proportional to # assigned pairs

# Programming Project #2: Online resource allocation

- Input:
  - # time slots
  - Time slot ID, # requests, # resources, request IDs, and resource IDs at the time slot
  - Number of edges and edges between a request ID and a resource ID at the time slot
- Procedure:
  - Assign the resource to a request at the time slot when the resource is released
- Output:
  - # assigned pairs
  - The time slot
- Implement a designated **randomized** algorithm

# Programming Project #2: Online resource allocation

- **Designated Algorithm:**
  1. For each time slot:
  2. Uniform-randomly assign a value in [0, 1] for each arrival requests
  3. Assign the available resource with the smallest ID to the unsatisfied request with the maximum random value
  4. Repeat step 3 until there is no available resource
  5. Go to step 1 for the next time slot
- **Note:**
  1. Use default_random_engine and uniform_real_distribution
  2. Let the random seed be a parameter of main function
  3. Define classes **Resource** and **Request**, and use vector to store the objects of class Resource and Request
  4. Generate new resource and request by push_back
  5. Design **static** member variables and **static** member functions to count and get the number of satisfied requests

# Programming Project #2: Online resource allocation

```
// You need forward declaration
class Resource{
  int id;
  bool matched;
  int requestId;
public:
  // constructors…
  bool operator->* (Request &b){…}
  // ->* check whether both of them are un-matched.
  // If they are, then match them and return true;
  // otherwise, return false
  // accessor
};
class Request{
  int id;
  bool matched;
  int resourceId;
  double weight; // a random value, indicates the priority
public:
  // constructors…
  friend class Resource;
  // accessor
};
```

# Discussion

- Why does the randomized algorithm work?
- What can be added to improve the performance?
- Discussion & bonus

# Discussion

- Why does the randomized algorithm work?
- It can avoid being stuck in the trap of adversary
- Adversary: deliberately choose difficult data to maximize the gap between the generated solution and the offline optimum

- What can be added to improve the performance?
- Distribution of the input

- Discussion & bonus

# Further Reading

- Randomized primal-dual analysis of RANKING for online bipartite matching, in SODA 2013
- Online stochastic matching beating 1-1/e, in FOCS 2009
- …

- 演算法頂尖研討會: STOC, FOCS, SODA…
- 一個字，神

# Input Sample: request.txt

Format:

#timeSlots
timeSlotID1 #requests #resources
requestID1 requestID2 …
resourceID1 resourceID2 …
#edges
edgeID1 requestID resourceID
edgeID2 …
timeSlotID2…

之後揭曉，先自己設計input

# Output Sample: result.txt

Format:

#satisfiedRequests
requestID resourceID
…

下次揭曉，用自己設計input產生答案

# Input Sample: request.txt

Format:

```
3
0 5 2
0 1 2 3 4
0 1
3
0 3 0
1 0 1
2 4 1
1 1 1
5
2
2
3 0 2
4 4 2
```

```
2 3 3
6 7 8
3 4 5
9
5 0 3
6 1 3
7 3 3
8 6 3
9 8 3
10 5 4
11 2 5
12 4 5
13 5 5
```

Format:

#timeSlots
timeSlotID1 #requests #resources
requestID1 requestID2 ...
resourceID1 resourceID2 ...
#edges
edgeID1 requestID resourceID
edgeID2 ...
timeSlotID2...

# Output Sample: result.txt

Format:

6

0 2

2 5

3 0

4 1

5 4

8 3

Format:

#satisfiedRequests

requestID resourceID

…

# Note

- Deadline: 4/16 Tue
- 小老師deadline: 4/2 Tue (暫定)

- E-course

- **C++ Source code**

- Show a good programming style