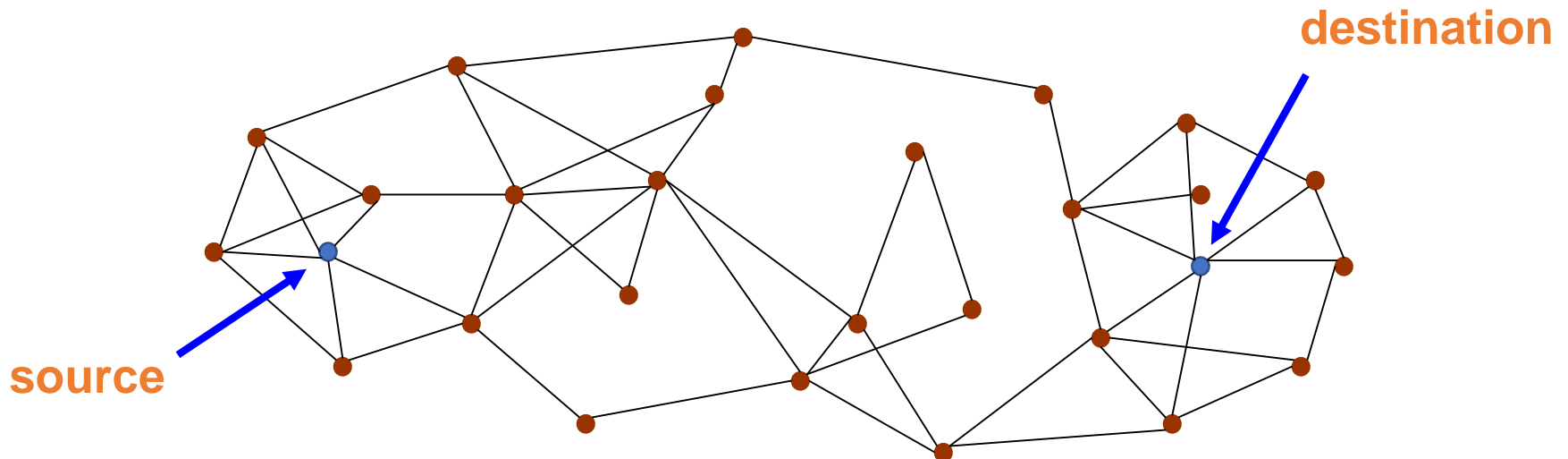# Data Structure Programming Project #1

郭建志

Background

- Large-scale geographic routing
- Every node only has the local information (i.e., its neighbors' information) and the destination's information

Background

- Large-scale geographic routing
- Every node only has the local information (i.e., its neighbors' information) and the destination's information
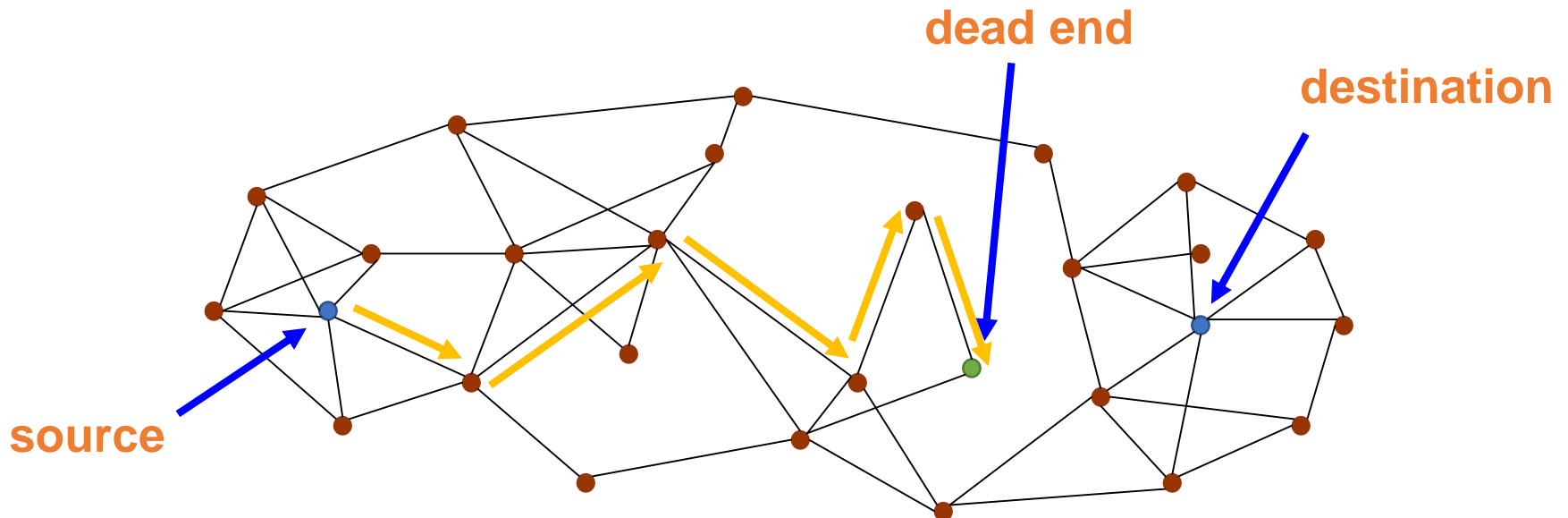- So, how should a source route a packet to a designated destination?

destination

source

Naïve Routing Scheme: Greedy

- Forward the packet towards the neighboring node that is closer to the destination
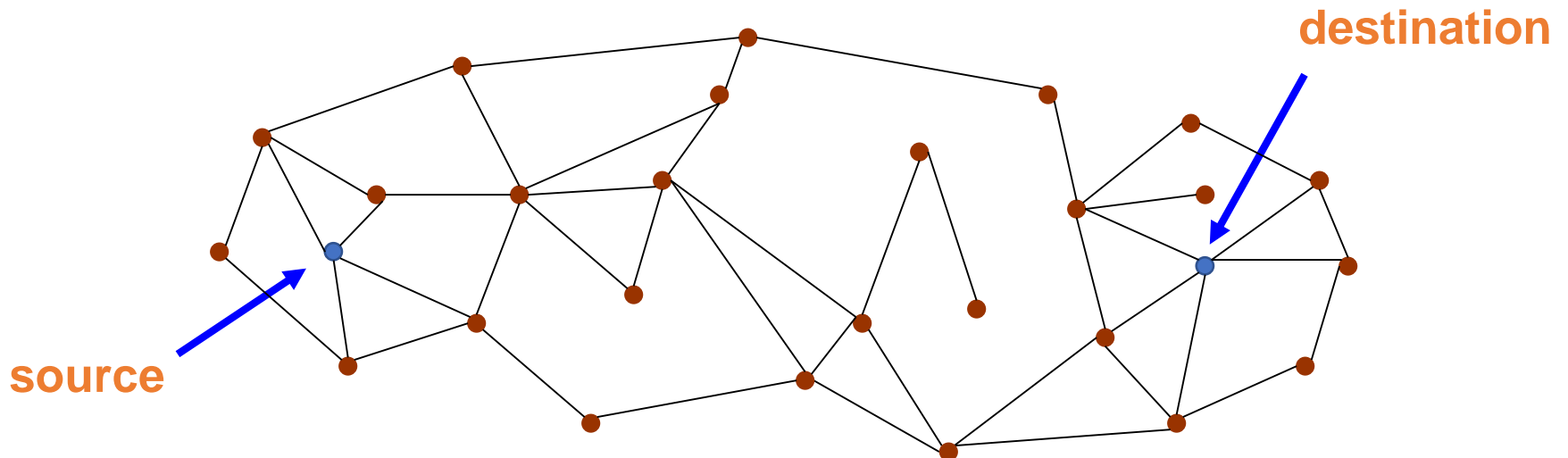
Naïve Routing Scheme: Greedy
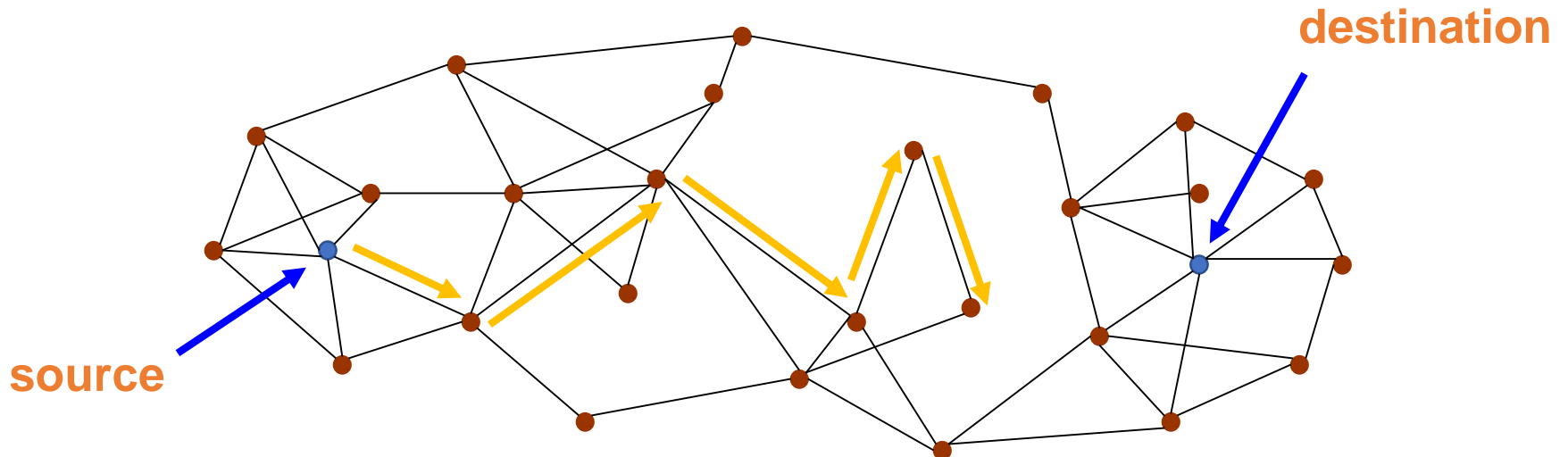
- Forward the packet towards the neighboring node that is closer to the destination
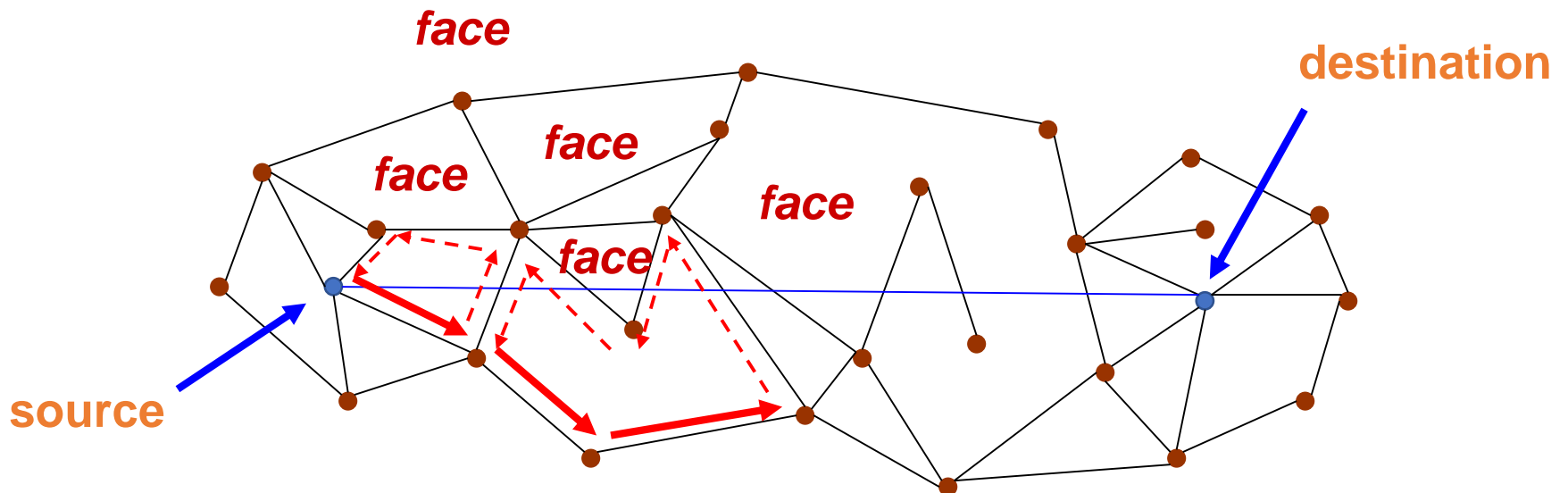- However, the packet may be stuck at the dead end

Face Routing

- Generate a planar subgraph: no crossing links
- Right-hand rule
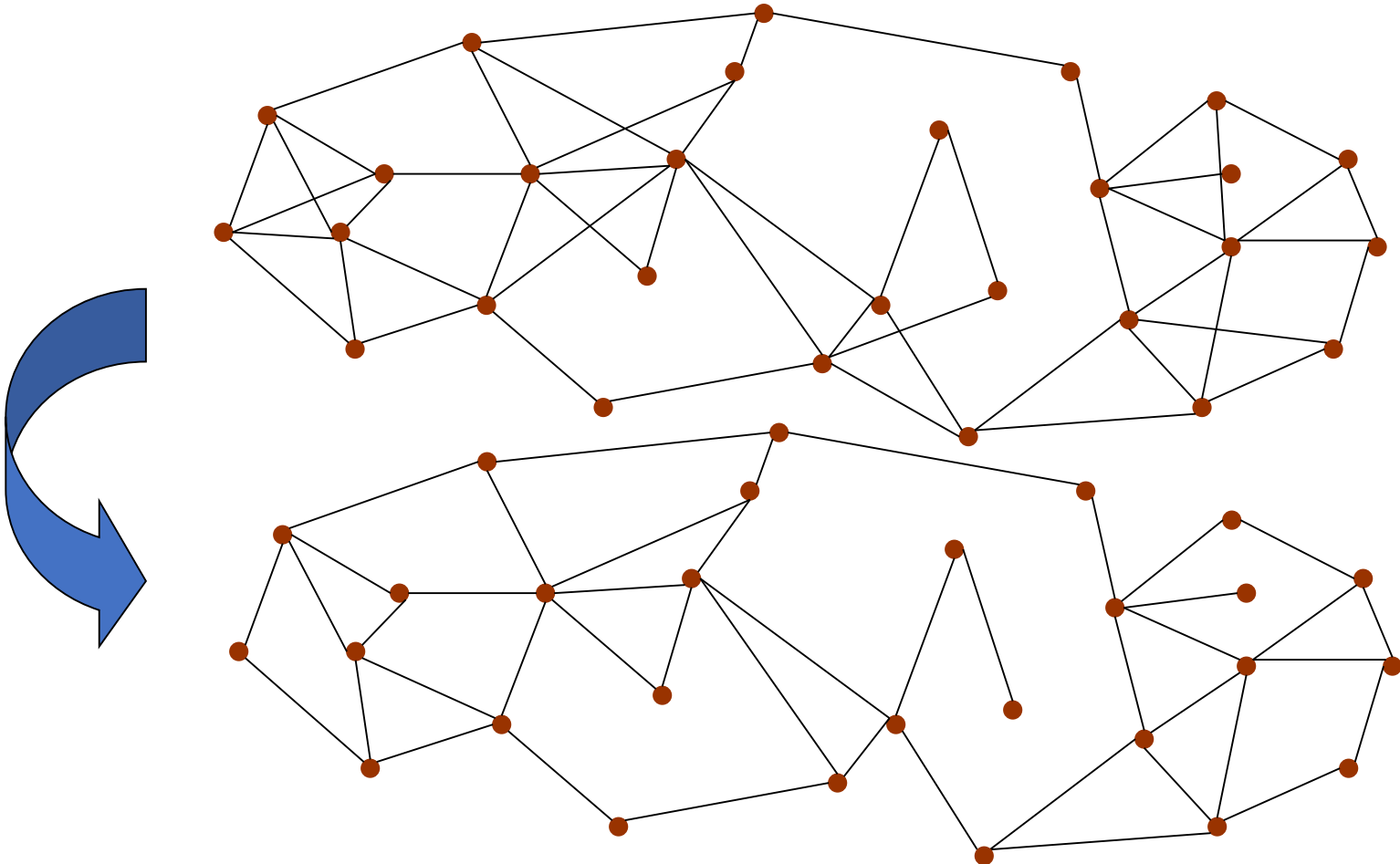- Change faces

destination

source

Face Routing

- Generate a planar subgraph: no crossing links
- Right-hand rule
- Change faces

Programming Project #1:
Generate a Planar Subgraph

- Generate a planar subgraph: no crossing links

Programming Project #1: Generate a Planar Subgraph

- Input:
  - Number of nodes
  - Nodes with non-negative coordinates (x, y) (the input graph is connected when we add links if dist(u,v) ≤ 1)

- Procedure:
  - Add a link between any two nodes u, v as dist(u,v) ≤ 1
  - Remove some links to generate a planar graph

- Output:
  - The edges before and after planarization

- The grade is proportional to the number of remaining edges

Input file:

5		
0	1.5	2.3
1	1.2	2.6
2	1.3	1.2
...		

Output file:

12		
0	0	1
1	1	2
2	1	3
...		
8		
0	0	1
2	1	3
4	3	4

Programming Project #1: Generate a Planar Subgraph

- Input:
  - Number of nodes
  - Nodes with non-negative coordinates (x, y) (the input graph is connected when we add links if dist(u,v) ≤ 1)

- Procedure:
  - Add a link between any two nodes u, v as dist(u,v) ≤ 1
  - Remove some links to generate a planar graph

- Output:
  - The edges before and after planarization

- The grade is proportional to the number of remaining edges

Input file:

```
5
0    1.5        2.3
1    1.2        2.6
2    1.3        1.2
...
```

Output file:

```
12
0           0            1
1           1            2
2           1
...
8
            
2           1
4           3            4
```

...

Programming Project #1: Generate a Planar Subgraph
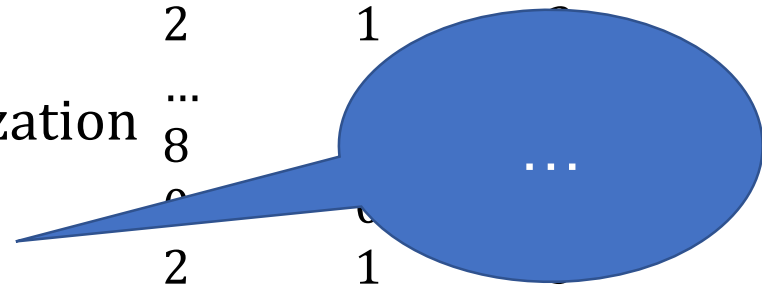
- Input:
  - Number of nodes
  - Nodes with non-negative coordinates (x, y) (the input graph is connected when we add links if dist(u,v) ≤ 1)

- Procedure:
  - Add a link between any two nodes u, v as dist(u,v) ≤ 1
  - Remove some links to generate a planar graph

- Output:
  - The edges before and after planarization

- The grade is proportional to the number of remaining edges

Input file:

```
5
0    1.5      2.3
1    1.2      2.6
2    1.3      1.2
...
```

Output file:

```
12
0         0         1
1         1         2
2         1         
...
8
         
2         1         
4         3         4
```

棄選了

Programming Project #1: Generate a Planar Subgraph

- Input:
  - Number of nodes
  - Nodes with non-negative coordinates (x, y) (the input graph is connected when we add links if dist(u,v) ≤ 1)

- Procedure:
  - Add a link between any two nodes u, v as dist(u,v) ≤ 1
  - Remove some links to generate a planar graph

- Output:
  - The edges before and after planarization

- Implement a designated algorithm to remove the links

Input file:

```
5
0    1.5      2.3
1    1.2      2.6
2    1.3      1.2
...
```

Output file:

```
12
0          0          1
1          1          2
2          1          3
...
8
0          0          1
2          1          3
4          3          4
```

Programming Project #1: Generate a Planar Subgraph

- Output:
  - The edges before and after planarization

- **Notice that:**
  - The first node ID should be smaller than the second one for each link
  - The links should be sequentially indexed in ascending order of the first node ID
  - If there is a tie, the two links are indexed in ascending order of the second node ID
  - The IDs of remaining links after planarization should be identical to the ones before the planarization

Input file: node.txt
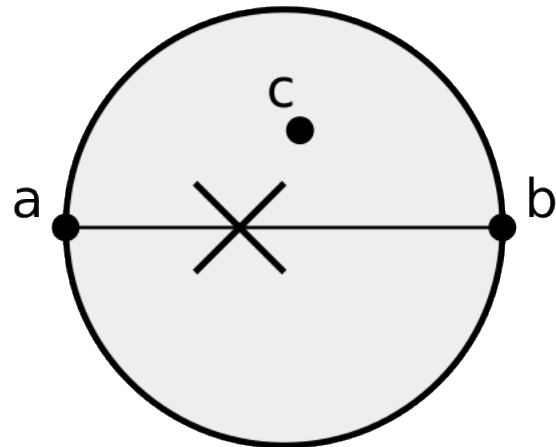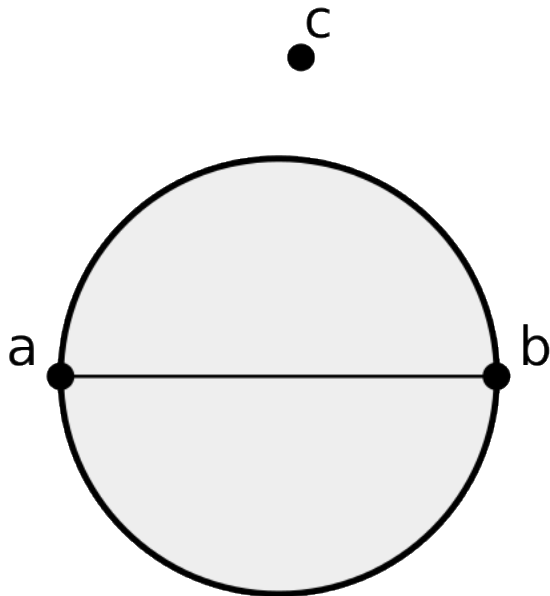
```
5
0    1.5    2.3
1    1.2    2.6
2    1.3    1.2
...
```

Output file: link.txt

```
12
0    0    1
1    1    2
2    1    3
...
8
0    0    1
2    1    3
4    3    4
```
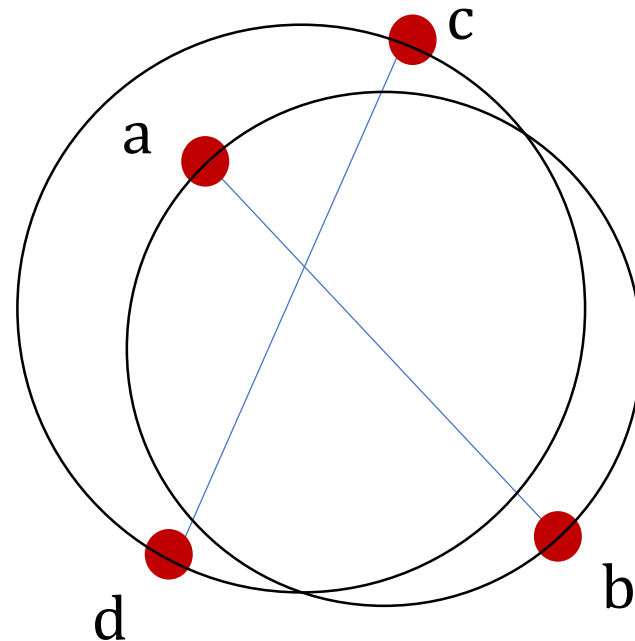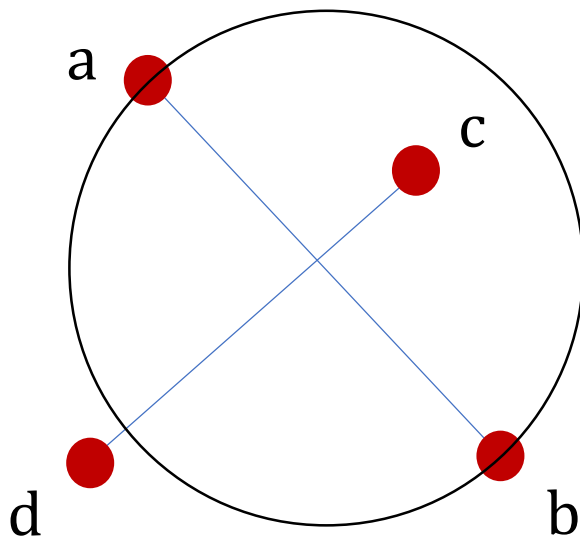
Programming Project #1: Generate a Planar Subgraph

- Implement a designated algorithm to remove the links

- Cut a link between two neighboring nodes a and b, if there is other node c is within their the closed disc, where line segment $\overline{ab}$ is a diameter
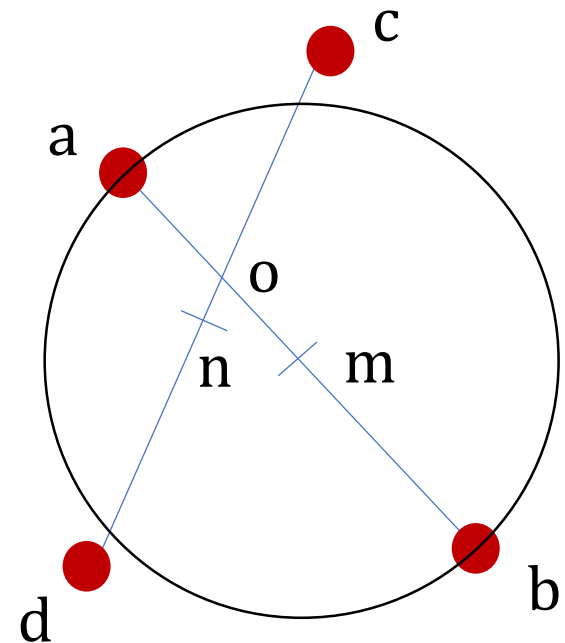
Why does the algorithm work?

- Prove by contradiction
- Assume that there is an intersection node on the segments $\overline{ab}$ and $\overline{cd}$

Why does the algorithm work?

- Let m and n denote the middle points of $\overline{ab}$ and $\overline{cd}$, respectively

- Without loss of generality, we assume that $\overline{am}$ and $\overline{cn}$ intersect at node o

- Since node a is not in the circle of which $\overline{cd}$ is the diameter, $\overline{cn} < \overline{an}$...(1)

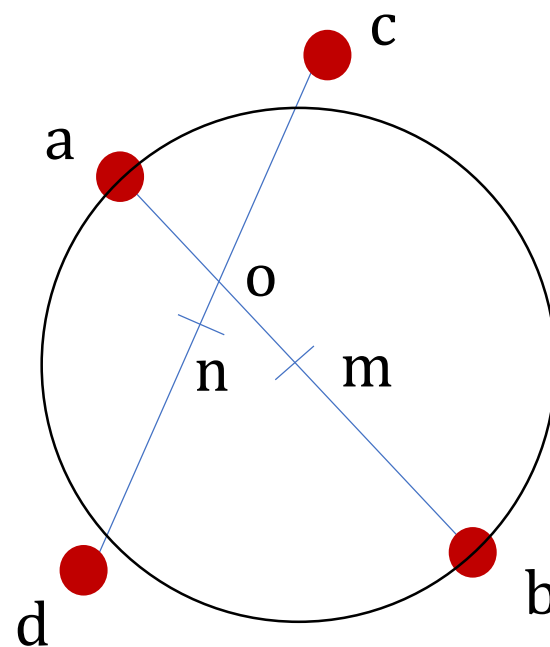- Similarly, since node c is not in the circle of which $\overline{ab}$ is the diameter, $\overline{am} < \overline{cm}$...(2)

Why does the algorithm work?

- $\overline{cn} < \overline{an}...(1)$
- $\overline{am} < \overline{cm}...(2)$
- $(1)+(2) = \overline{cn} + \overline{am} < \overline{an} + \overline{cm}...(3)$

- Due to triangular inequality, $\overline{ao} + \overline{on} \geq \overline{an}...(4)$
- Similarly, $\overline{co} + \overline{om} \geq \overline{cm}...(5)$
- $(4)+(5) = \overline{ao} + \overline{om} + \overline{co} + \overline{on}$
  $= \overline{am} + \overline{cn} \geq \overline{an} + \overline{cm}...(6)$

- $(3) \rightarrow\leftarrow (6)$

Input Sample: node.txt

```
11
0    1.4142    1.1534
1    1.97      1.85
2    0.8823    1.3926
3    1.9996    1.9484
4    0.6301    1.9145
5    0.3329    1.1756
6    1.952     0.5866
7    1.8181    0.4043
8    0.4949    1.946
9    0.2079    1.4758
10   0.1364    1.4084
```

Output Sample: link.txt

22		
0	0	1
1	0	2
2	0	3
3	0	6
4	0	7
5	1	3
6	2	4
7	2	5
8	2	8
9	2	9
10	2	10
11	4	5
12	4	8
13	4	9
14	4	10
15	5	8
16	5	9
17	5	10
18	6	7
19	8	9
20	8	10
21	9	10

11		
0	0	1
1	0	2
3	0	6
5	1	3
6	2	4
7	2	5
12	4	8
17	5	10
18	6	7
19	8	9
21	9	10

Note

- Deadline: 10/11 Thu

- E-course

- **C Source code**

- **Readme including**: *brief description* and *time complexity* analysis of you implementation