CSCI2100B Data Structures
Advanced Graph

Irwin King

king@cse.cuhk.edu.hk
http://www.cse.cuhk.edu.hk/~king

Department of Computer Science & Engineering
The Chinese University of Hong Kong

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Outline

- The Single-Source Shortest Path Problem
- The Maximum Flow Problem
- The Maximum Cut Problem
- The Traveling Salesman Problem

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Outline

- The Single-Source Shortest Path Problem
- The Maximum Flow Problem
- The Maximum Cut Problem
- The Traveling Salesman Problem

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

The Single-Source Shortest Path Problem

- Input: Directed graph G = (V, E), edge lengths c_e for each $e \in E$, source vertex $s \in V$.
- Goal: For every destination $v \in V$, compute the length (sum of edge costs) of a shortest s-v path.

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

On Dijkstra's Algorithm

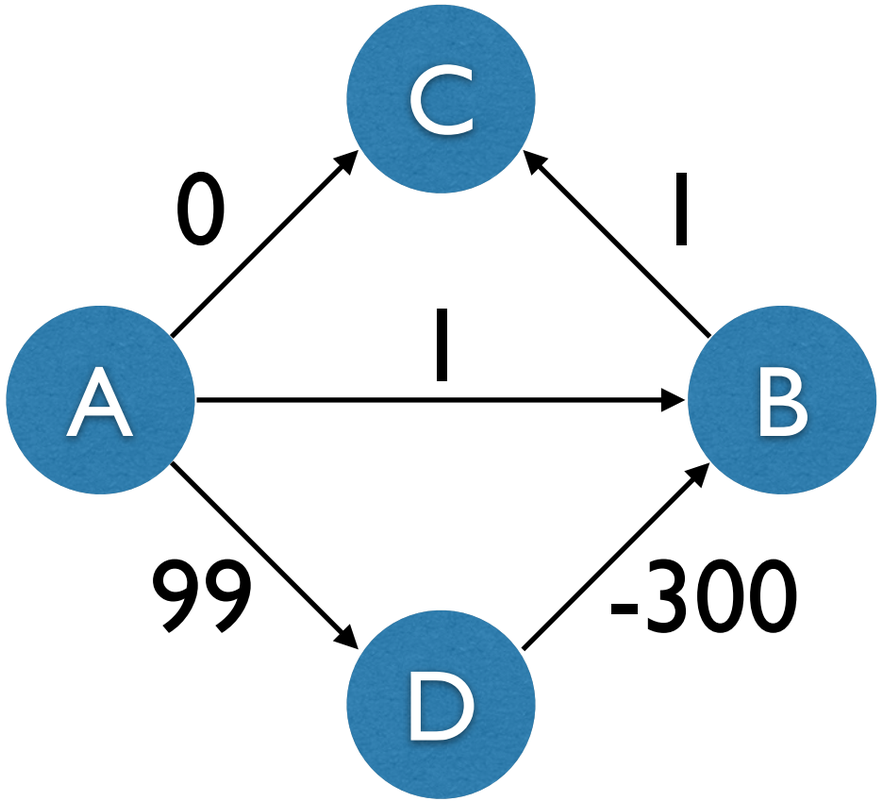
- Advantages: O(m log n) running time using heaps (n = number of vertices, m = number of edges)
- Disadvantages:

1. Not always correct with negative edge lengths [e.g. if edges → financial transactions]
2. Not very distributed (relevant for Internet routing)

- Solution: The Bellman-Ford algorithm

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

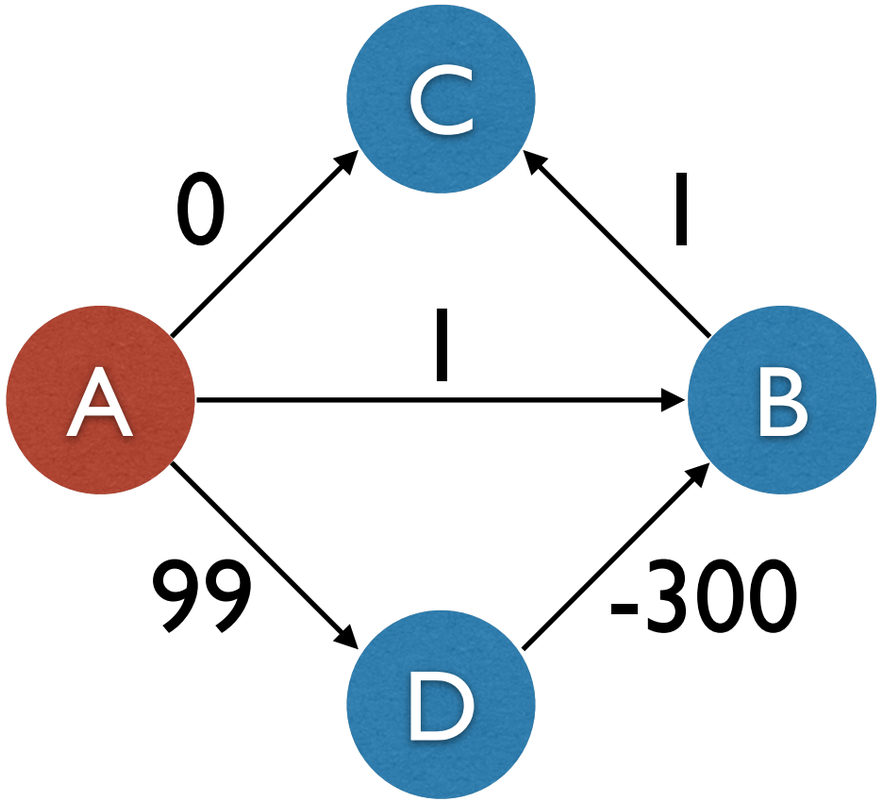
A Failed Example



v	Known	d_v	p_v
A	0	0	-
B	0	-	-
C	0	-	-
D	0	-	-

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

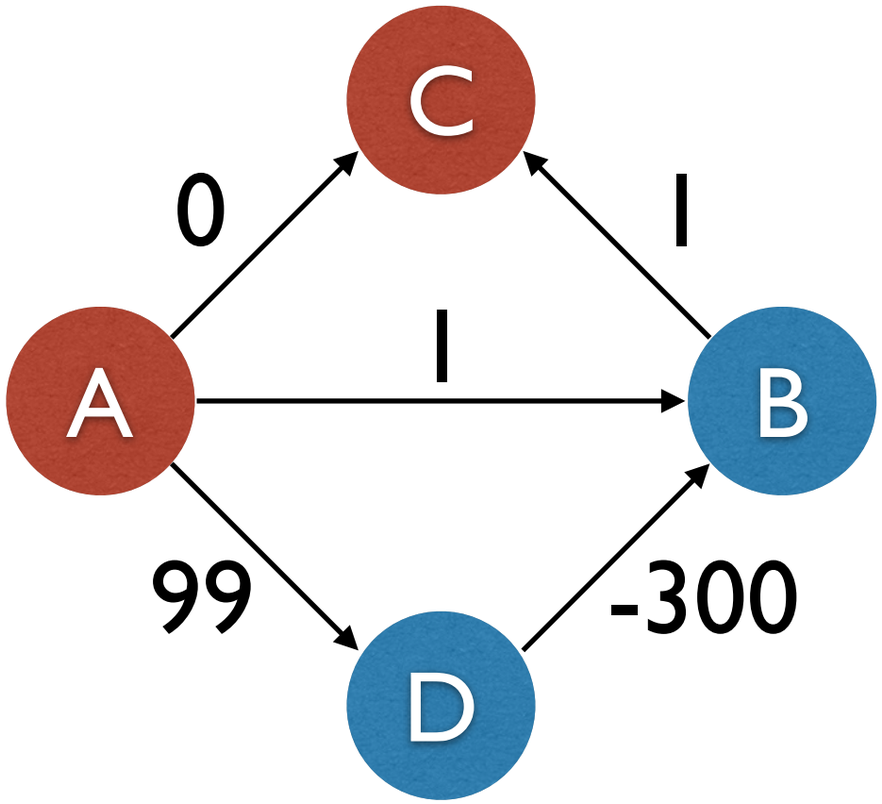
A Failed Example



v	Known	d_v	p_v
A	1	0	-
B	0	1	A
C	0	0	A
D	0	99	A

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

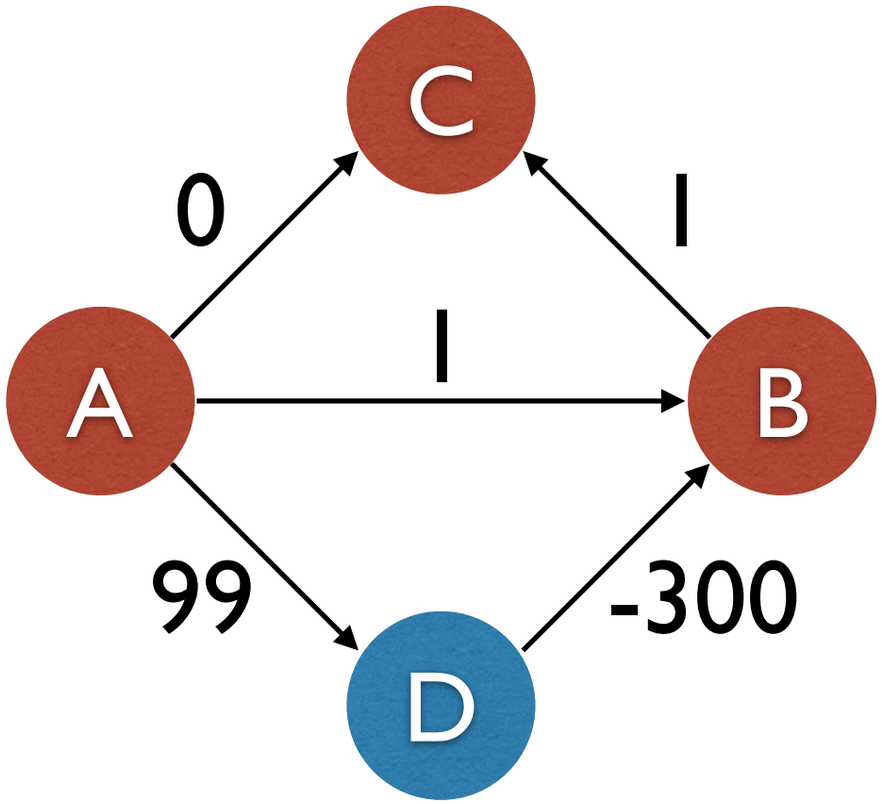
A Failed Example



v	Known	d_v	p_v
A	1	0	-
B	0	1	A
C	1	0	A
D	0	99	A

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

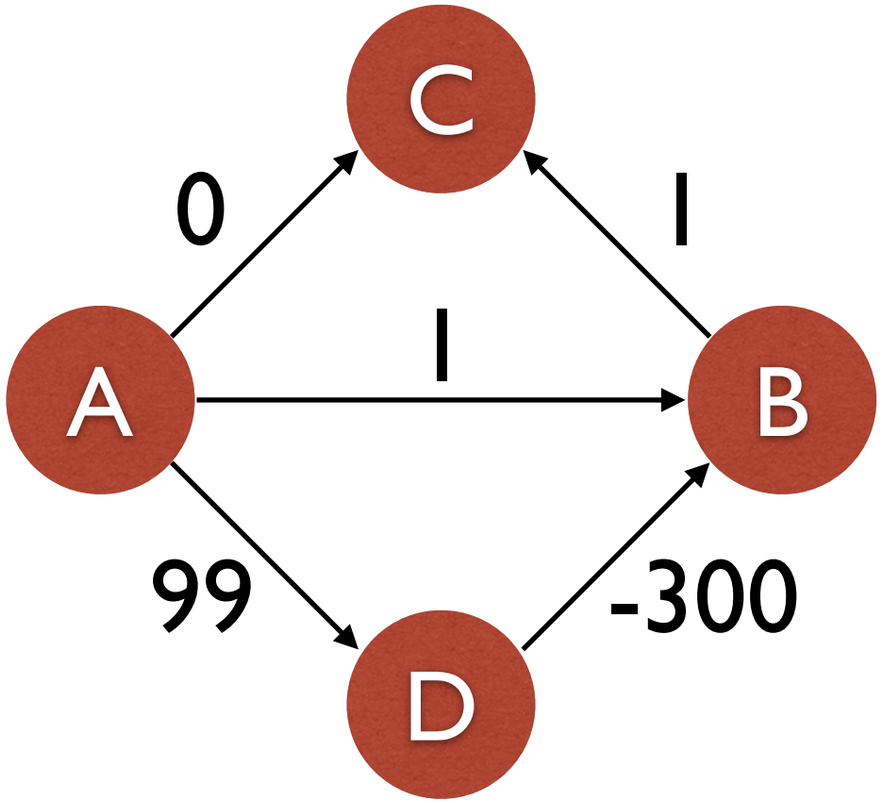
A Failed Example



v	Known	d_v	p_v
A	1	0	-
B	1	1	A
C	1	0	A
D	0	99	A

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

A Failed Example



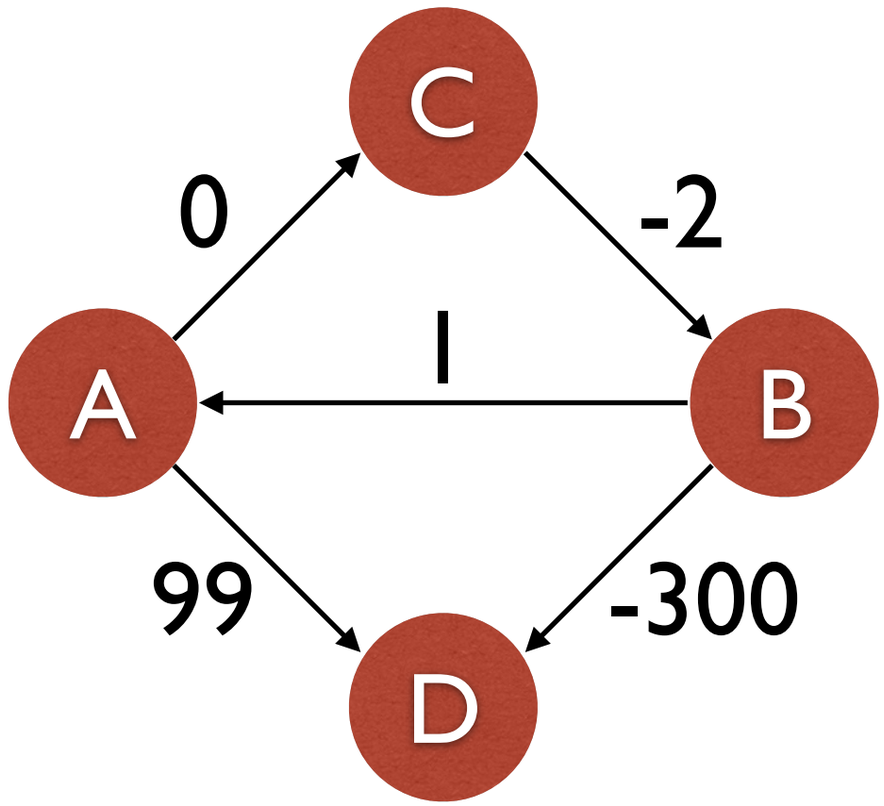
v	Known	d_v	p_v
A	1	0	-
B	1	1	A
C	1	0	A
D	1	99	A

However, the shortest path from A to B should be A->D->B with length -201, not 1.

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Negative cycles

- Def. A negative cycle is a directed cycle whose sum of edge weights is negative.
- If a graph contains a negative-weight cycle, then some shortest paths may not exist
- Negative Cycle
  - A->C->B (0-2+1=-1)



CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Bellman-Ford algorithm

- Finds all shortest-path lengths from a *source* $s \in V$ to all $v \in V$ or determines that a negative-weight cycle exists

Initialize $d[s]=0$ and $d[v]=\infty$ for all other vertices.

for $i \leftarrow 1$ **to** $|V| - 1$

  do for each edge $(u, v) \in E$

    do if $d[v] > d[u] + w(u, v)$

      then $d[v] \leftarrow d[u] + w(u, v)$ } Relaxation Step

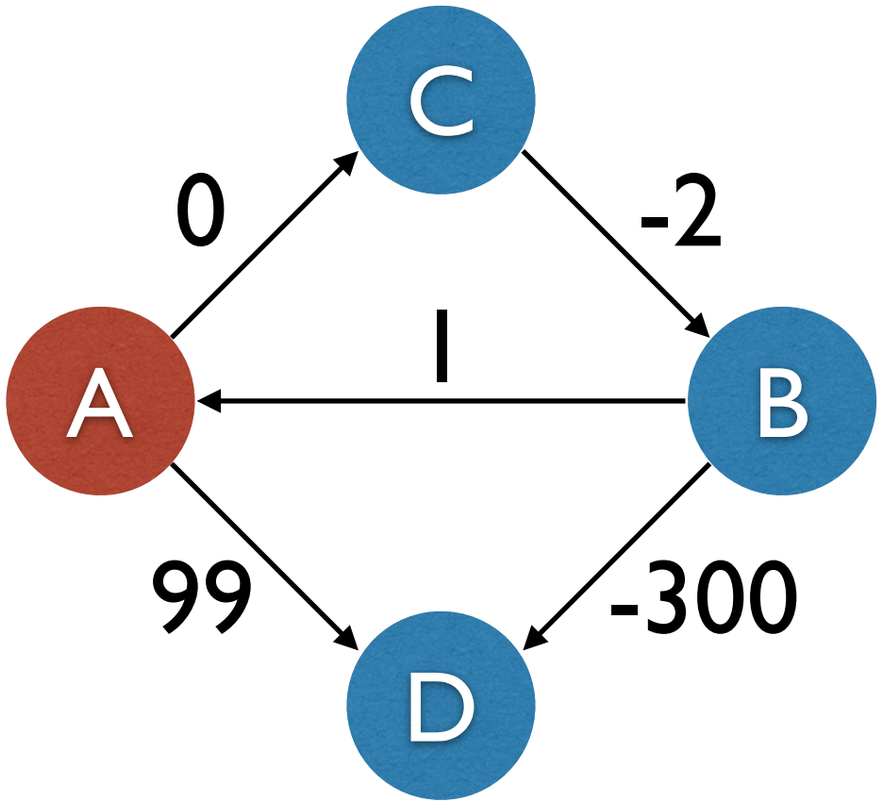
for each edge $(u, v) \in E$

  do if $d[v] > d[u] + w(u, v)$ } Check negative-weight cycle

    then report that a negative-weight cycle exists

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Example

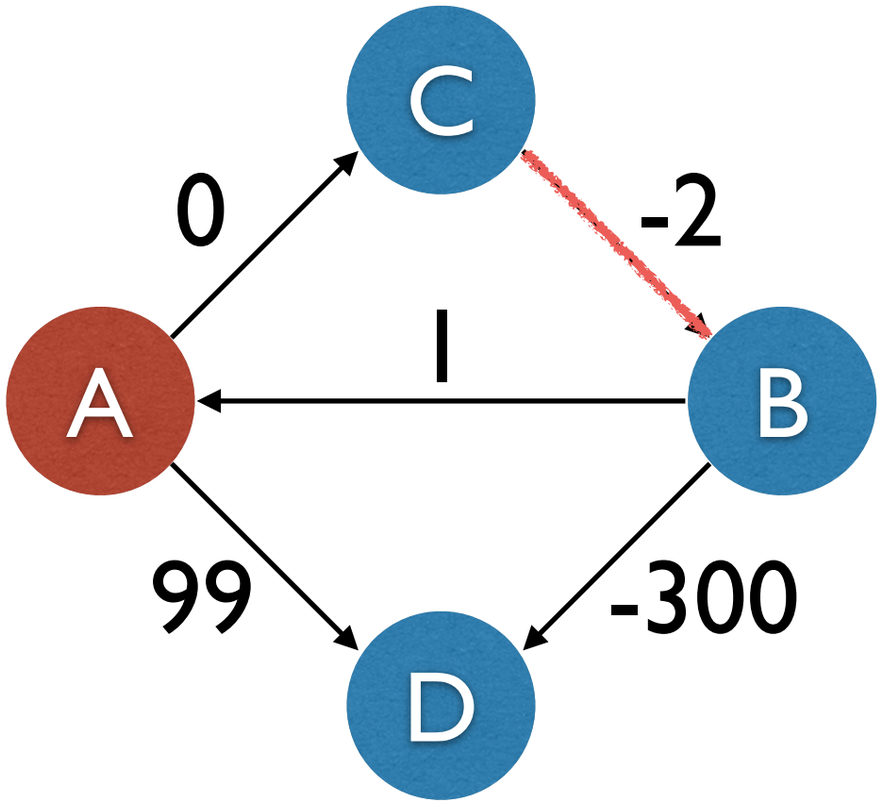


A	B	C	D
0	∞	∞	∞

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Example

- The first iteration

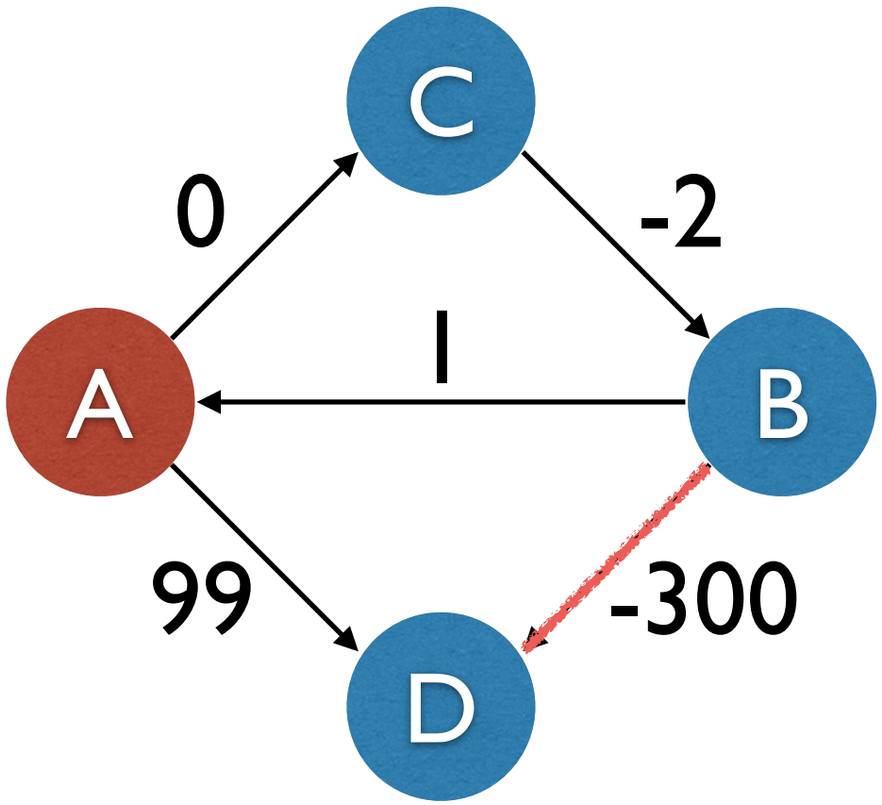


A	B	C	D
0	∞	∞	∞

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Example

- The first iteration

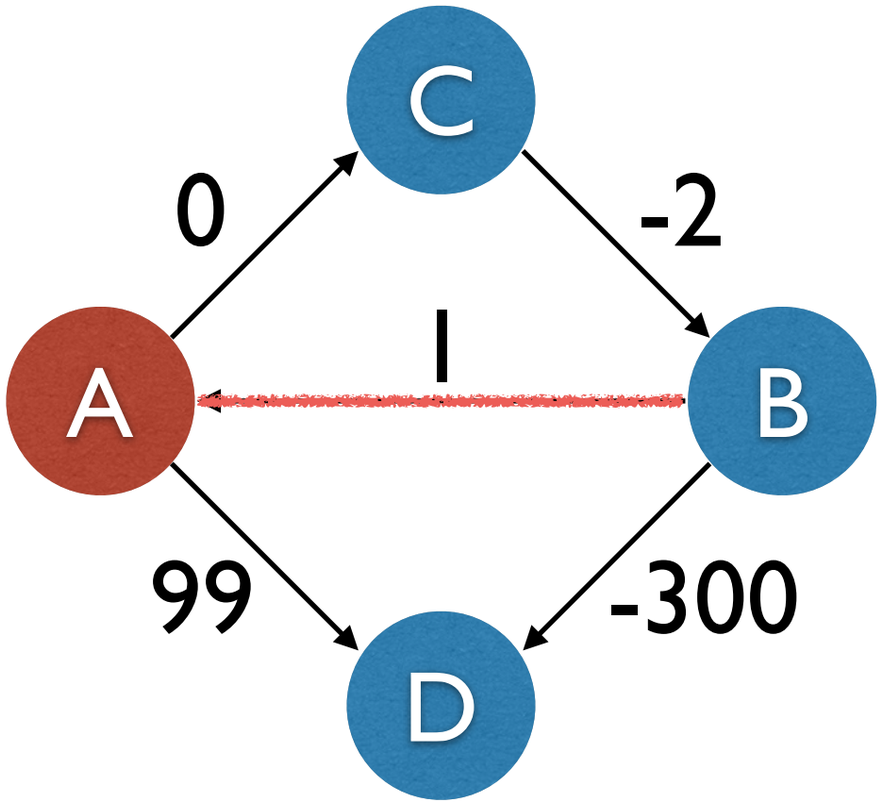


A	B	C	D
0	∞	∞	∞

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Example

- The first iteration

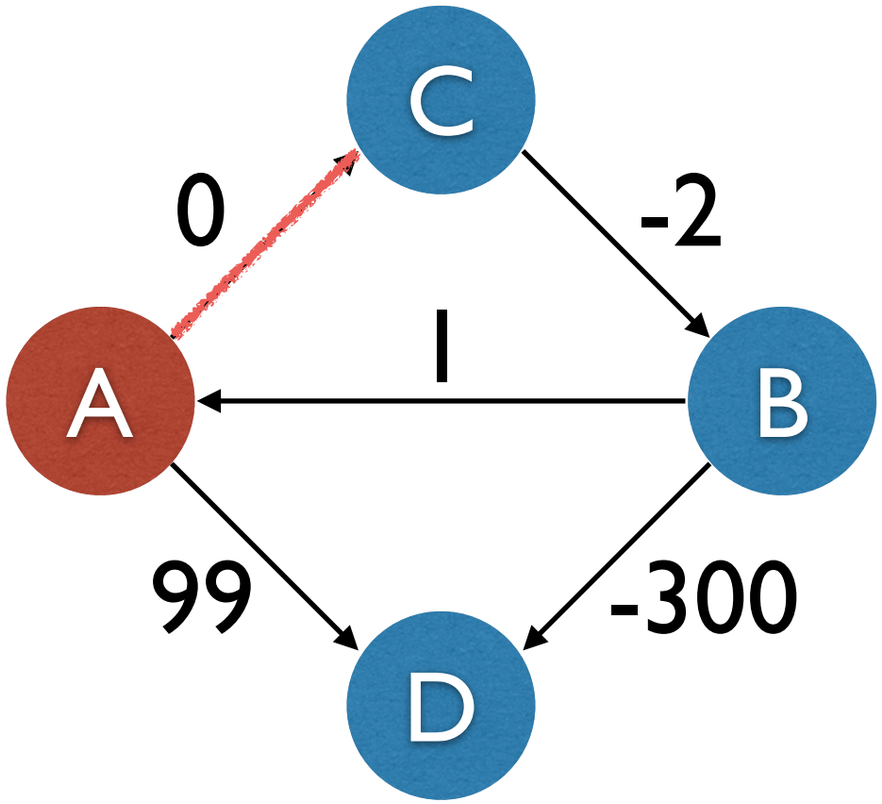


A	B	C	D
0	∞	∞	∞

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Example

- The first iteration

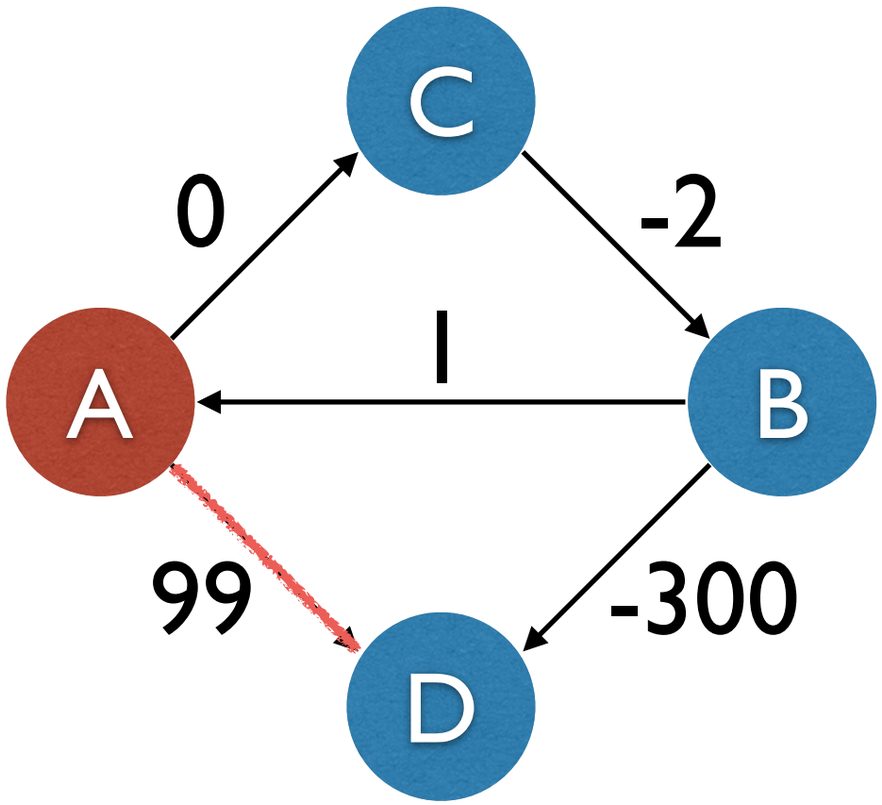


A	B	C	D
0	∞	0	∞

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Example

- The first iteration

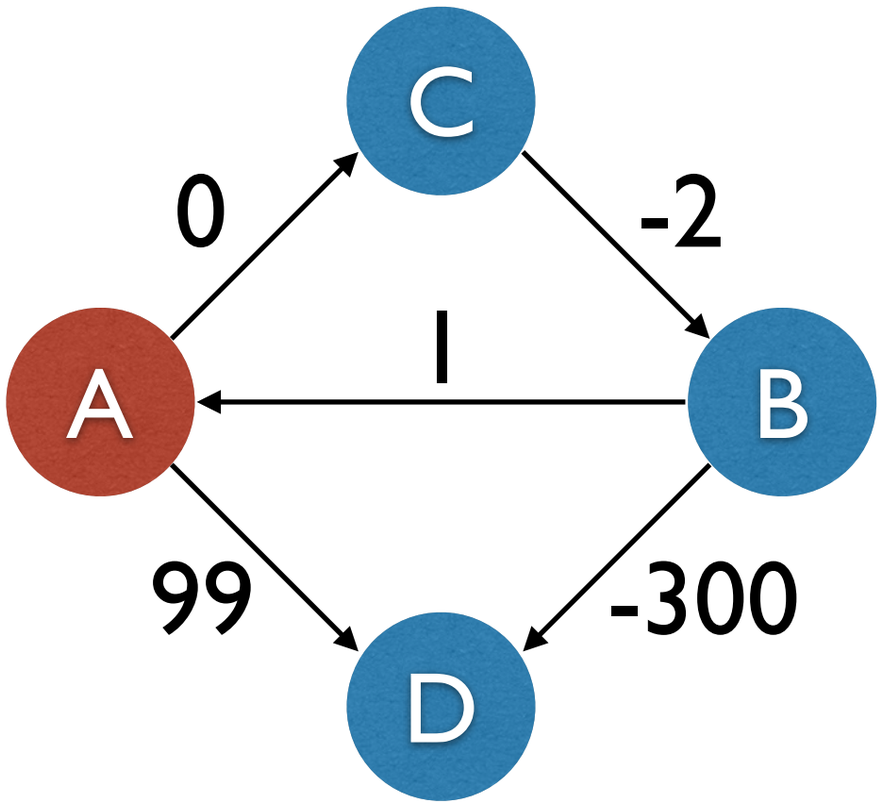


A	B	C	D
0	∞	0	99

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Example

- The second iteration

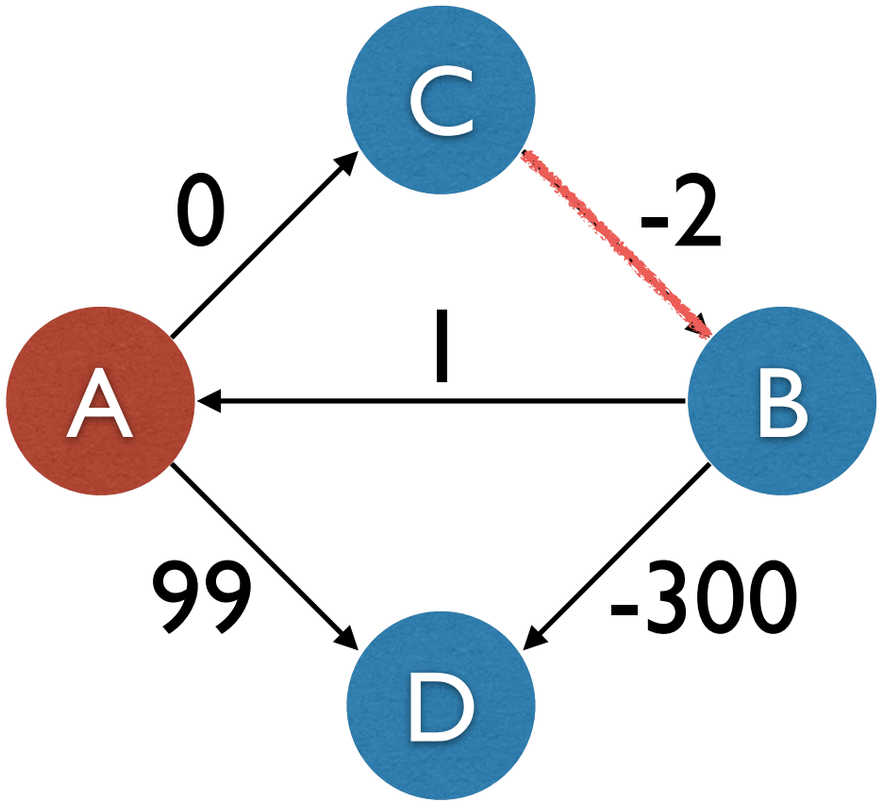


A	B	C	D
0	∞	0	99
0	∞	0	99

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Example

- The second iteration

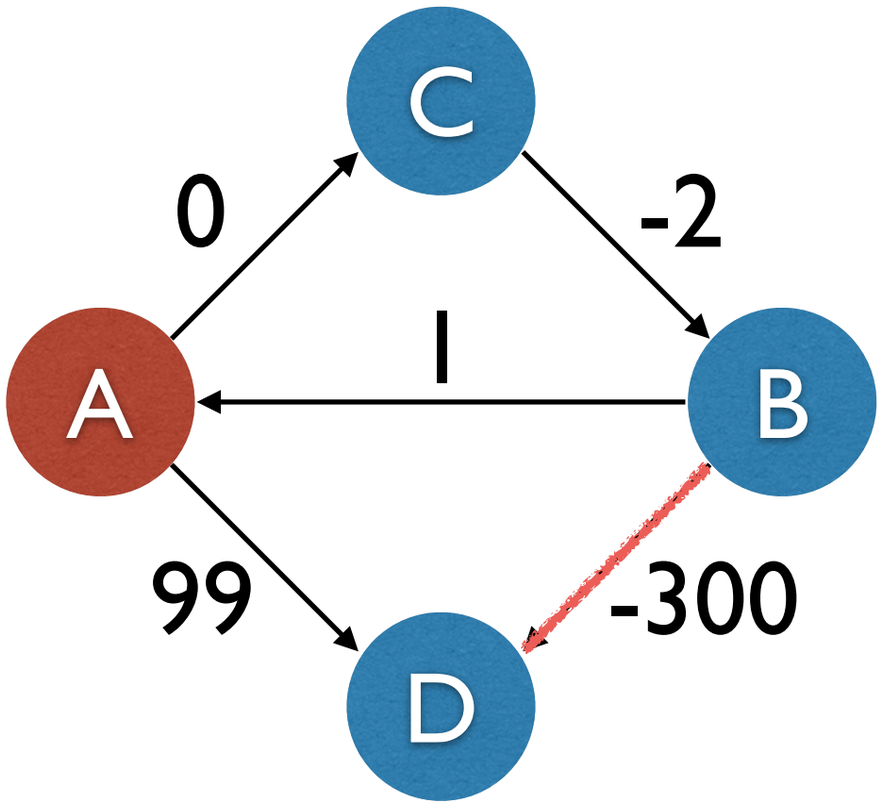


A	B	C	D
0	∞	0	99
0	-2	0	99

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Example

- The second iteration

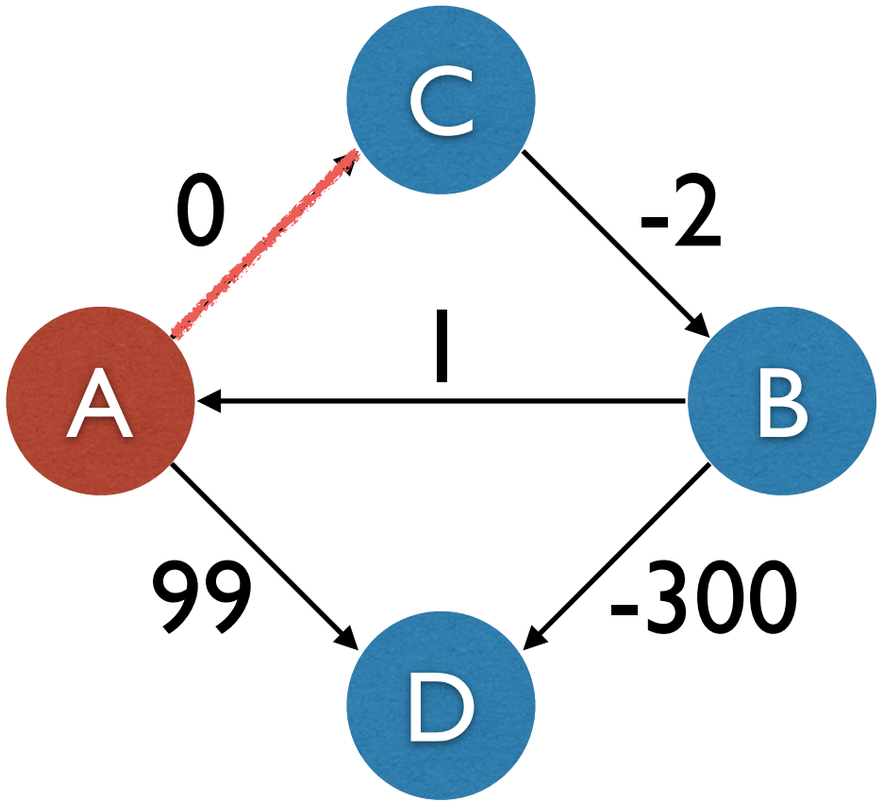


A	B	C	D
0	∞	0	99
0	-2	0	-302

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Example

- The second iteration

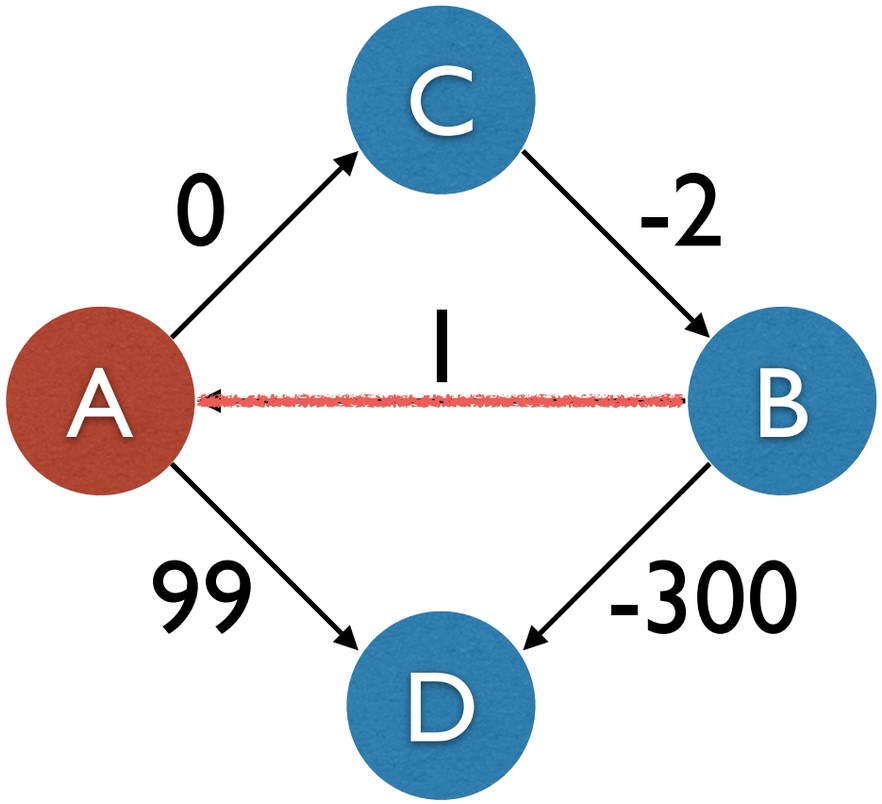


A	B	C	D
0	∞	0	99
0	-2	0	-302

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Example

- The second iteration

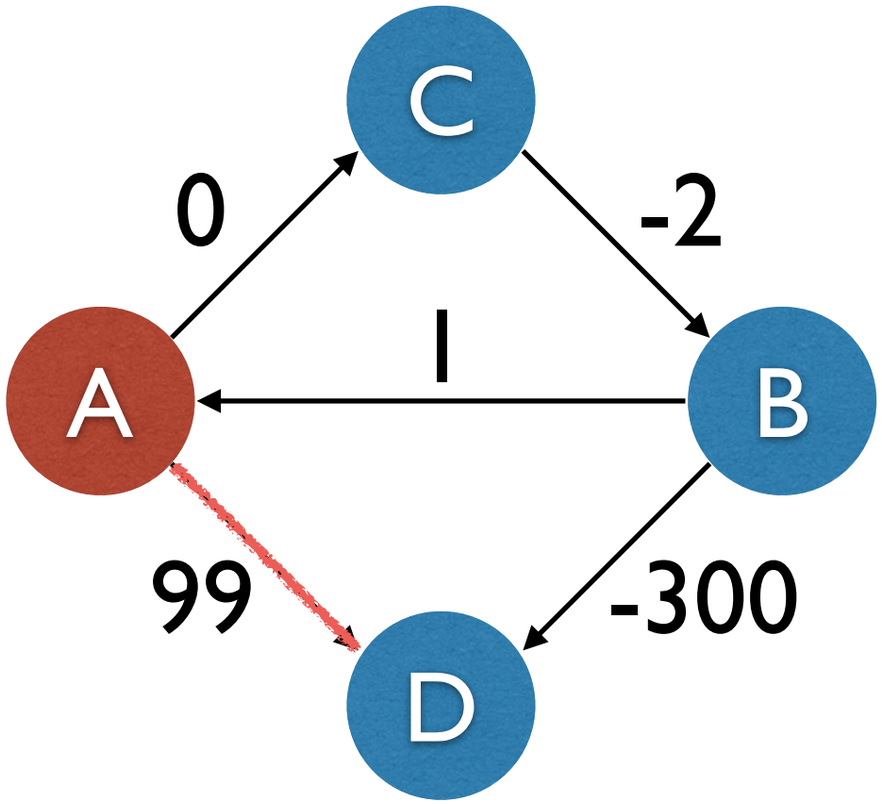


A	B	C	D
0	∞	0	99
-1	-2	0	-302

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Example

- The second iteration

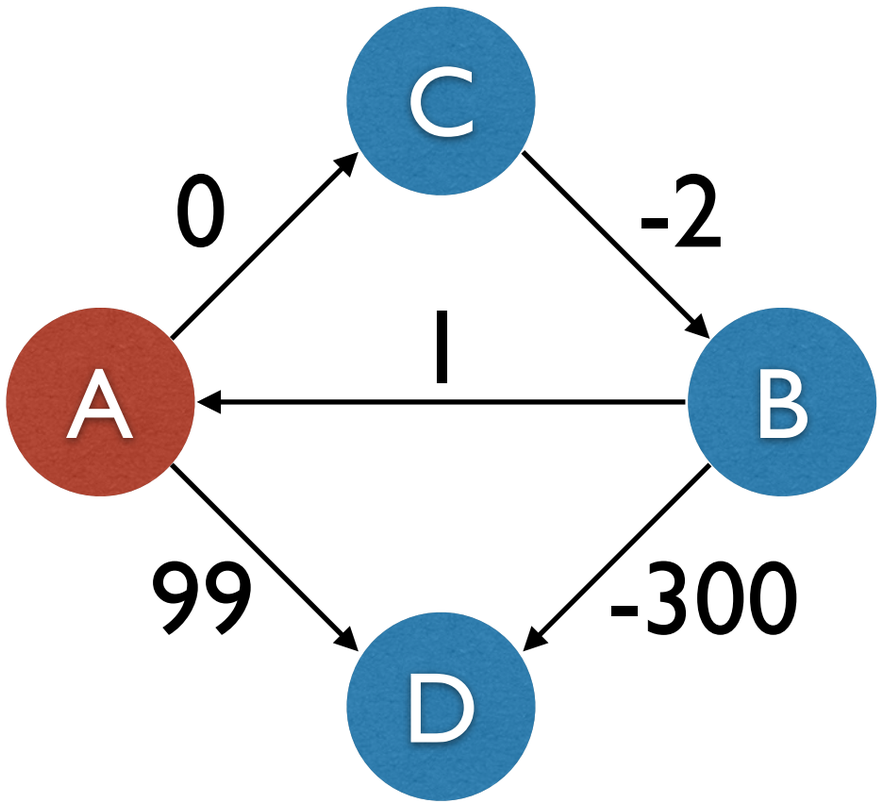


A	B	C	D
0	∞	0	99
0	-2	0	-302

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Example

- The third iteration

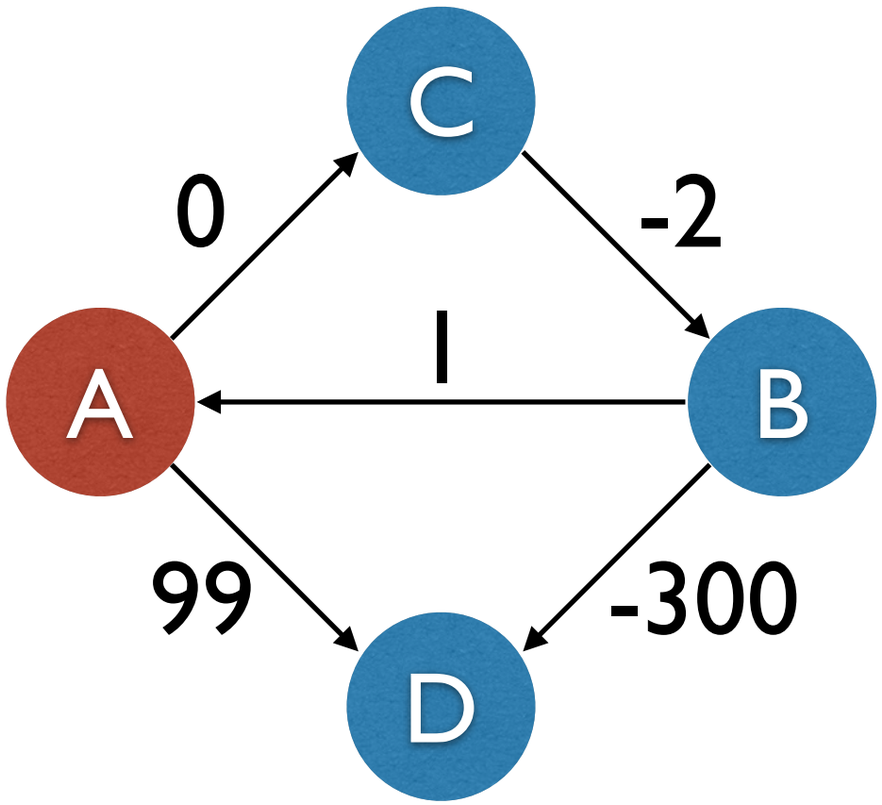


A	B	C	D
0	∞	0	99
0	-2	0	-302
-1	-2	0	-302

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Example

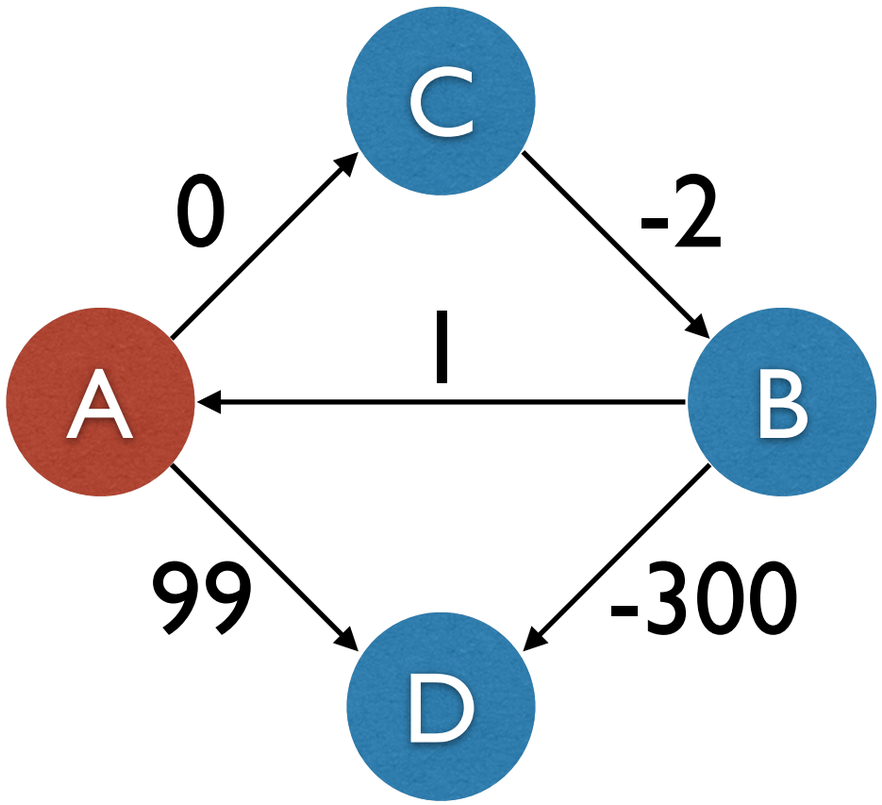
- The forth iteration



A	B	C	D
0	∞	0	99
0	-2	0	-302
-1	-2	0	-302
-1	-2	-1	-302

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Example



A	B	C	D
0	∞	0	99
0	-2	0	-302
-1	-2	0	-302
-1	-2	-1	-302

$d[B] > d[C]$ -2, thus negative-weight cycle exits

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Bellman-Ford algorithm: analysis

- **Proposition:** Dynamic programming algorithm computes SPT in any edge- weighted digraph with no negative cycles in time proportional to $E \times V$.

- **Practical Improvement**

  - Observation. If $d[v]$ does not change during pass i, no need to relax any edge pointing from v in pass $i+1$.

  - FIFO implementation. Maintain queue of vertices whose $d[]$ changed.

- **Overall effect**

  - The running time is still proportional to $E \times V$ in worst case.

  - But much faster than that in practice.

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Single source shortest-paths implementation: cost summary

algorithm	restriction	typical case	worst case	extra space
topological sort	no directed cycles	E + V	E + V	V
Dijkstra (binary heap)	no negative weights	E log V	E log V	V
Bellman-Ford	no negative cycles	E V	E V	V
Bellman-Ford (queue-based)	no negative cycles	E + V	E V	V

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

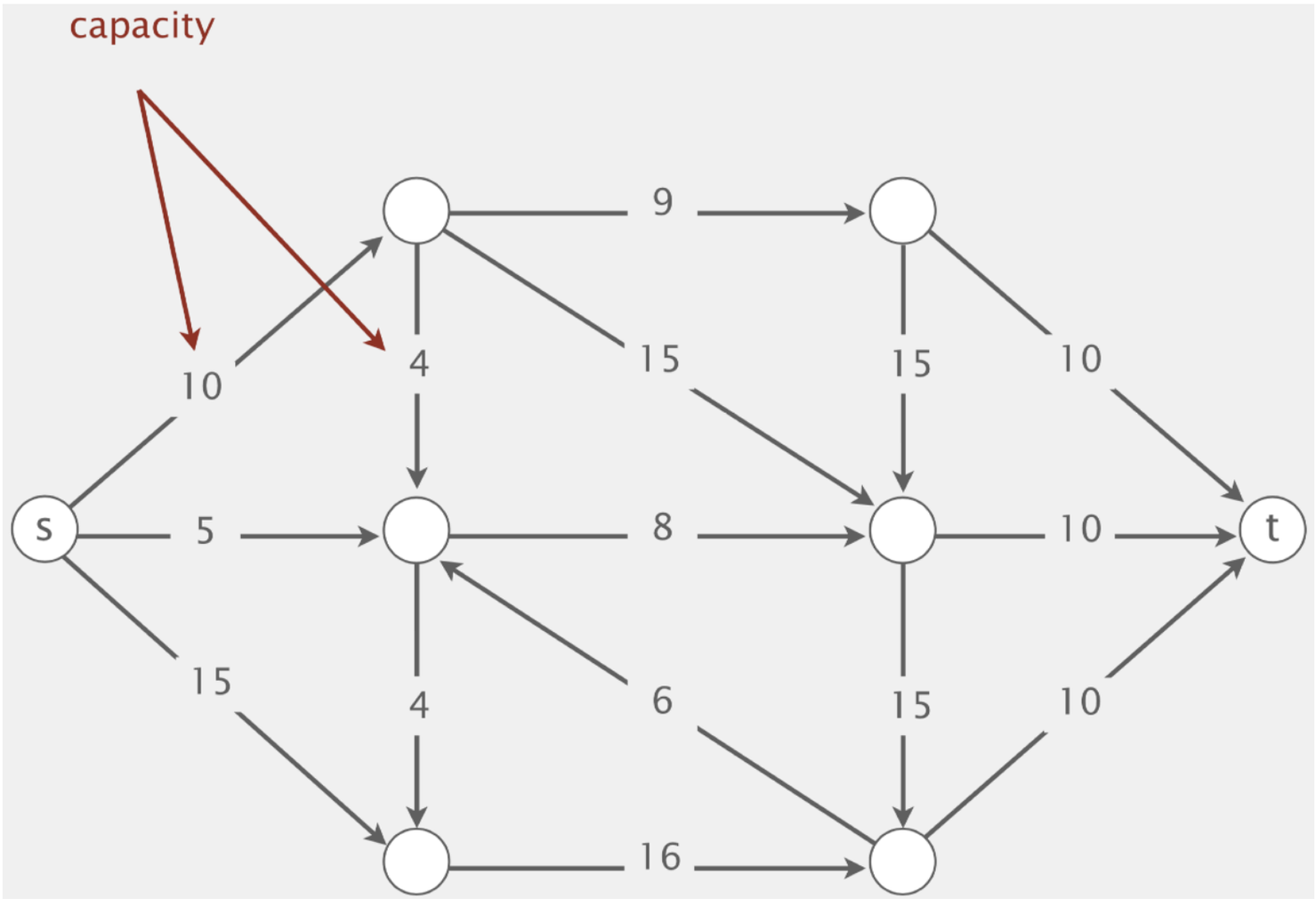
Outline

- The Single-Source Shortest Path Problem
- The Maximum Flow Problem
- The Maximum Cut Problem
- The Traveling Salesman Problem

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

The Mincut Problem

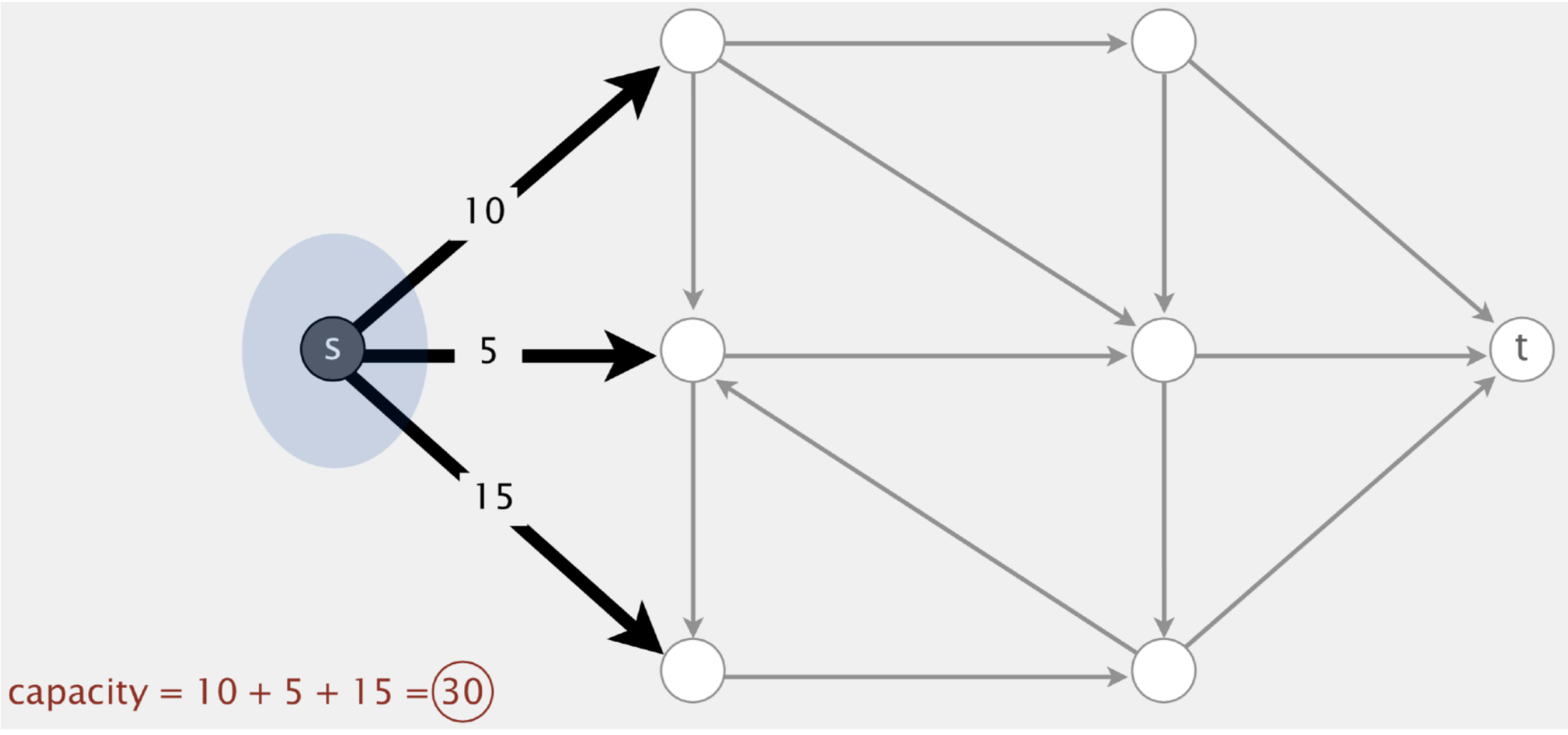
- Input. An edge-weighted digraph, source vertex *s*, and target vertex *t*.
  - each edge has a positive capacity



CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

The Mincut Problem

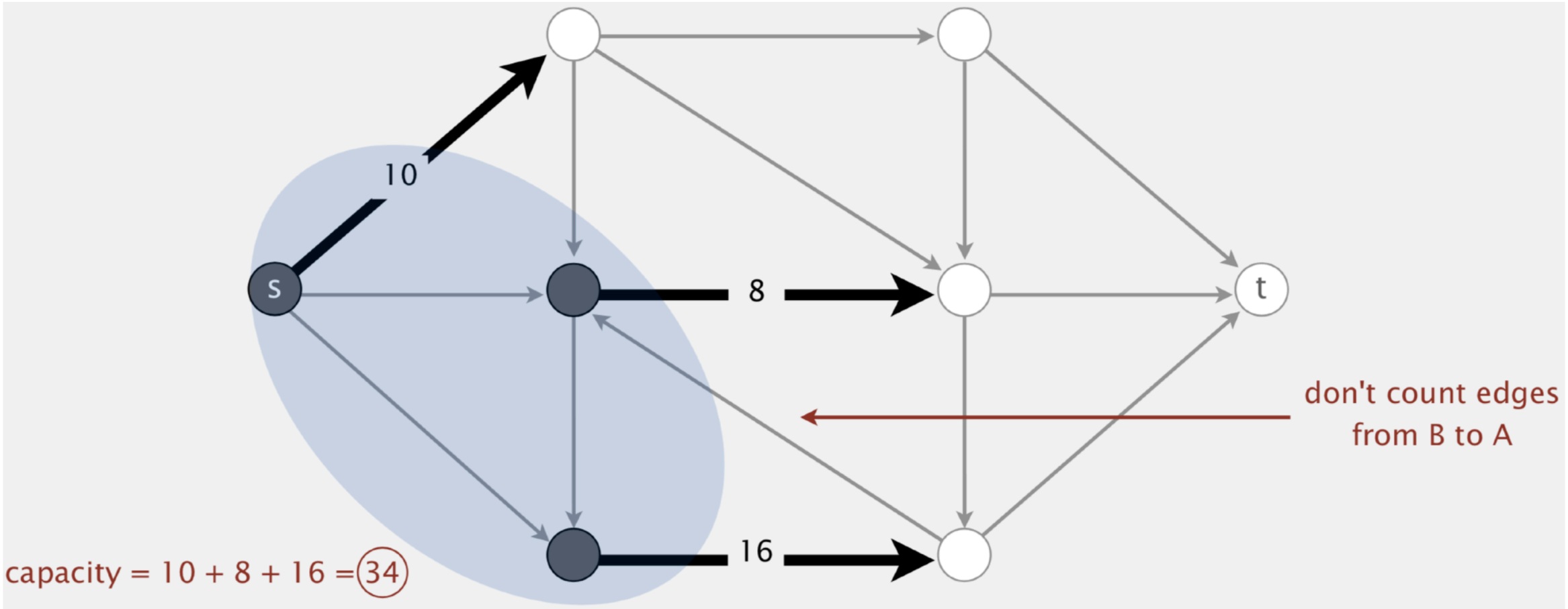
- A *st*-cut (cut) is a partition of the vertices into two disjoint sets, with *s* in one set *A* and *t* in the other set *B*.
- Its capacity is the sum of the capacities of the edges from *A* to *B*.



CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

The Mincut Problem

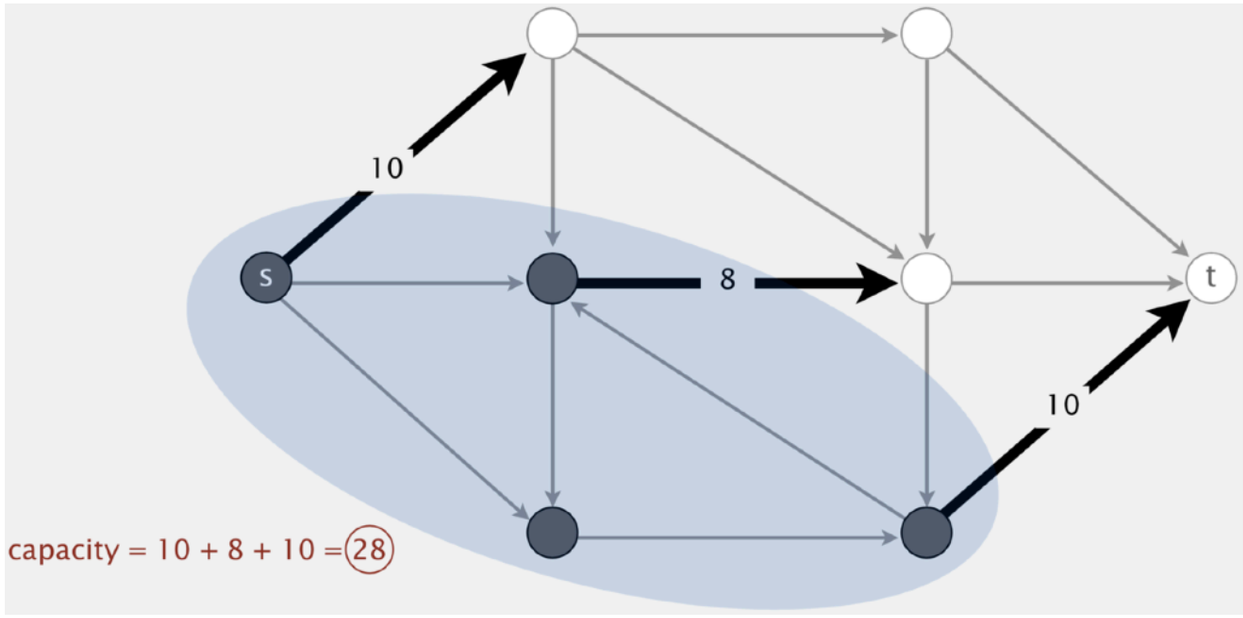
- A *st*-cut (cut) is a partition of the vertices into two disjoint sets, with *s* in one set *A* and *t* in the other set *B*.
- Its capacity is the sum of the capacities of the edges from *A* to *B*.



CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

The Mincut Problem

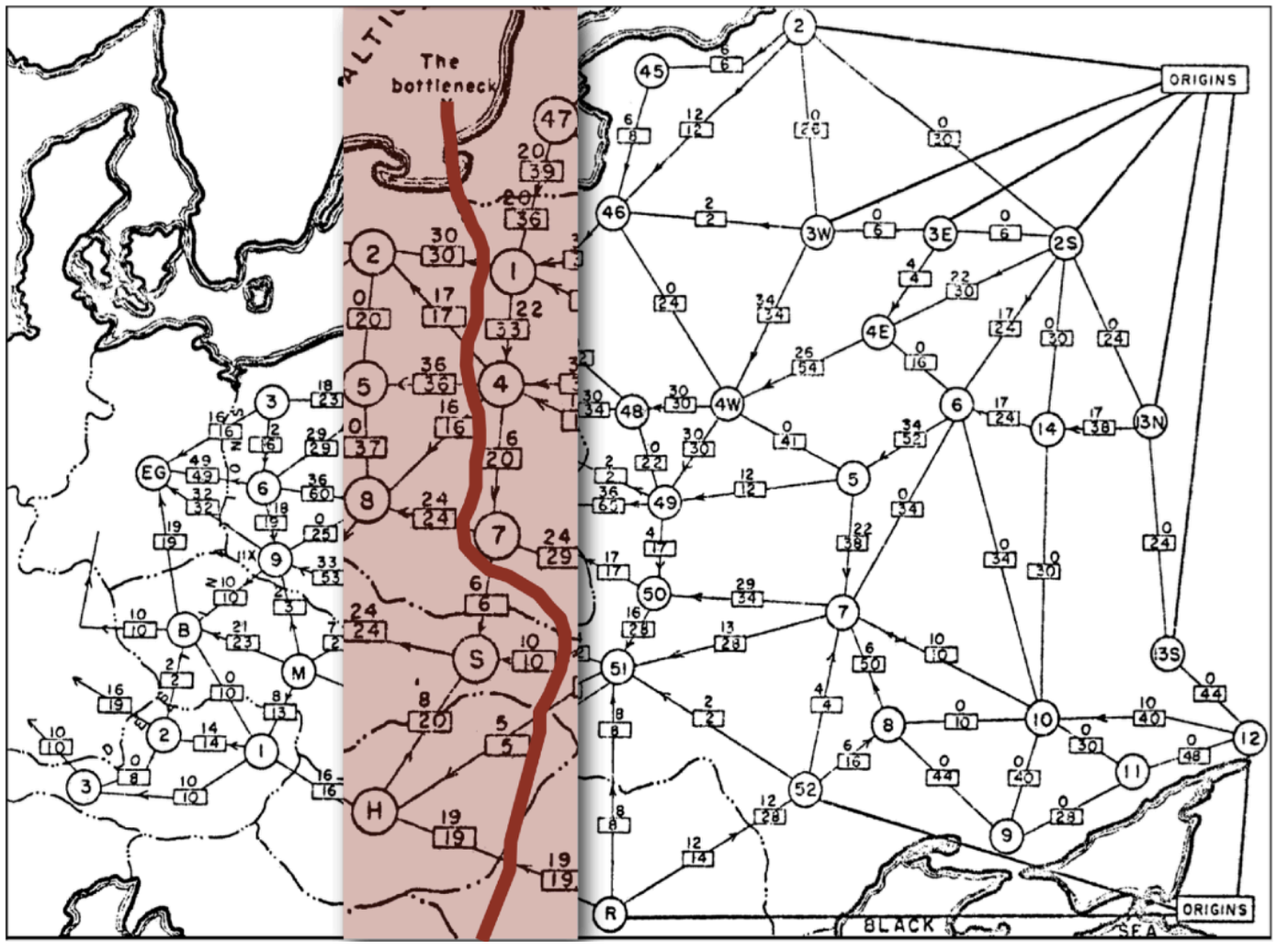
- A *st*-cut (cut) is a partition of the vertices into two disjoint sets, with *s* in one set *A* and *t* in the other set *B*.
- Its capacity is the sum of the capacities of the edges from *A* to *B*.
- Minimum st-cut (mincut) problem. Find a cut of minimum capacity.



CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Mincut Applications

- "Free world" goal. Cut supplies (if cold war turns into real war).



- rail network connecting Soviet Union with Eastern European countries (map declassified by Pentagon in 1999)

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

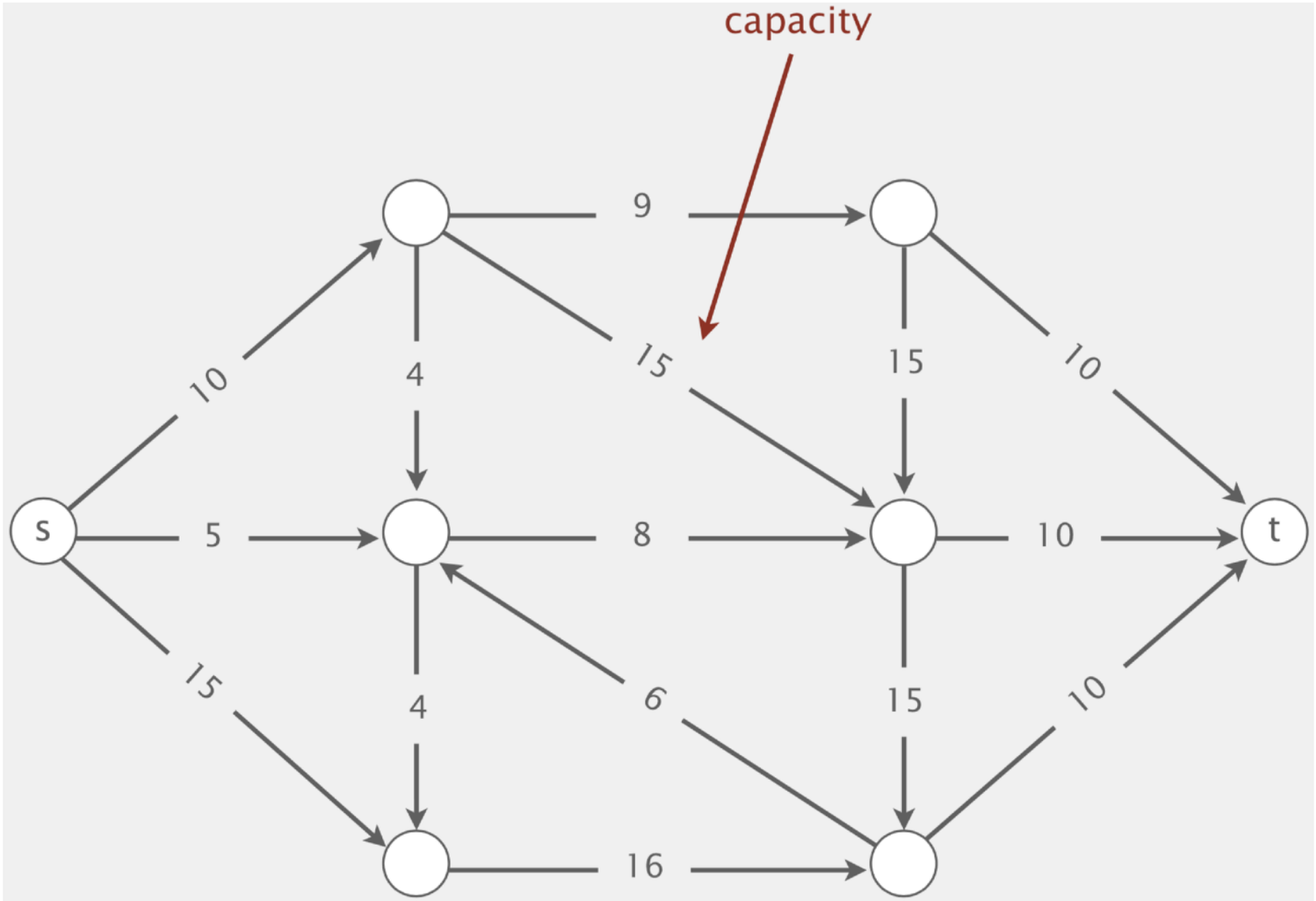
Mincut Applications

- Government-in-power's goal. Cut off communication to set of people.

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

The Maxflow Problem

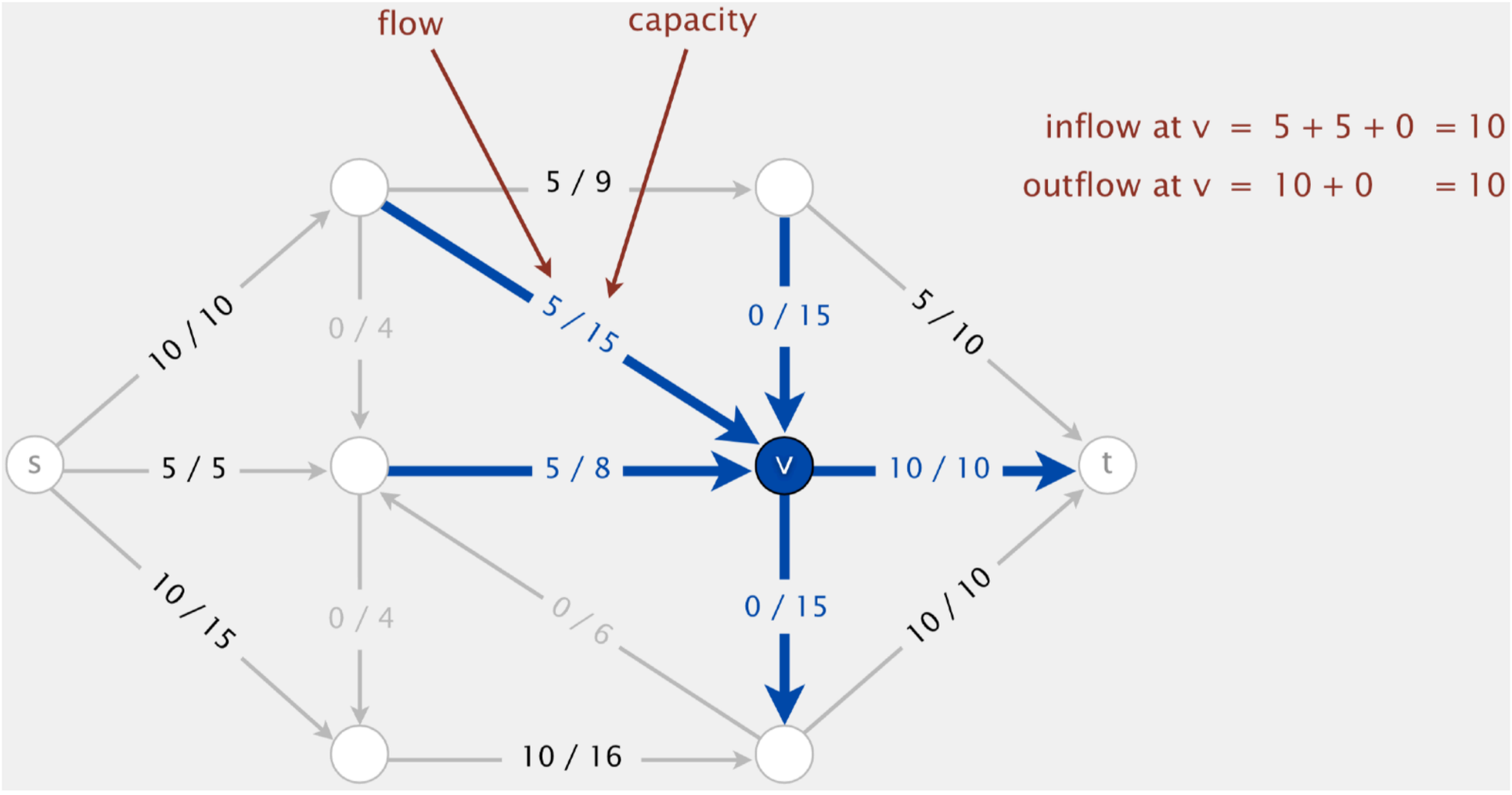
- Input. An edge-weighted digraph, source vertex *s*, and target vertex *t*.
  - each edge has a positive capacity



CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

The Max-Flow Problem

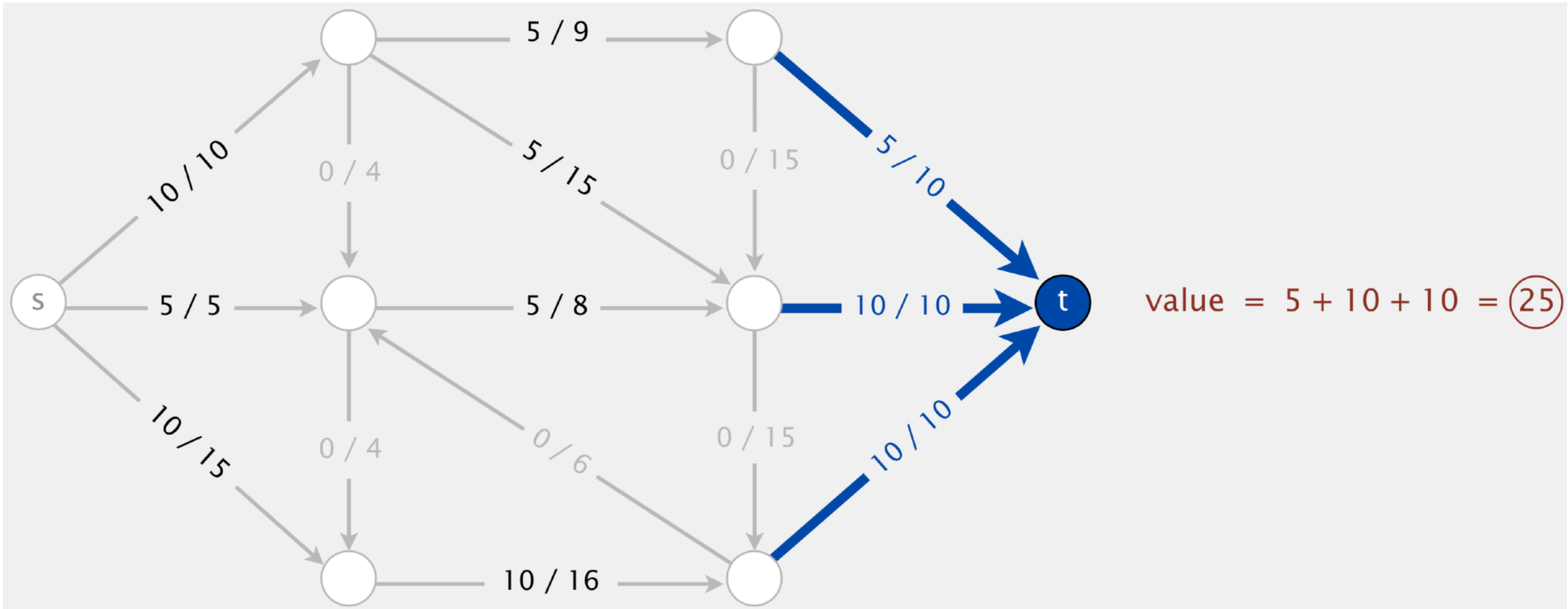
- An *st*-flow (flow) is an assignment of values to the edges such that:
  - Capacity constraint: $0 \leq$ edge's flow $\leq$ edge's capacity.
  - Local equilibrium: inflow = outflow at every vertex (except *s* and *t*).



CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

The Max-Flow Problem

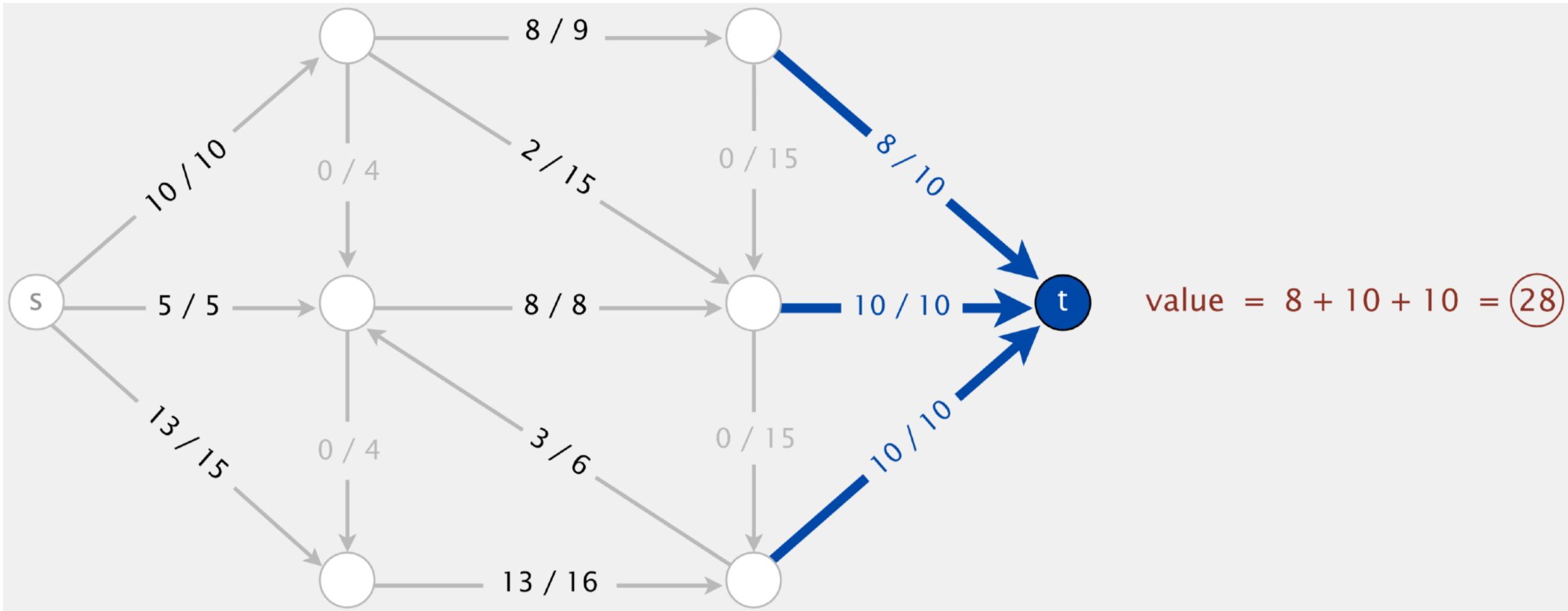
- The value of a flow is the inflow at *t*.
  - we assume no edge points to s or from t



CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

The Max-Flow Problem

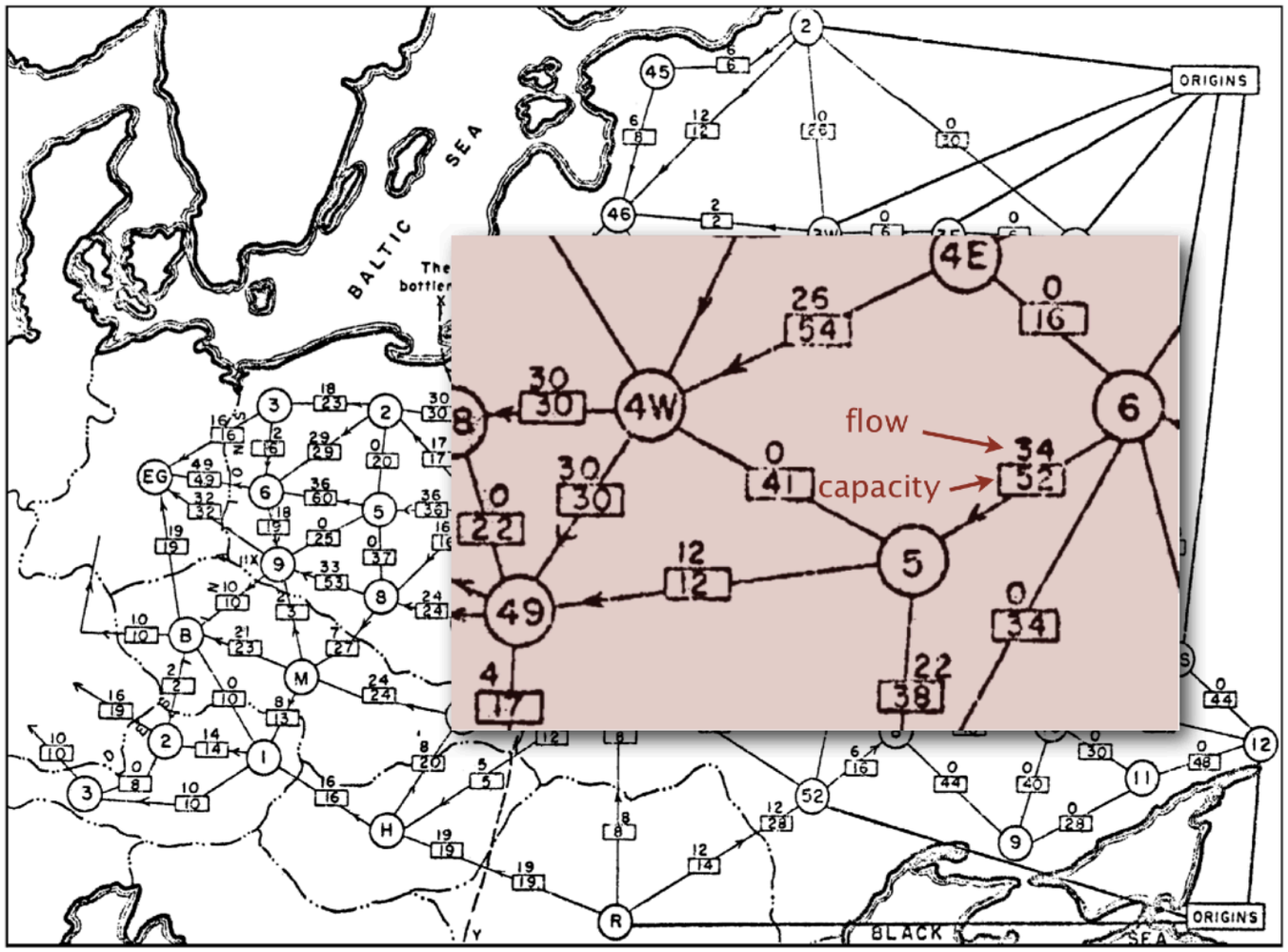
- The value of a flow is the inflow at *t*.
  - we assume no edge points to s or from t
- Maximum st-flow (maxflow) problem. Find a flow of maximum value.



CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Maxflow Applications

- Soviet Union goal. Maximize flow of supplies to Eastern Europe.



- rail network connecting Soviet Union with Eastern European countries (map declassified by Pentagon in 1999)

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

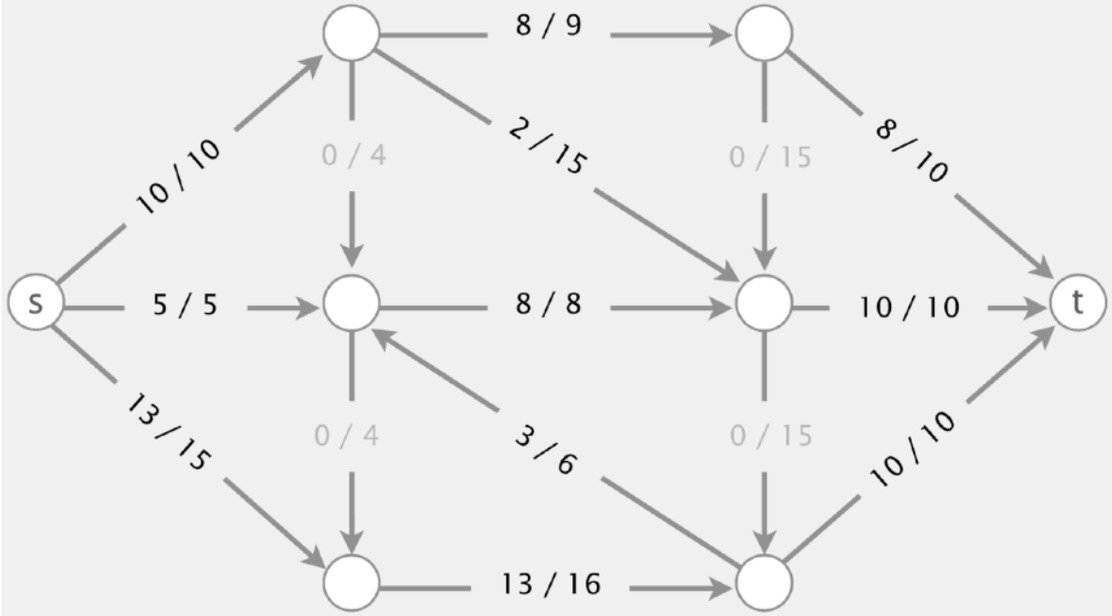
Maxflow Applications

- "Free world" goal. Maximize flow of information to specified set of people.

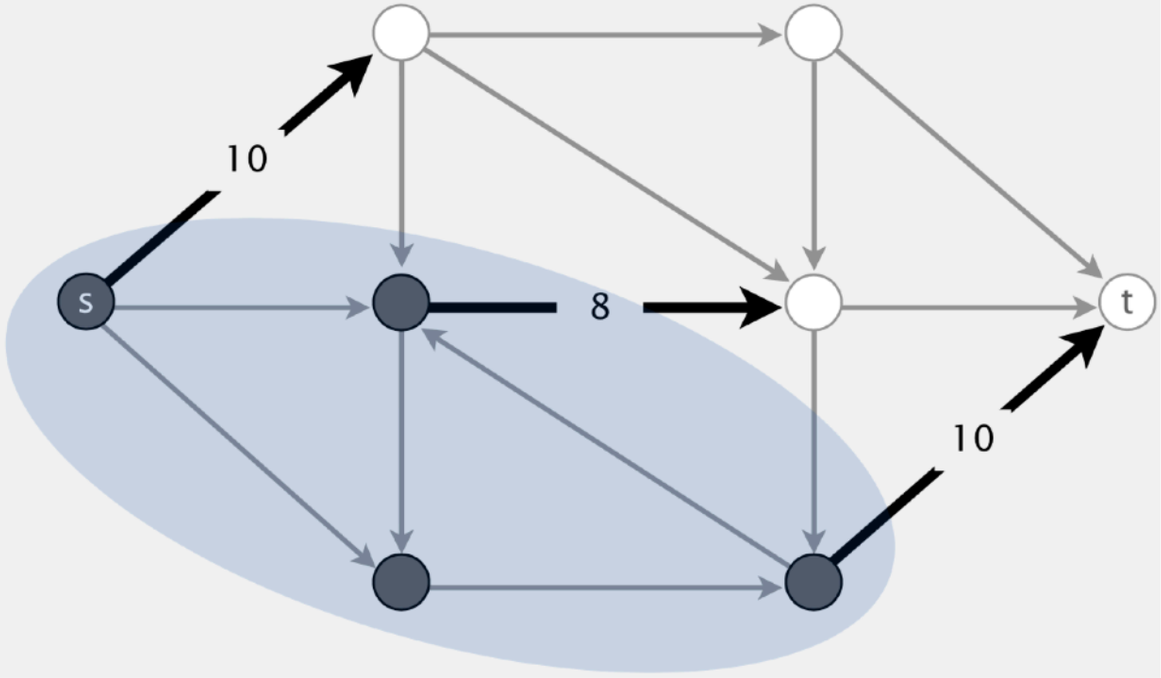
CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Summary

- Input. A weighted digraph, source vertex *s*, and target vertex *t*.
- Mincut problem. Find a cut of minimum capacity.
- Maxflow problem. Find a flow of maximum value.



value of flow = 28



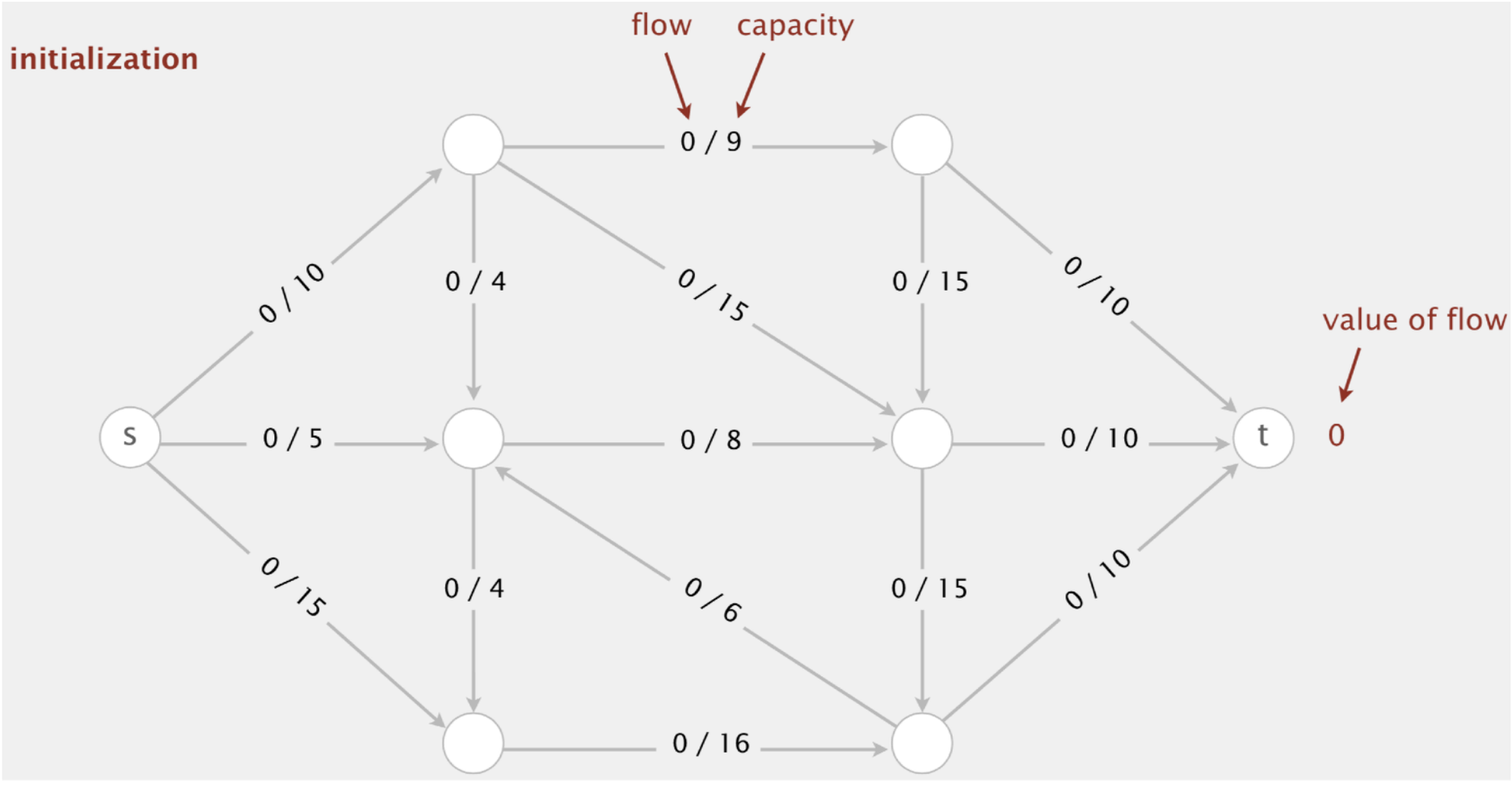
capacity of cut = 28

These two problems are dual!

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Ford-Fulkerson Algorithm

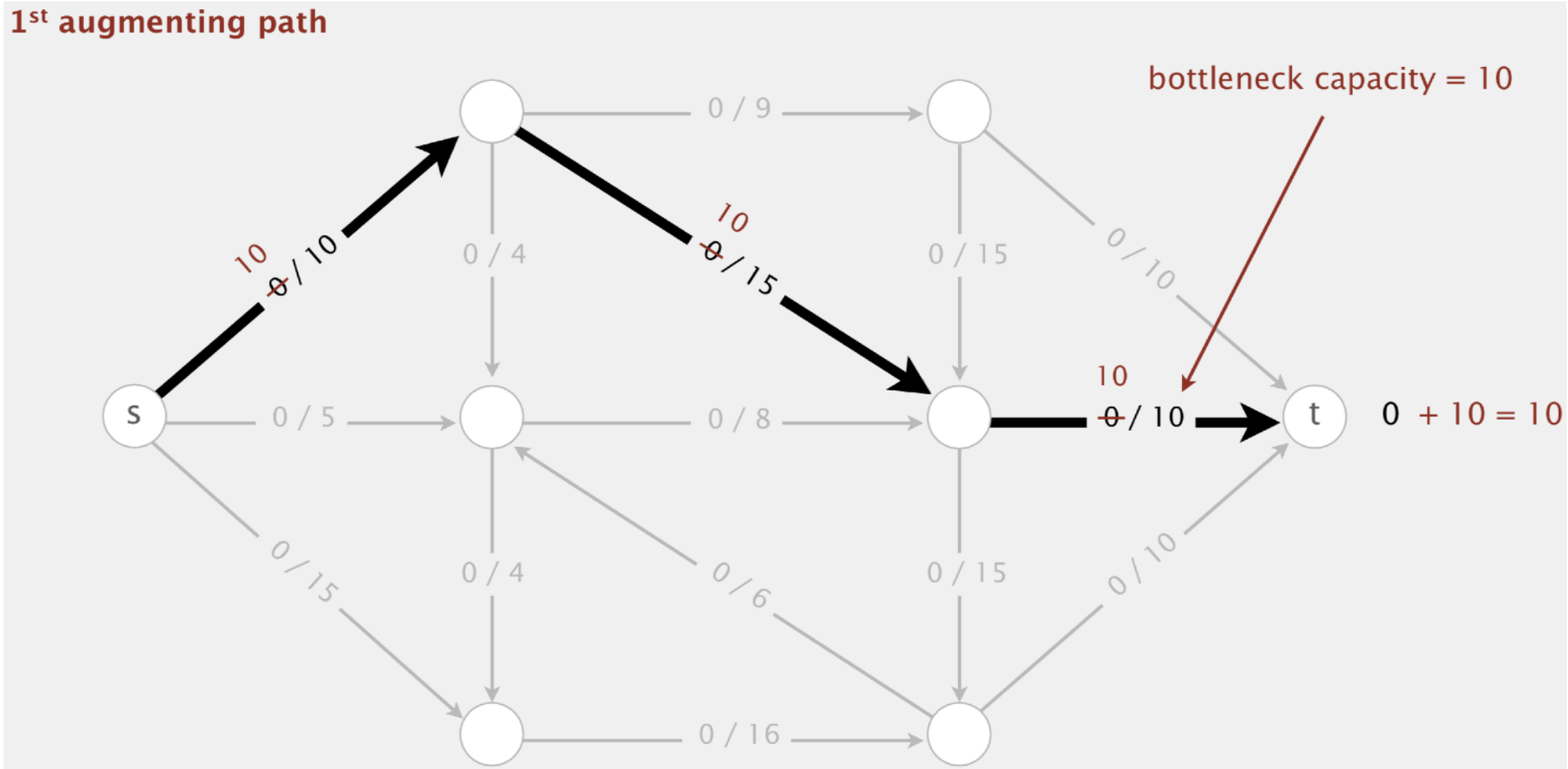
- Initialization. Start with 0 flow.



CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Ford-Fulkerson Algorithm

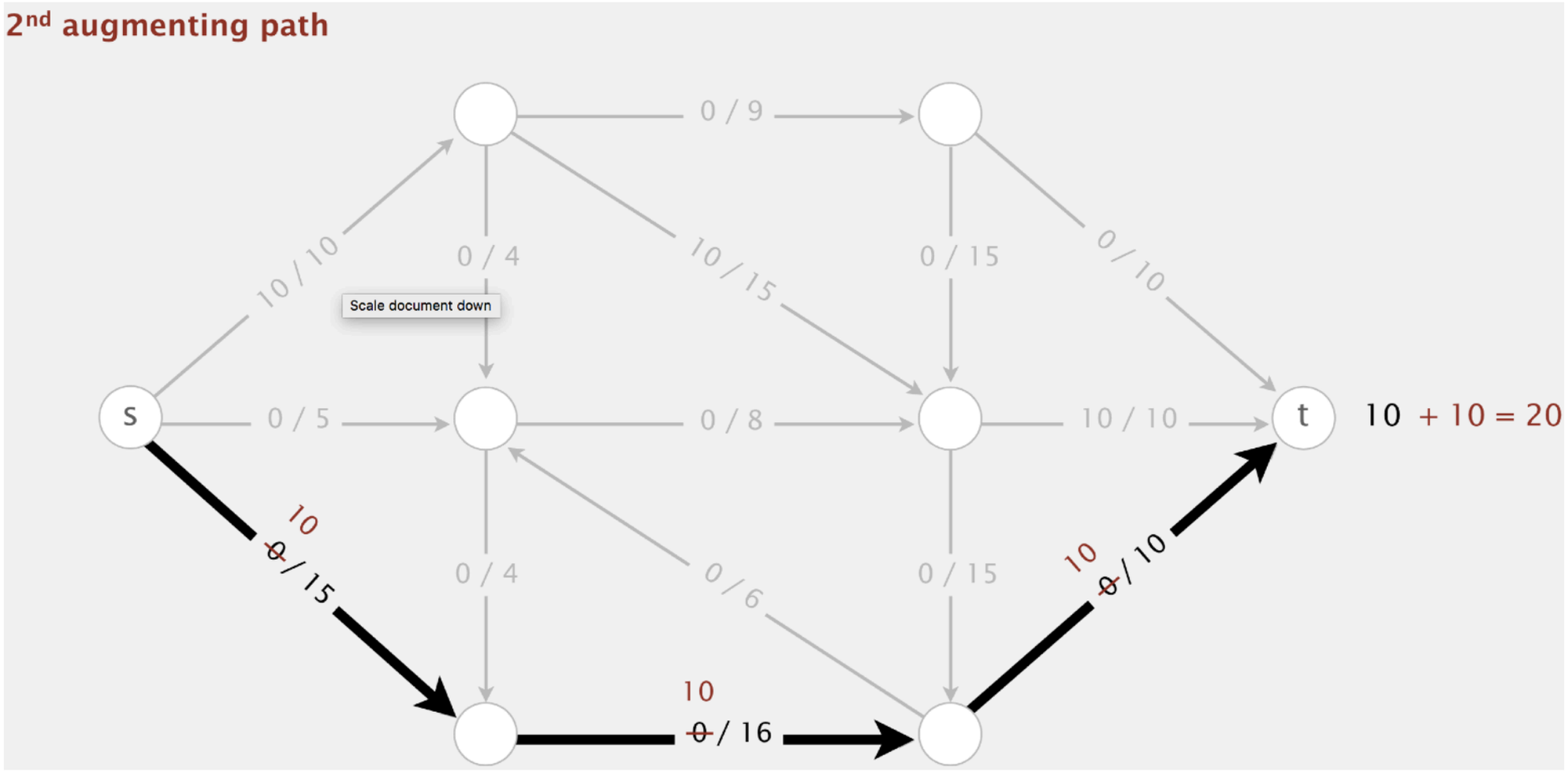
- Idea: increase flow along augmenting paths
- Augmenting path. Find an undirected path from *s* to *t* such that:
  - Can increase flow on forward edges (not full).
  - Can decrease flow on backward edge (not empty).



CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Ford-Fulkerson Algorithm

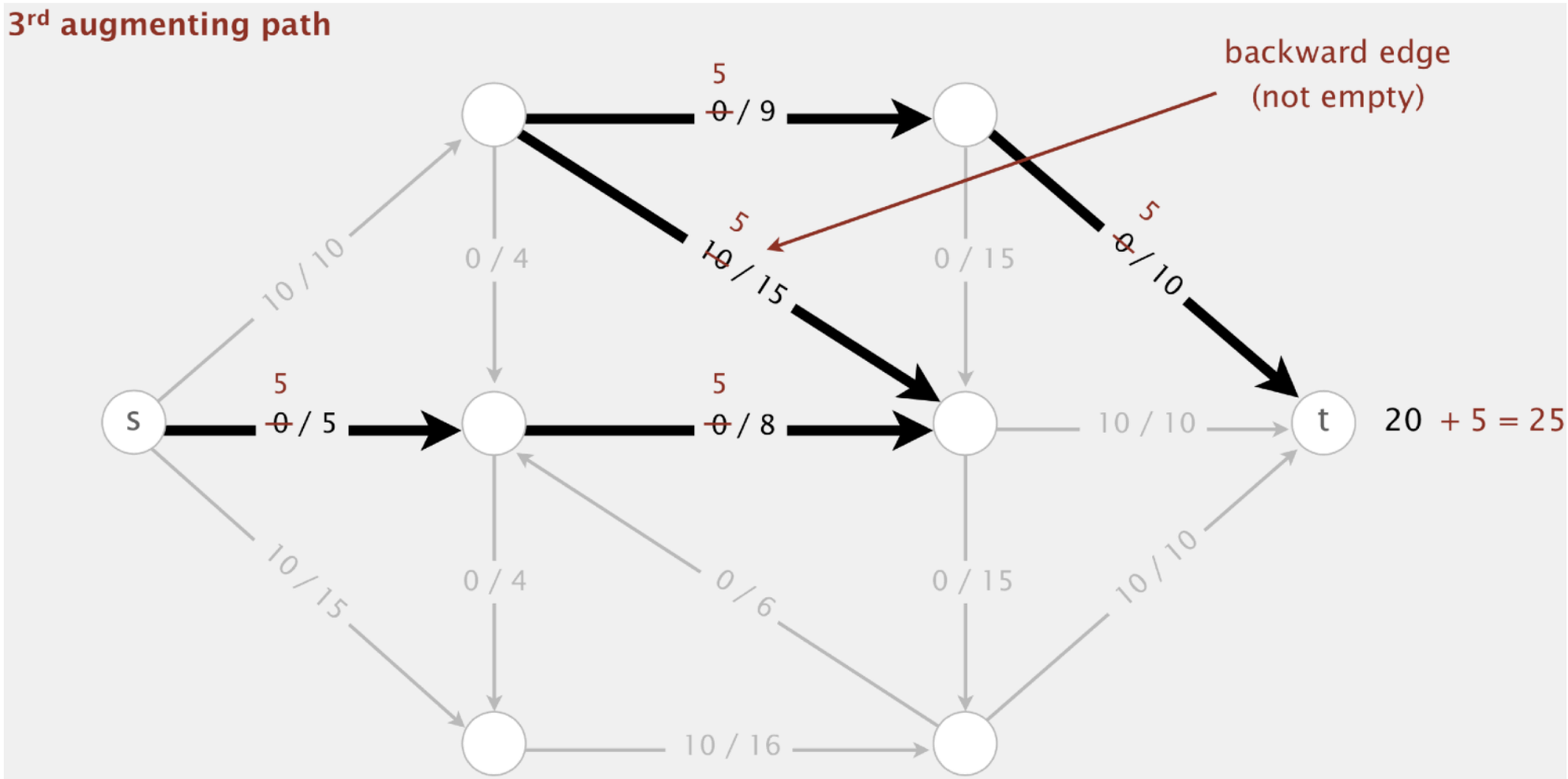
- Idea: increase flow along augmenting paths
- Augmenting path. Find an undirected path from *s* to *t* such that:
  - Can increase flow on forward edges (not full).
  - Can decrease flow on backward edge (not empty).



CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Ford-Fulkerson Algorithm

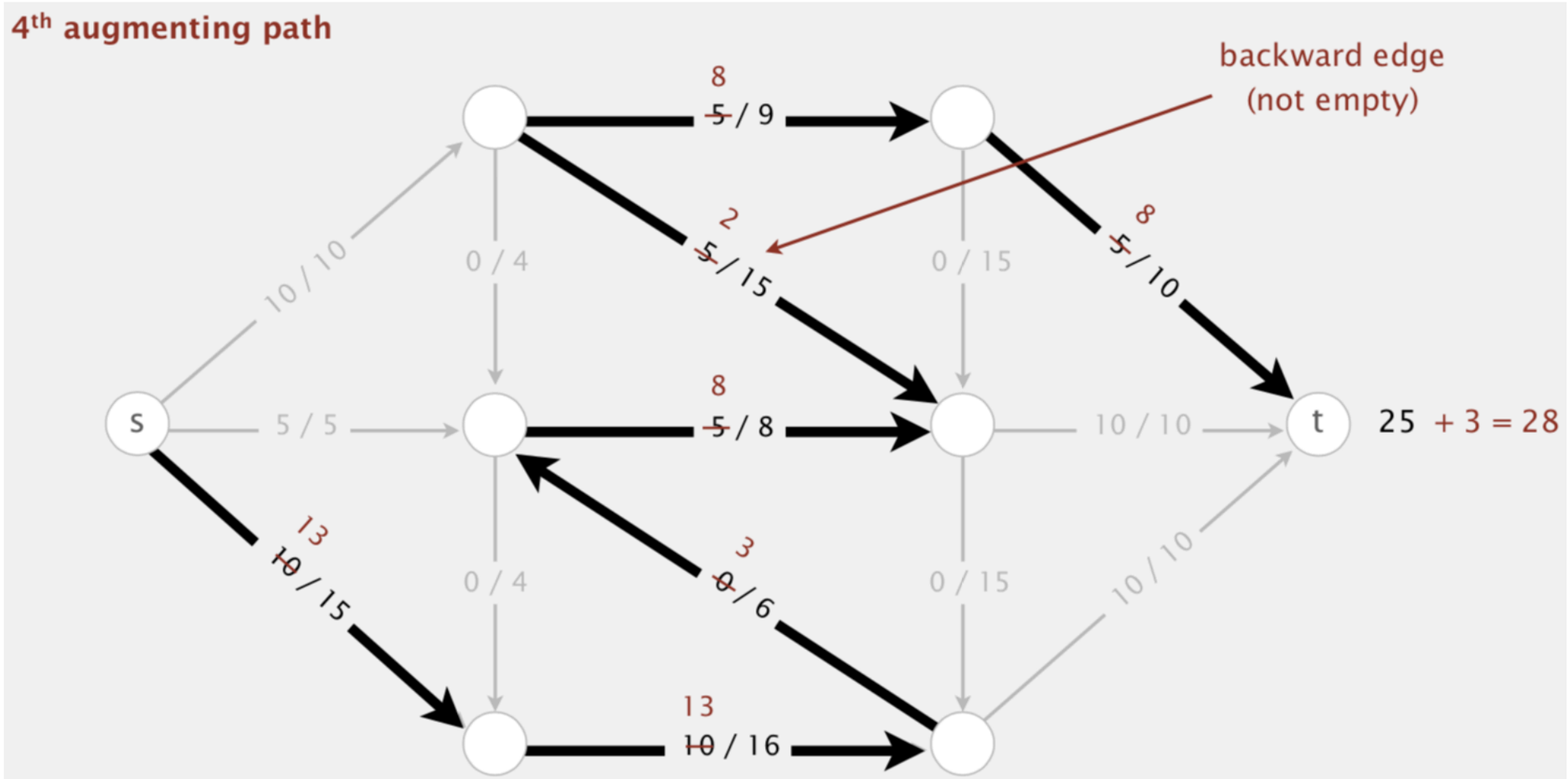
- Idea: increase flow along augmenting paths
- Augmenting path. Find an undirected path from *s* to *t* such that:
  - Can increase flow on forward edges (not full).
  - Can decrease flow on backward edge (not empty).



CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Ford-Fulkerson Algorithm

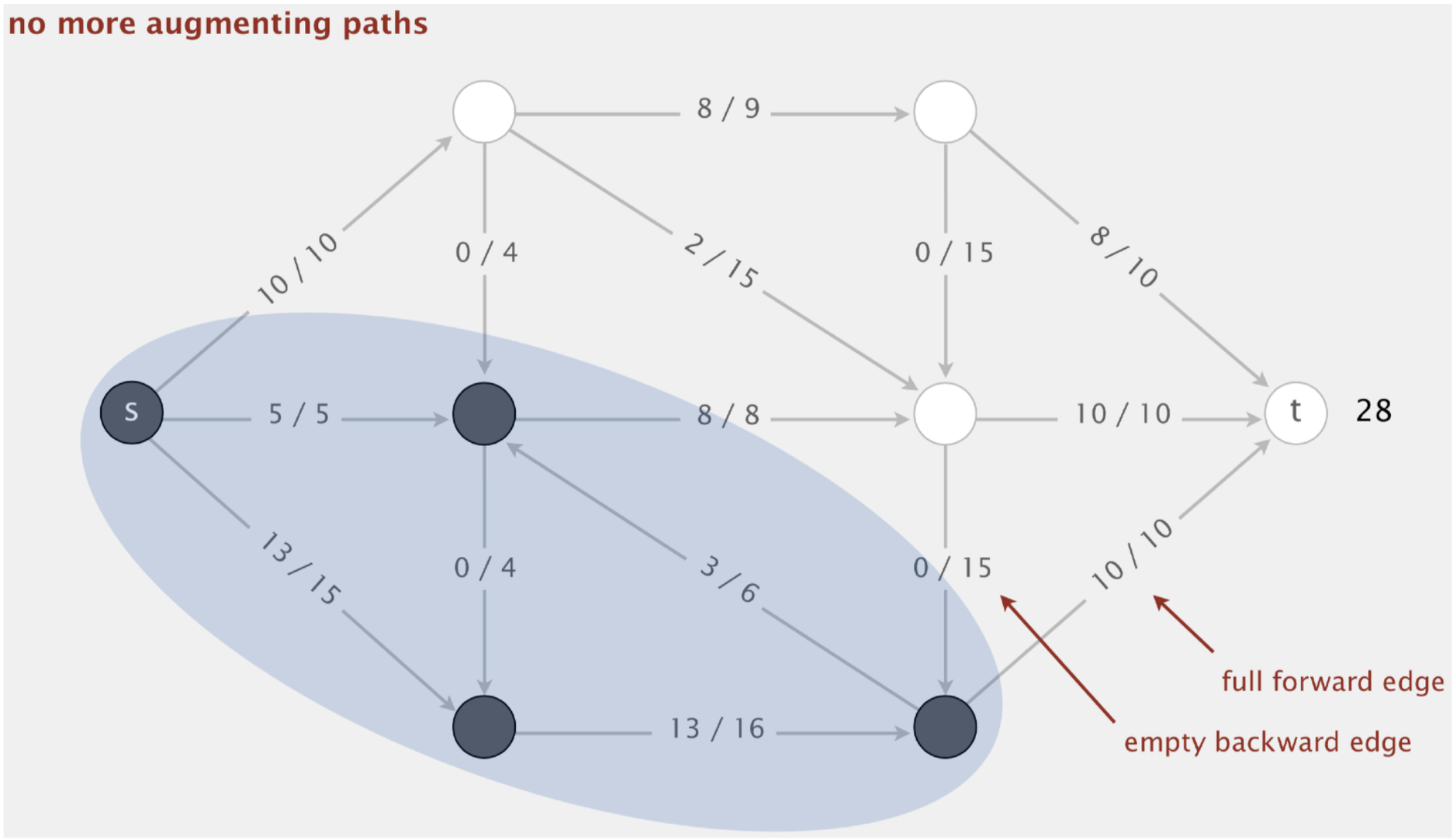
- Idea: increase flow along augmenting paths
- Augmenting path. Find an undirected path from *s* to *t* such that:
  - Can increase flow on forward edges (not full).
  - Can decrease flow on backward edge (not empty).



CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Ford-Fulkerson Algorithm

- Termination. All paths from *s* to *t* are blocked by either a
  - Full forward edge.
  - Empty backward edge.



CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Ford-Fulkerson Algorithm

- Start with 0 flow.
- While there exists an augmenting path:
  - find an augmenting path
  - compute bottleneck capacity
  - increase flow on that path by bottleneck capacity

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

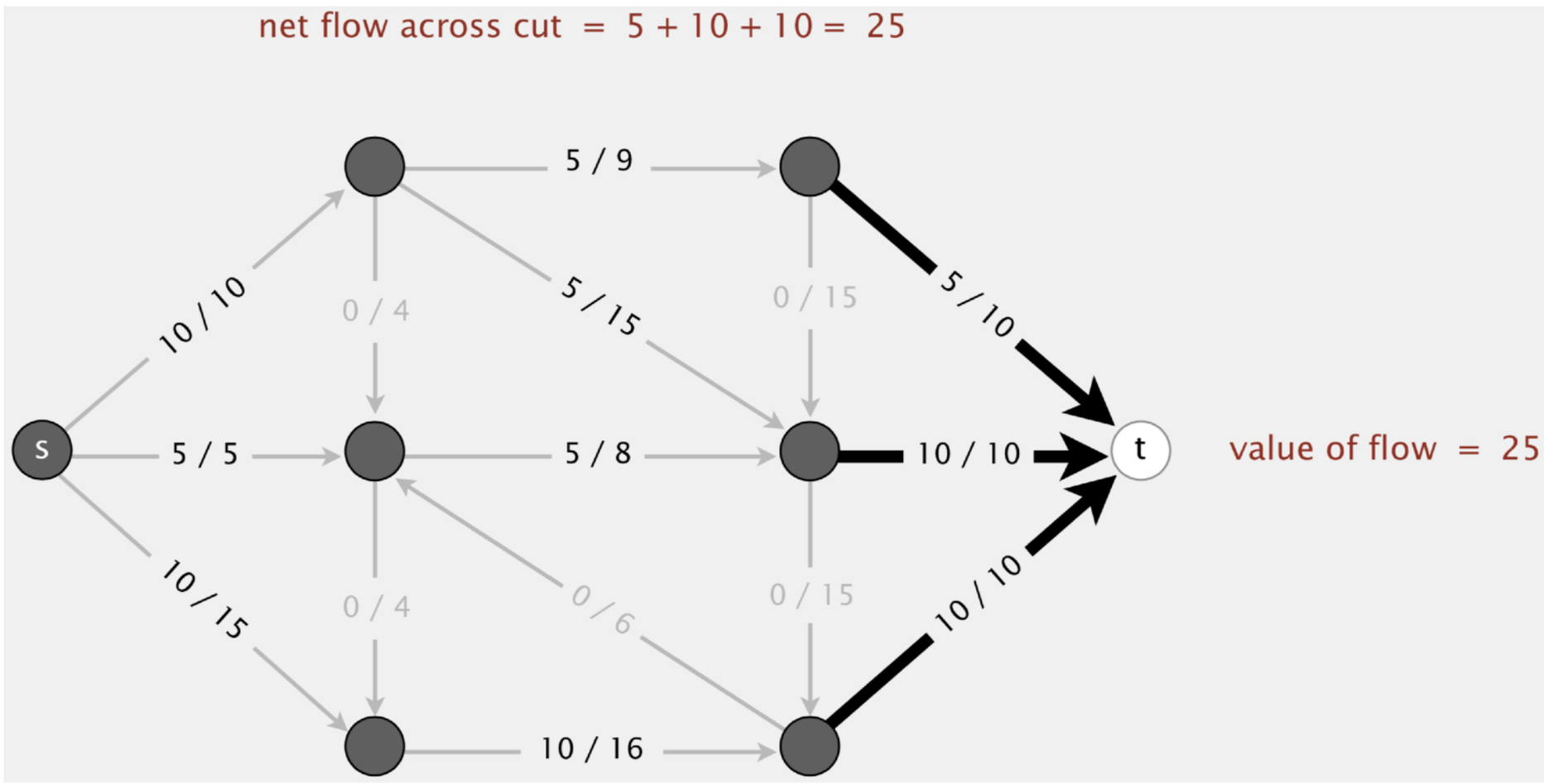
Questions

1. How to compute a mincut?
2. How to find an augmenting path?
3. If FF terminates, does it always compute a maxflow?
4. Does FF always terminate? If so, after how many augmentations?

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

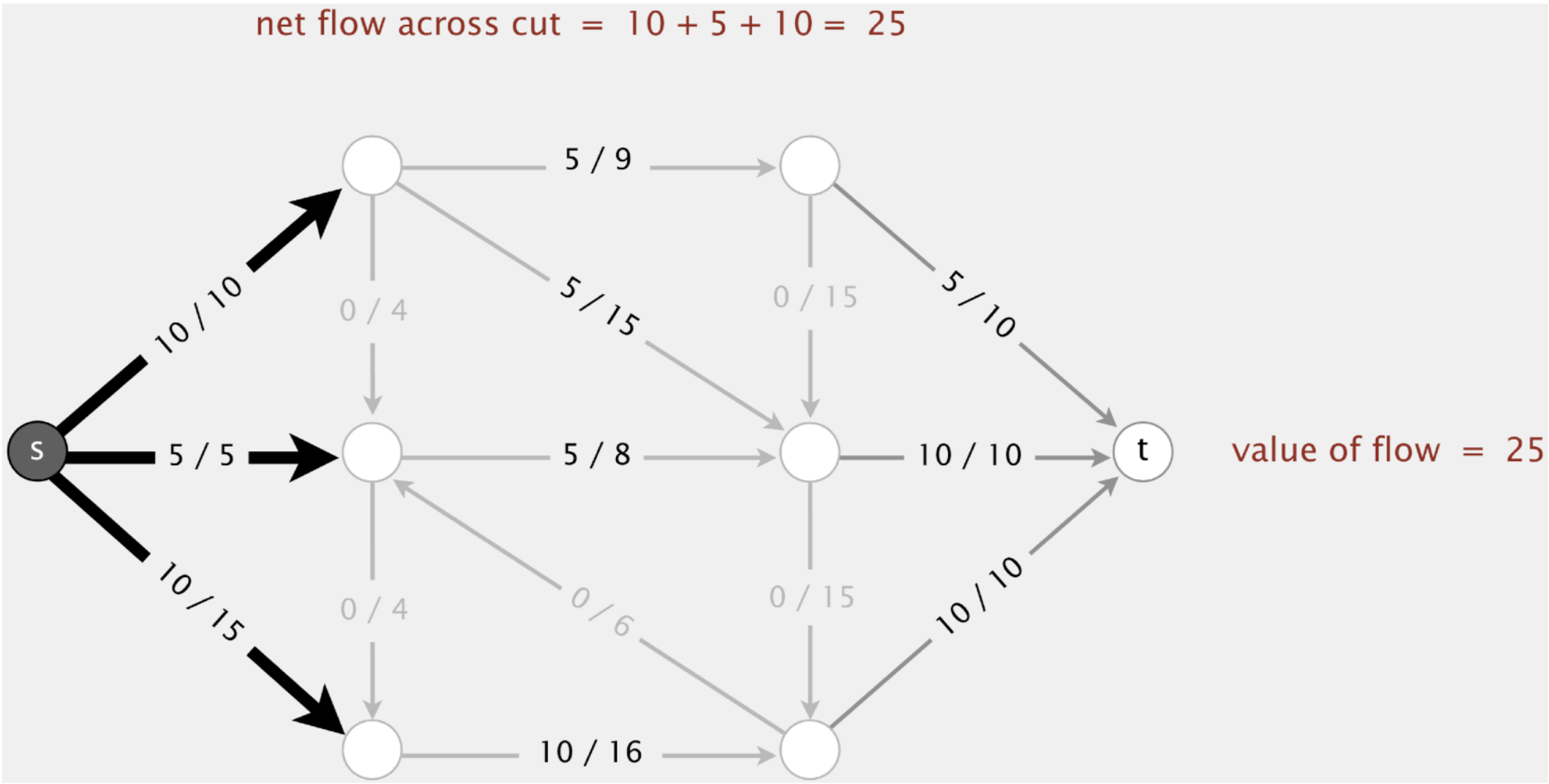
Relationship between flows and cuts

- The net flow across a cut (A, B) is the sum of the flows on its edges from A to B minus the sum of the flows on its edges from B to A.
- Flow-value lemma. Let f be any flow and let (A, B) be any cut. Then, the net flow across (A, B) equals the value of f.



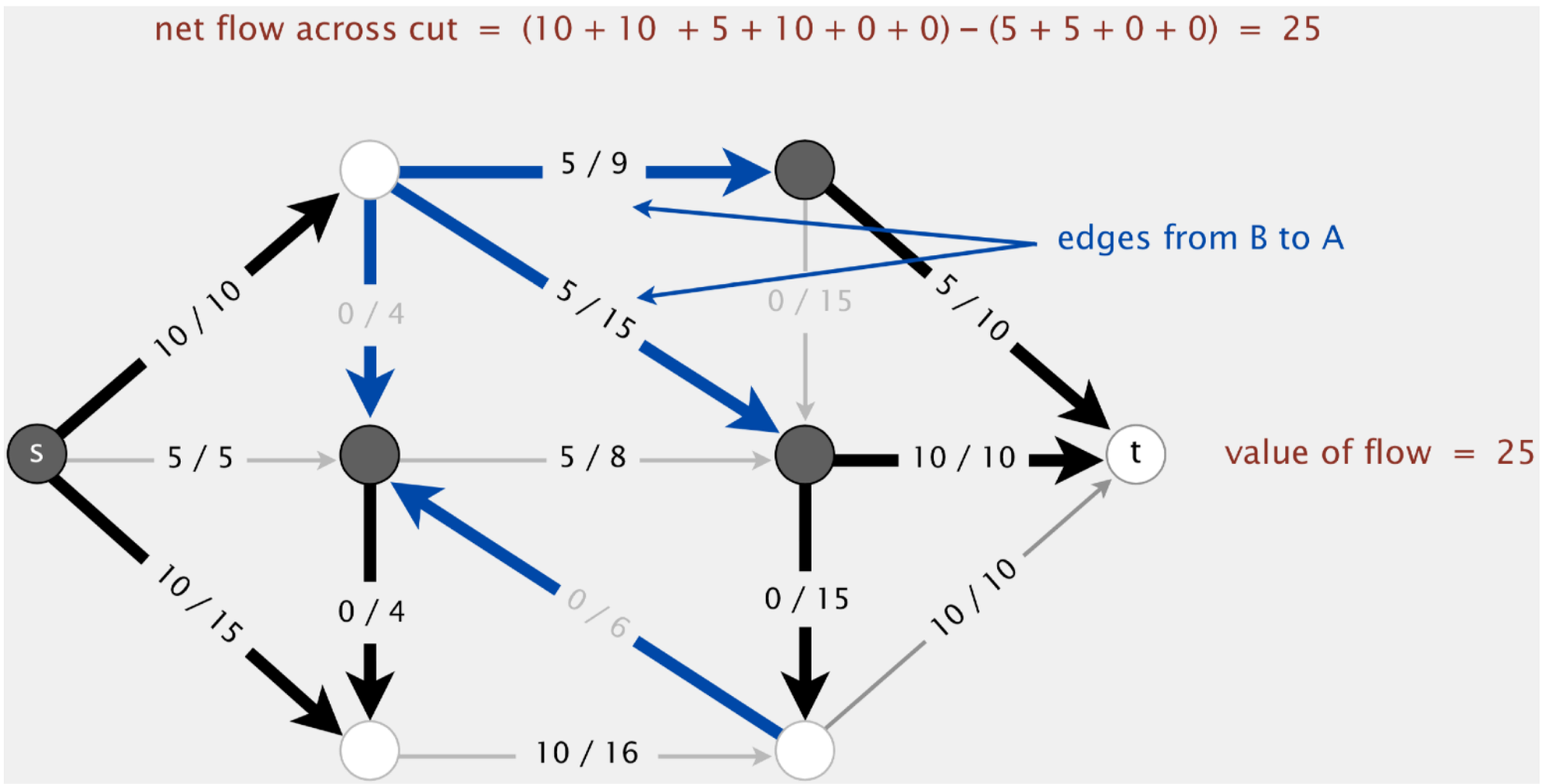
Relationship between flows and cuts

- The net flow across a cut (A, B) is the sum of the flows on its edges from A to B minus the sum of the flows on its edges from B to A.
- Flow-value lemma. Let f be any flow and let (A, B) be any cut. Then, the net flow across (A, B) equals the value of f.



Relationship between flows and cuts

- The net flow across a cut (A, B) is the sum of the flows on its edges from A to B minus the sum of the flows on its edges from B to A.
- Flow-value lemma. Let f be any flow and let (A, B) be any cut. Then, the net flow across (A, B) equals the value of f.



Relationship between flows and cuts

- The net flow across a cut (A, B) is the sum of the flows on its edges from A to B minus the sum of the flows on its edges from B to A.
- Flow-value lemma. Let f be any flow and let (A, B) be any cut. Then, the net flow across (A, B) equals the value of f.
- Pf. By induction on the size of B.
  - Base case: $B = \{ t \}$.
  - Induction step: remains true by local equilibrium when moving any vertex from A to B.
- Corollary. Outflow from s = inflow to t = value of flow.

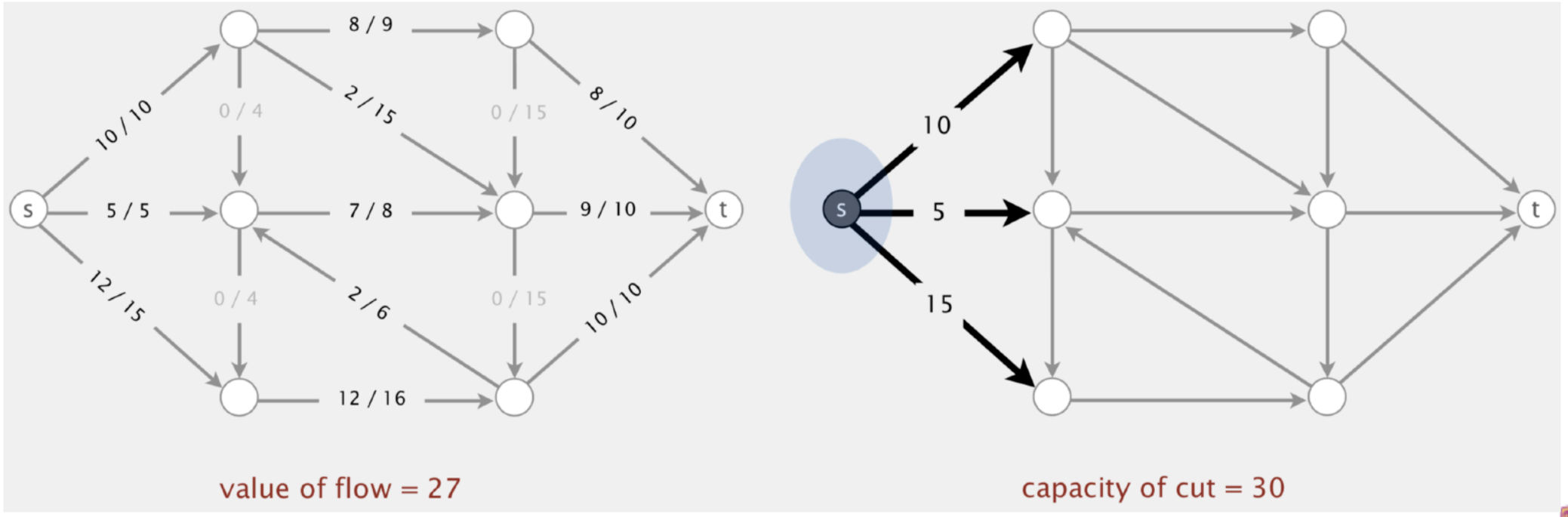
CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Relationship between flows and cuts

- Weak duality. Let f be any flow and let (A, B) be any cut. Then, the value of the flow ≤ the capacity of the cut.

- Pf. Value of flow f = net flow across cut (A, B) ≤ capacity of cut (A, B).

flow-value lemma — flow bounded by capacity



CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Maxflow-mincut theorem

- Augmenting path theorem: A flow f is a maxflow iff no augmenting paths.
- Maxflow-mincut theorem: Value of the maxflow = capacity of mincut.

Pf. The following three conditions are equivalent for any flow f :

1. There exists a cut whose capacity equals the value of the flow f.
2. f is a maxflow.
3. There is no augmenting path with respect to f.

[1 ⇒ 2]

- Suppose that (A, B) is a cut with capacity equal to the value of f.
- Then, the value of any flow f' ≤ capacity of (A, B) = value of f.
- Thus, f is a maxflow.

weak duality

by assumption

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Maxflow-mincut theorem

- Augmenting path theorem: A flow *f* is a maxflow iff no augmenting paths.
- Maxflow-mincut theorem: Value of the maxflow = capacity of mincut.

Pf. The following three conditions are equivalent for any flow *f* :

1. There exists a cut whose capacity equals the value of the flow *f*.
2. *f* is a maxflow.
3. There is no augmenting path with respect to *f*.

[2 ⇒ 3] We prove contrapositive: ~3 ⇒ ~2.

- Suppose that there is an augmenting path with respect to *f*.
- Can improve flow *f* by sending flow along this path.
- Thus, *f* is not a maxflow.

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Maxflow-mincut theorem

- Augmenting path theorem: A flow f is a maxflow iff no augmenting paths.
- Maxflow-mincut theorem: Value of the maxflow = capacity of mincut.

Pf. The following three conditions are equivalent for any flow f :

1. There exists a cut whose capacity equals the value of the flow f.
2. f is a maxflow.
3. There is no augmenting path with respect to f.

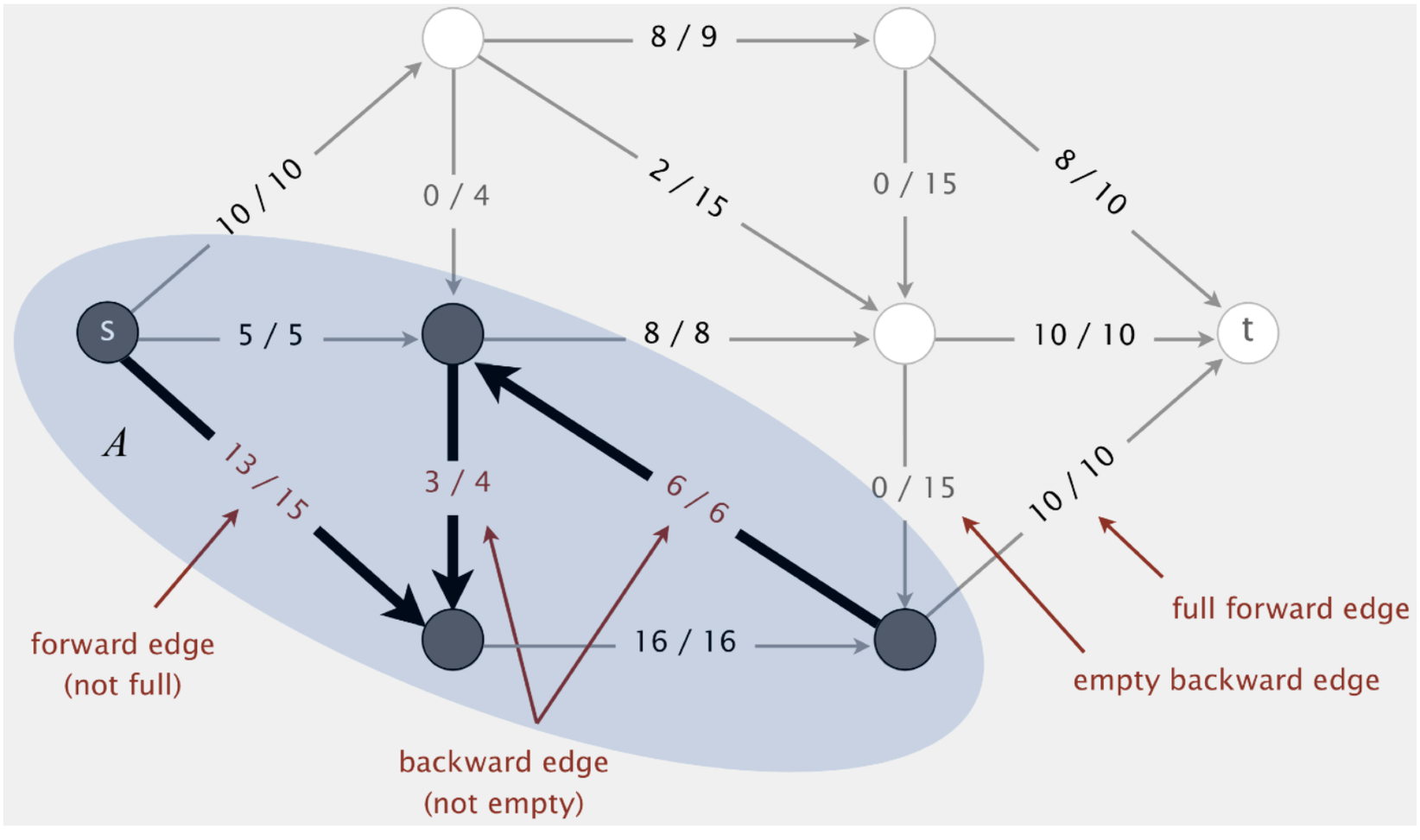
[3 ⇒ 1] Suppose that there is no augmenting path with respect to f.

- Let (A, B) be a cut where A is the set of vertices connected to s by an undirected path with no full forward or empty backward edges.
- By definition, s is in A; since no augmenting path, t is in B.
- Capacity of cut = net flow across cut ← forward edges full; backward edges empty
  = value of flow f. ← flow-value lemma

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Computing a mincut from a maxflow

- To compute mincut (*A*, *B*) from maxflow *f* :
  - By augmenting path theorem, no augmenting paths with respect to *f*.
  - Compute *A* = set of vertices connected to *s* by an undirected path with no full forward or empty backward edges.



CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Questions

1. How to compute a mincut? Easy. ✔
2. How to find an augmenting path? BFS works well.
3. If FF terminates, does it always compute a maxflow? Yes. ✔
4. Does FF always terminate? If so, after how many augmentations? yes, provided edge capacities are integers (or augmenting paths are chosen carefully). It requires clever analysis.

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Ford-Fulkerson algorithm with integer capacities

- Important special case. Edge capacities are integers between 1 and U.

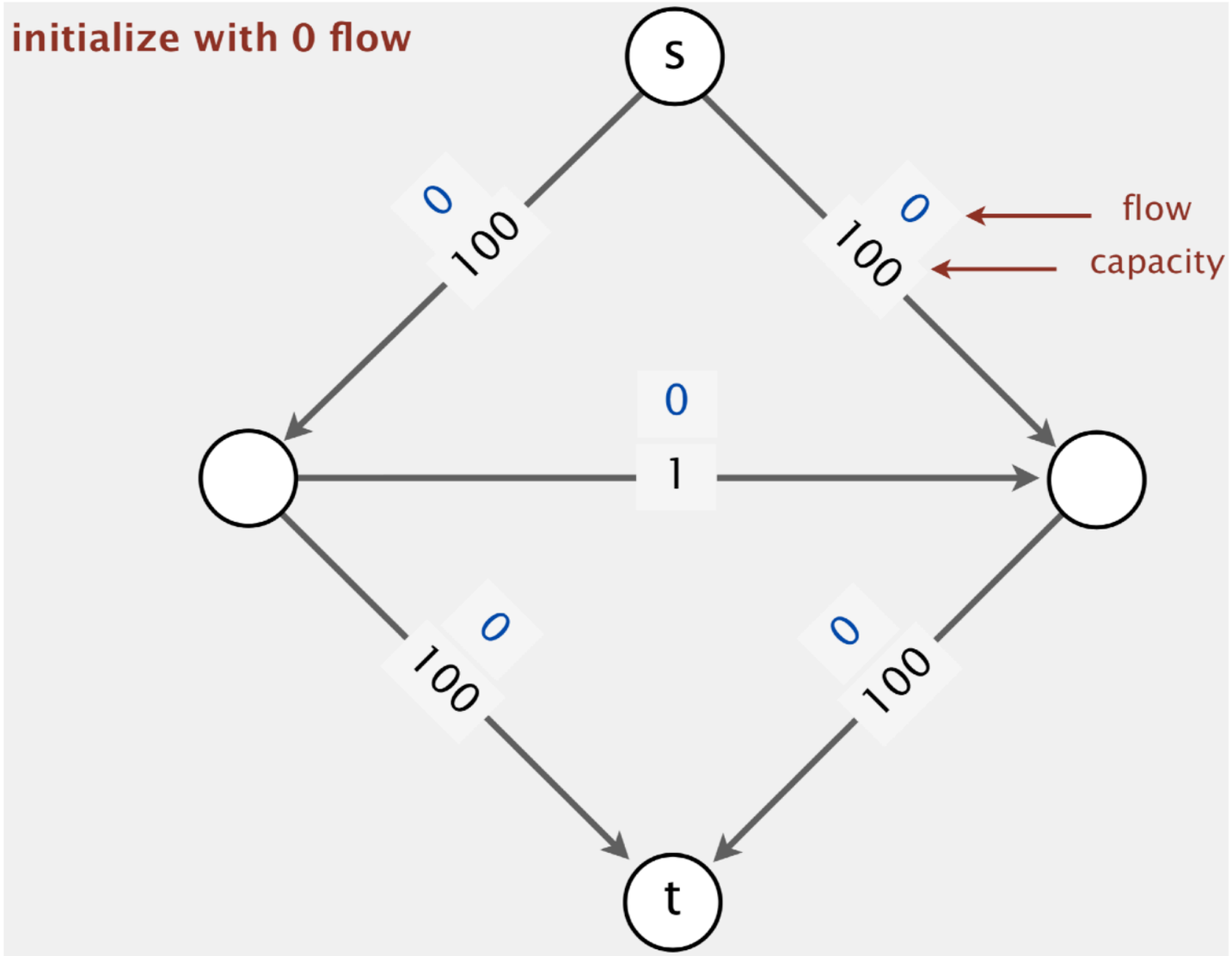
- Proposition. Number of augmentations ≤ the value of the maxflow.

- Pf. Each augmentation increases the value by at least 1.

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Worst Case

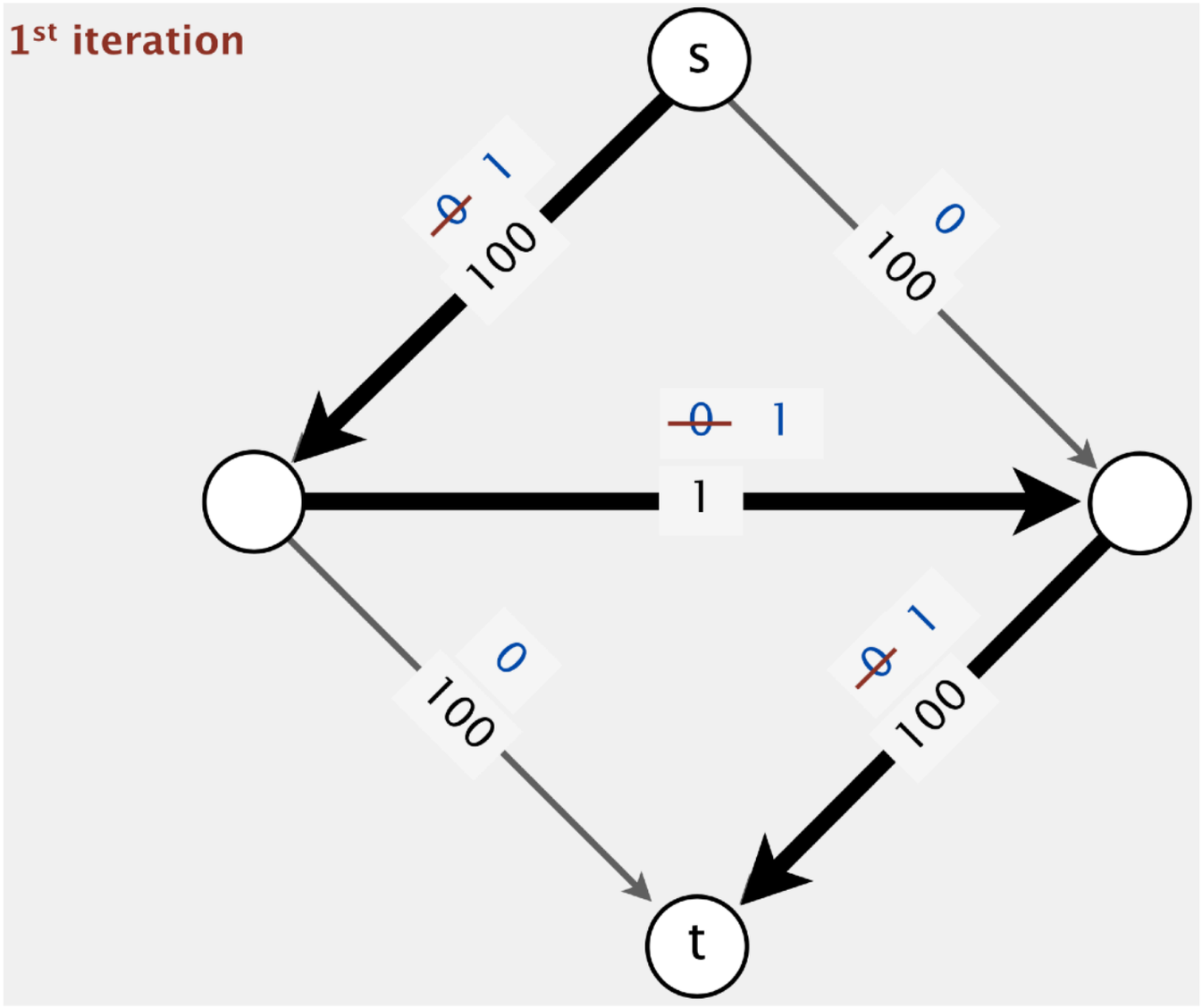
- Bad news. Even when edge capacities are integers, number of augmenting paths could be equal to the value of the maxflow.



CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Worst Case

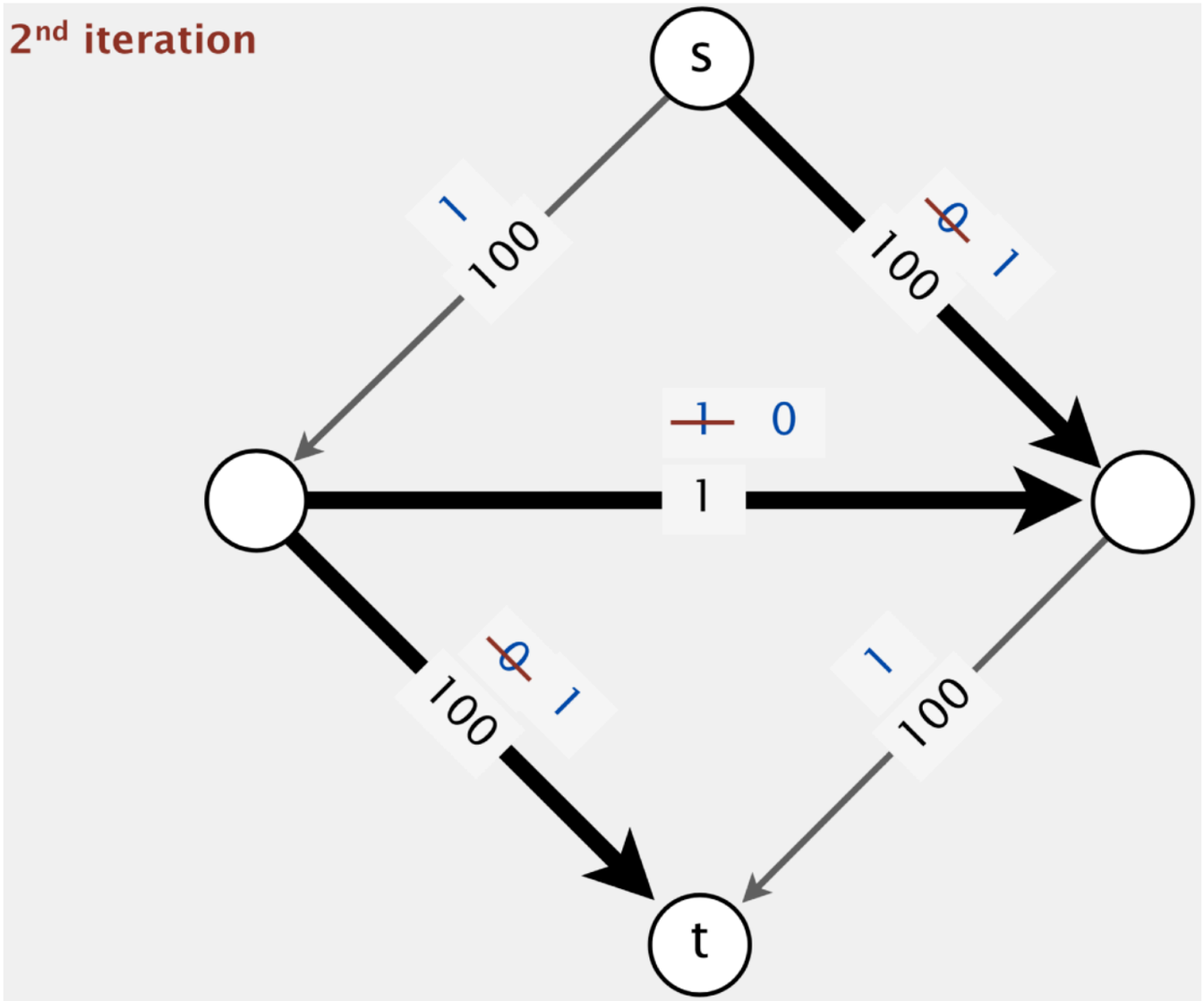
- Bad news. Even when edge capacities are integers, number of augmenting paths could be equal to the value of the maxflow.



CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Worst Case

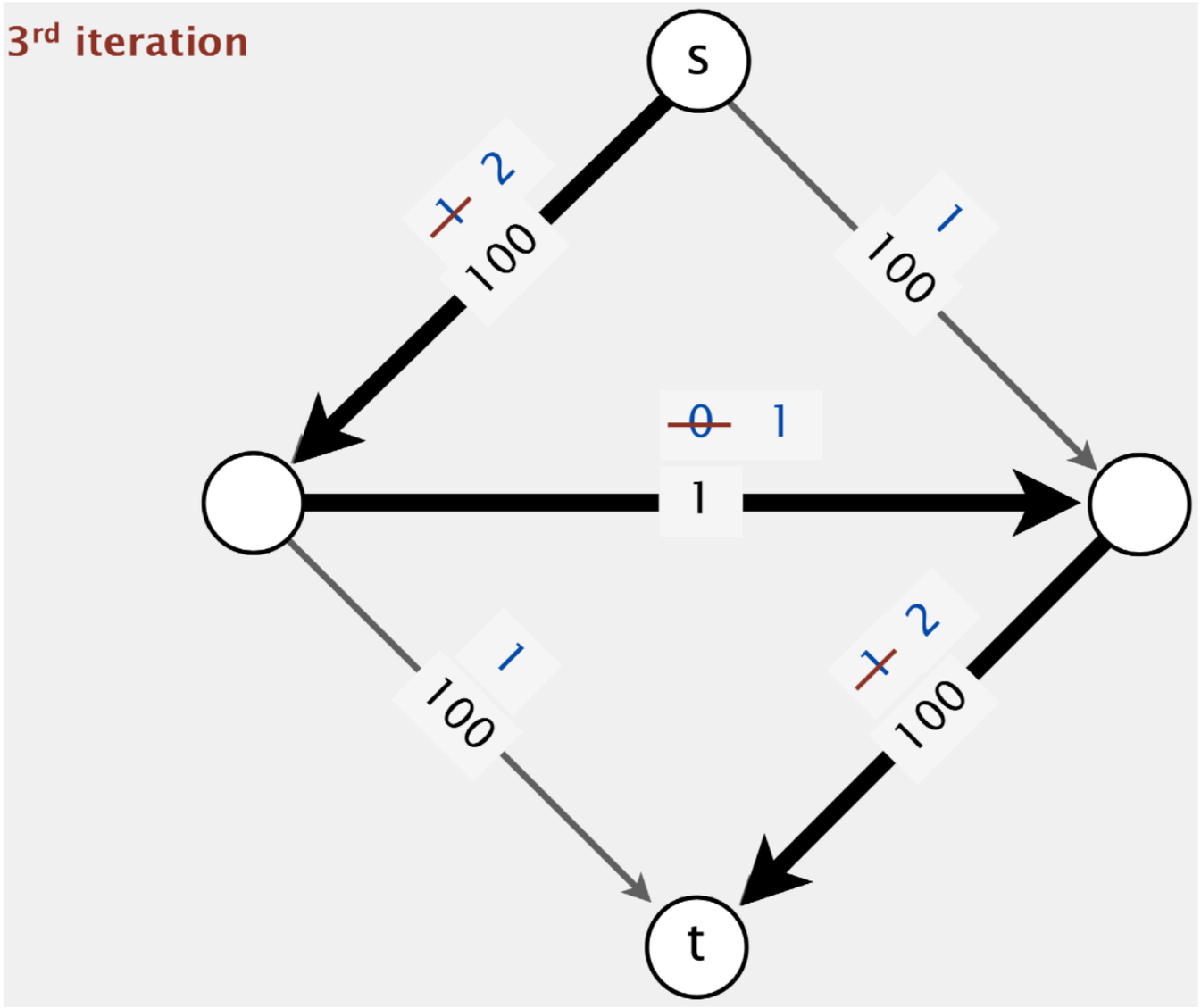
- Bad news. Even when edge capacities are integers, number of augmenting paths could be equal to the value of the maxflow.



CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Worst Case

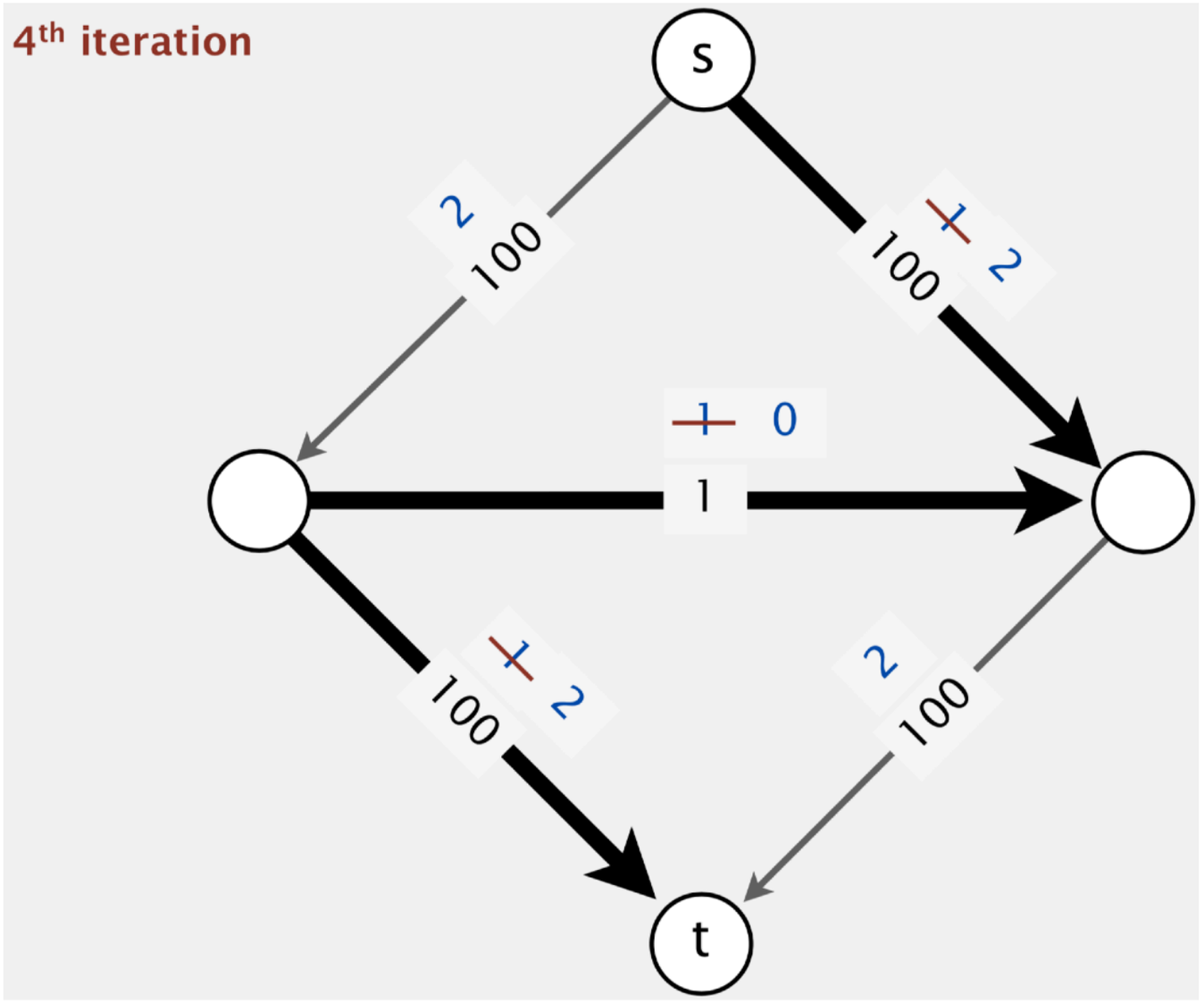
- Bad news. Even when edge capacities are integers, number of augmenting paths could be equal to the value of the maxflow.



CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Worst Case

- Bad news. Even when edge capacities are integers, number of augmenting paths could be equal to the value of the maxflow.



CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

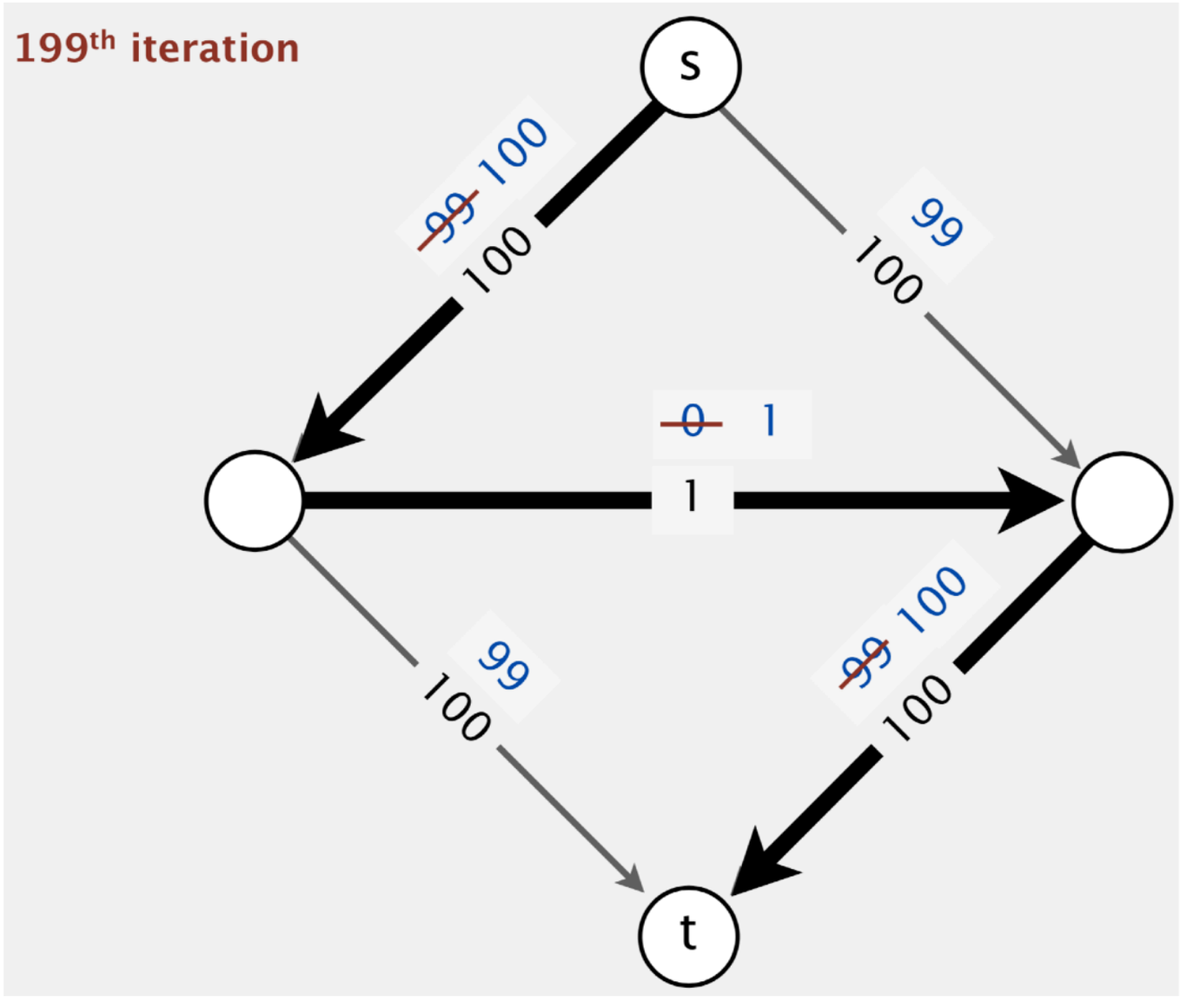
Worst Case

- Bad news. Even when edge capacities are integers, number of augmenting paths could be equal to the value of the maxflow.

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Worst Case

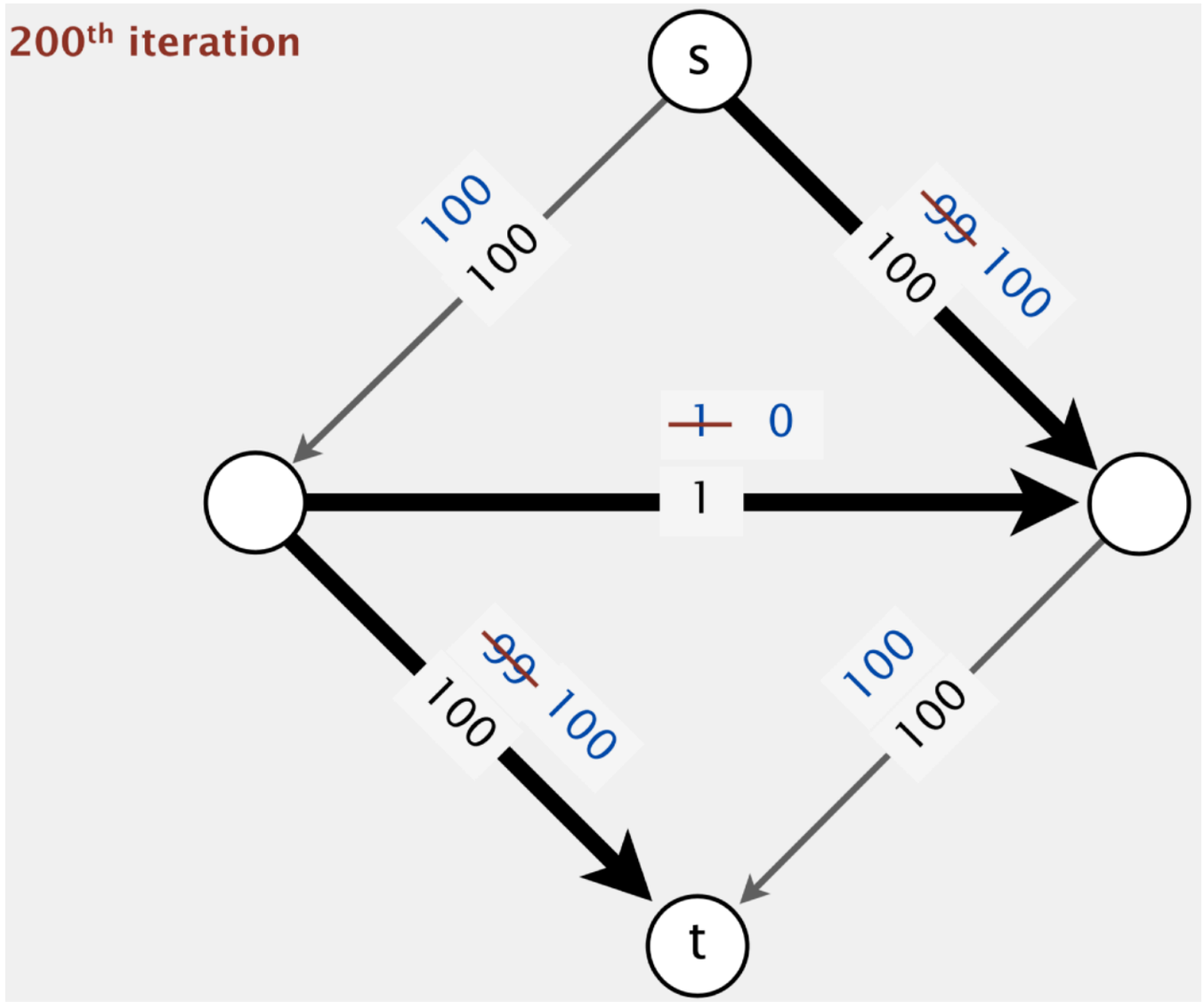
- Bad news. Even when edge capacities are integers, number of augmenting paths could be equal to the value of the maxflow.



CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Worst Case

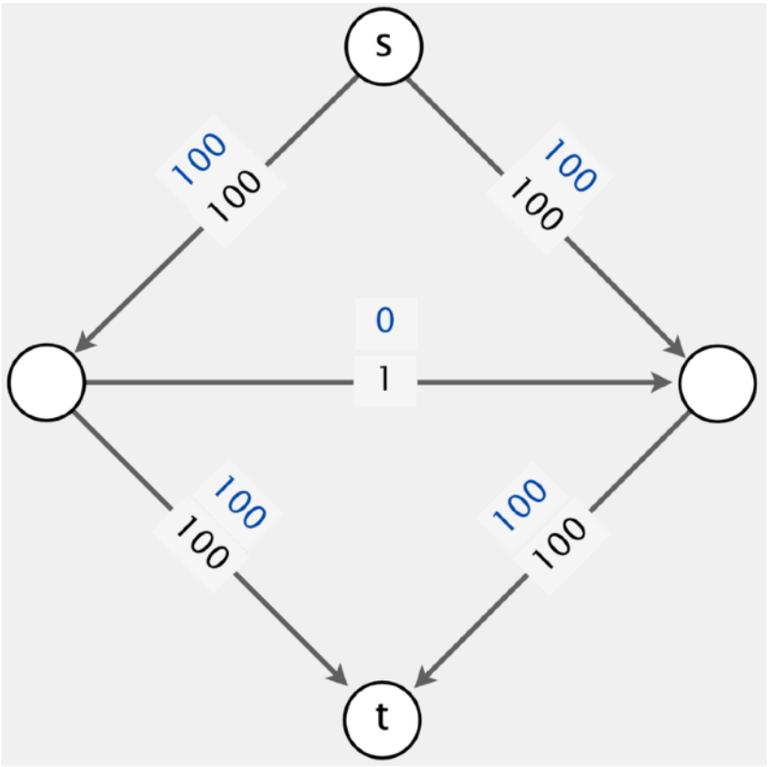
- Bad news. Even when edge capacities are integers, number of augmenting paths could be equal to the value of the maxflow.



CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Worst Case

- Bad news. Even when edge capacities are integers, number of augmenting paths could be equal to the value of the maxflow.
  - can be exponential in input size
- Good news. This case is easily avoided. [use shortest/ fattest path]



CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

How to choose augmenting paths?

- FF performance depends on choice of augmenting paths.

augmenting path	**number of paths**	**implementation**
shortest path	≤ 1/2E V	queue (BFS)
fattest path	≤ E ln(E U)	priority queue
random path	≤ EU	randomized queue
DFS path	≤ EU	stack (DFS)

digraph with V vertices, E edges, and integer capacities between 1 and U

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Maximum flow algorithms: theory

year	method	worst case	discovered by
1951	simplex	$E^3 U$	Dantzig
1955	augmenting path	$E^2 U$	Ford-Fulkerson
1970	shortest augmenting path	E^3	Dinitz, Edmonds-Karp
1970	fattest augmenting path	$E^2 \log E \log(E U)$	Dinitz, Edmonds-Karp
1977	blocking flow	$E^{5/2}$	Cherkasky
1978	blocking flow	$E^{7/3}$	Galil
1983	dynamic trees	$E^2 \log E$	Sleator-Tarjan
1985	capacity scaling	$E^2 \log U$	Gabow
1997	length function	$E^{3/2} \log E \log U$	Goldberg-Rao
2012	compact network	$E^2 / \log E$	Orlin
?	?	E	?

maxflow algorithms for sparse digraphs with E edges, integer capacities between 1 and U

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Outline

- The Single-Source Shortest Path Problem
- The Maximum Flow Problem
- The Maximum Cut Problem
- The Traveling Salesman Problem

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

The Maximum Cut Problem

- Input: An undirected graph G = (V , E).
- Goal: A cut (A,B) – a partition of V into two non-empty sets – that maximizes the number of crossing edges.
- Sad fact: NP-complete.
- Computationally tractable special case: Bipartite graphs (i.e., where there is a cut such that all edges are crossing)
  - Solve in linear time via breadth-first search

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Exercise

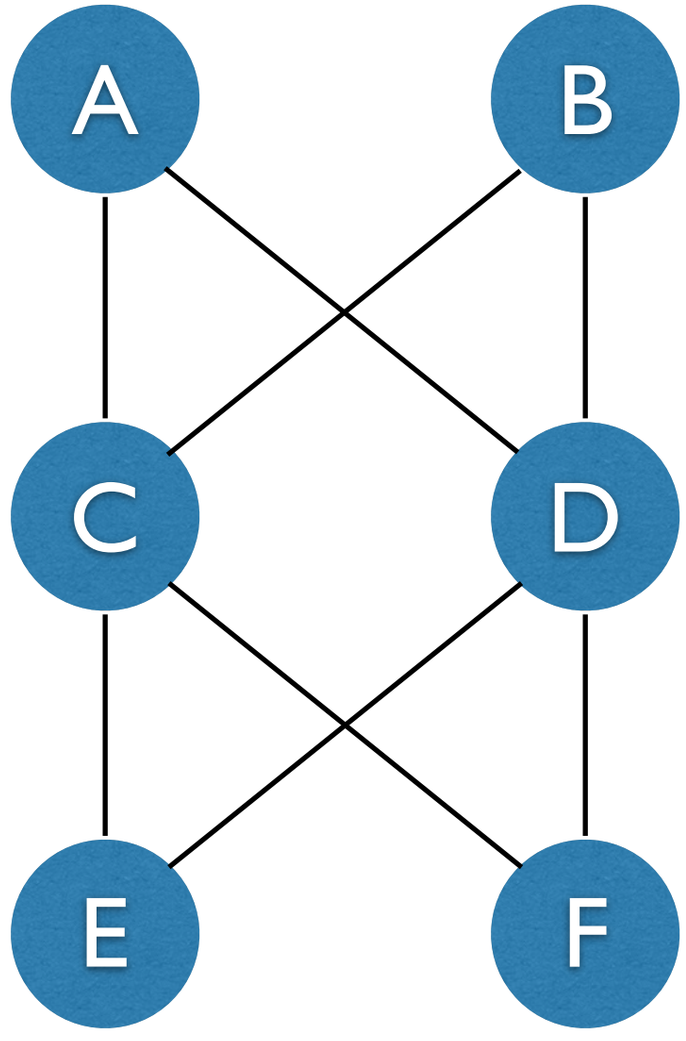
- Question: What is the value of a maximum cut in the following graph?

A. 4

B. 6

C. 8

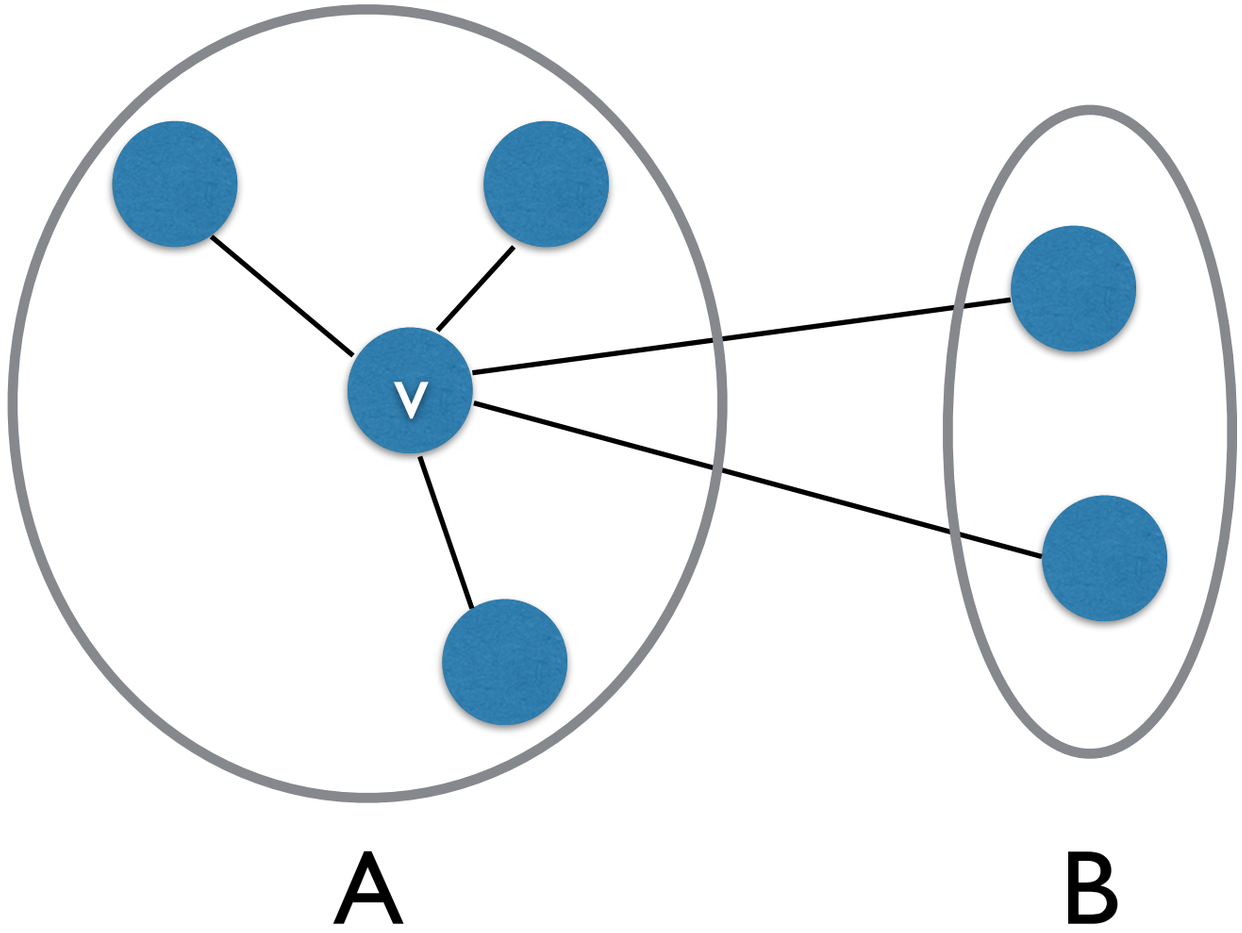
D. 10



CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

A Local Search Algorithm

- Notation: For a cut (A,B) and a vertex v, define
  $c_v(A, B)$ = # of edges incident on v that cross (A, B)
  $d_v(A, B)$ = # of edges incident on v that don't cross (A, B)



$c_v(A, B)=2$

$d_v(A, B)=3$

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

A Local Search Algorithm

1. Let (A,B) be an arbitrary cut of G
2. While there is a vertex v with $d_v(A, B) > c_v(A, B)$:
   - Move v to other side of the cut
   - increases number of crossing edges by $d_v(A, B) - c_v(A, B) > 0$
3. Return final cut (A,B)

- Note: Terminates within $\binom{n}{2}$ iterations, hence in polynomial time.

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

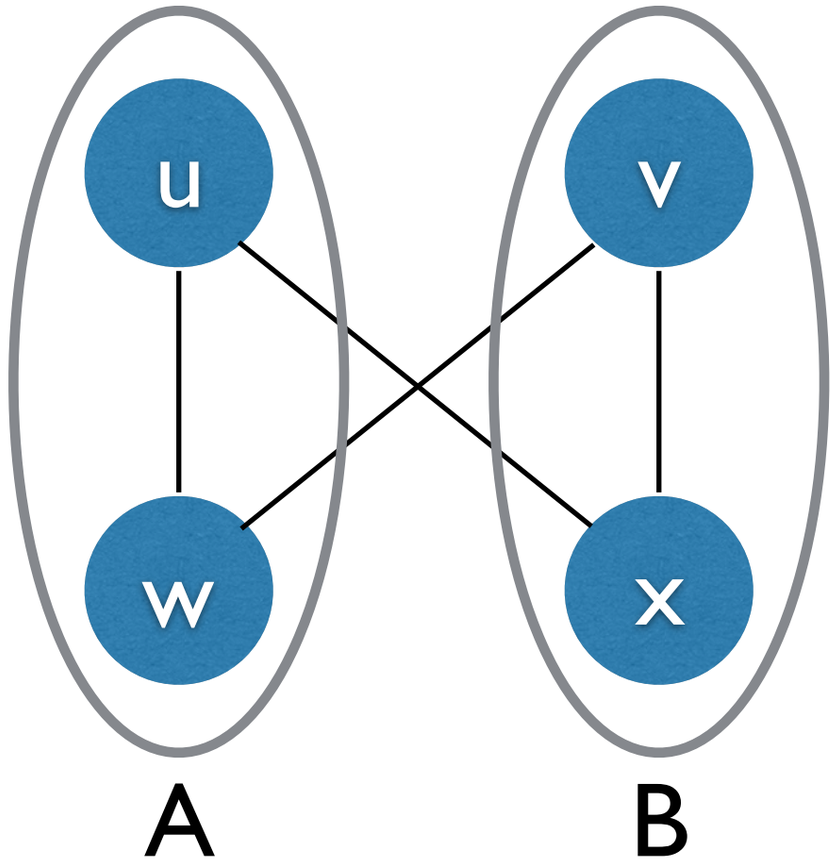
Performance Guarantees

- Theorem: This local search algorithm always outputs a cut in which the number of crossing edges is at least 50% of the maximum possible. (Even 50% of |E|)

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Performance Guarantees

- A tight example

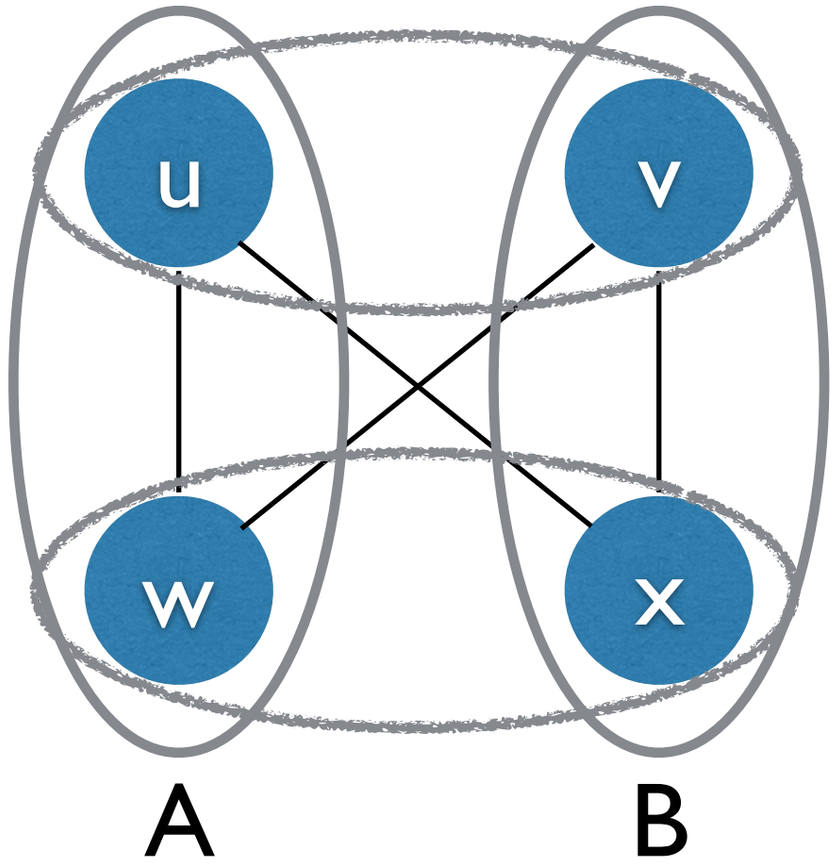


2 crossing edges

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Performance Guarantees

- A tight example



4 crossing edges

2 crossing edges

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Performance Guarantees

- Cautionary point: Expected number of crossing edges of a random cut already is 1/2|E|.

Proof: Consider a random cut (A,B). For edge e $\in$ E, define

- X_e=1, if e crosses (A,B),
- X_e=0, otherwise.
- We have E[X_e]=Pr[X_e = 1]=1/2.
- So E[# crossing edges]=E[$\sum_e X_e$]=$\sum$E[X_e]=|E|/2. QED

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Proof of Performance Guarantee

- Theorem: This local search algorithm always outputs a cut in which the number of crossing edges is at least 50% of the maximum possible. (Even 50% of |E|)

Proof. Let (A,B) be a locally optimal cut. Then, for every vertex v, $d_v(A,B) \leq c_v(A,B)$. Summing over all $v \in V$:

$$\sum_{v \in V} d_v(A,B) \leq \sum_{v \in V} c_v(A,B)$$

counts each non-crossing edge twice

counts each crossing edge twice

So: $2 \cdot$ [# of non-crossing edges] $\leq 2 \cdot$ [# of crossing edges]

$2 \cdot |E| \leq 4 \cdot$ [# of crossing edges]

# of crossing edges $\geq$ 1/2|E| QED!

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Weighted Maximum Cut Problem

- Generalization: Each edge e ∈ E has a nonnegative weight w_e, want to maximize total weight of crossing edges.
- Notes:

1. Local search still well defined
2. Performance guarantee of 50% still holds for locally optimal cuts (also for a random cut)
3. No longer guaranteed to converge in polynomial time [non-trivial exercise]

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

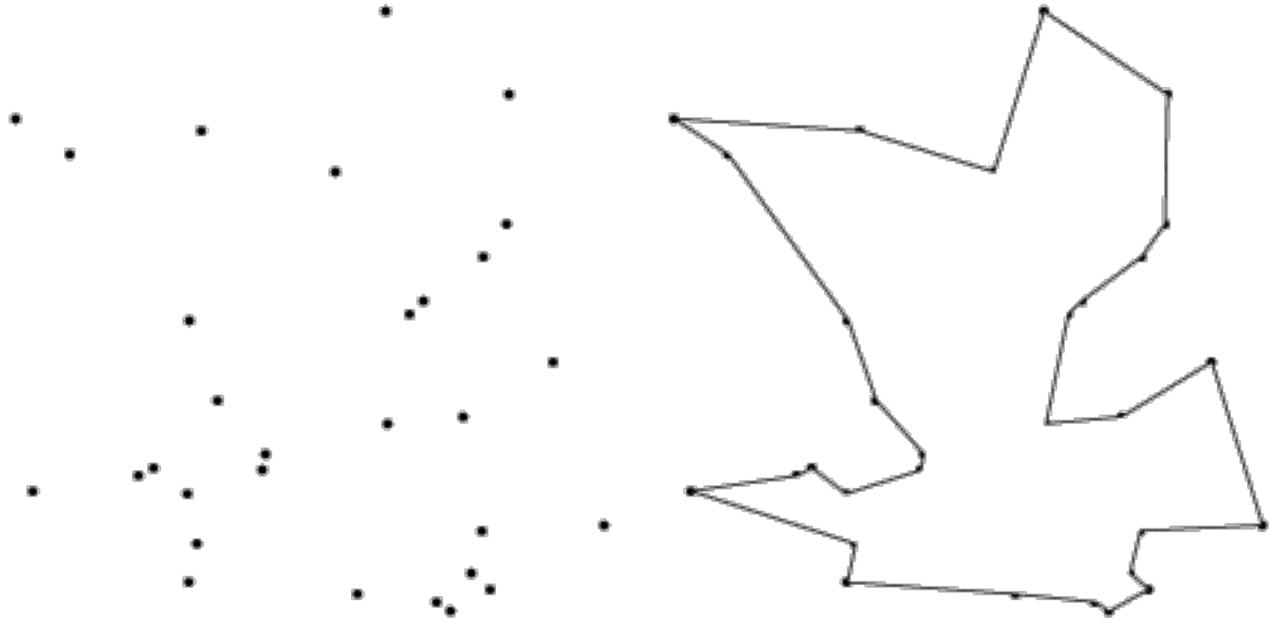
Outline

- The Single-Source Shortest Path Problem
- The Maximum Flow Problem
- The Maximum Cut Problem
- The Traveling Salesman Problem

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Problem Definition

- **Travelling Salesman Problem (TSP):** Given a set of cities and distance between each pair of cities, the problem is to find the shortest possible route that visits every city exactly once and returns to the origin city.



CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Applications of the TSP

- **Routing around Cities**
- **Computer Wiring**
  - connecting together computer components using minimum wire length
- **Archaeological Seriation**
  - ordering sites in time
- **Genome Sequencing**
  - arranging DNA fragments in sequence
- **Job Sequencing**
  - sequencing jobs in order to minimize total set-up time between jobs
- **Wallpapering to Minimize Waste**

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

How to solve?

- The problem is a famous NP hard problem. There is no polynomial time known solution for this problem.
- Solutions

1. Try every possibility

- O(n!) – grows faster than exponentially
- If it took 1 microsecond to calculate each possibility takes 10^{140} centuries to calculate all possibilities when n = 100

2. Optimizing Methods

- Obtain **guaranteed** optimal solution, but can take a very very long time

3. Heuristic Methods

- obtain 'good' solutions 'quickly' by intuitive methods.

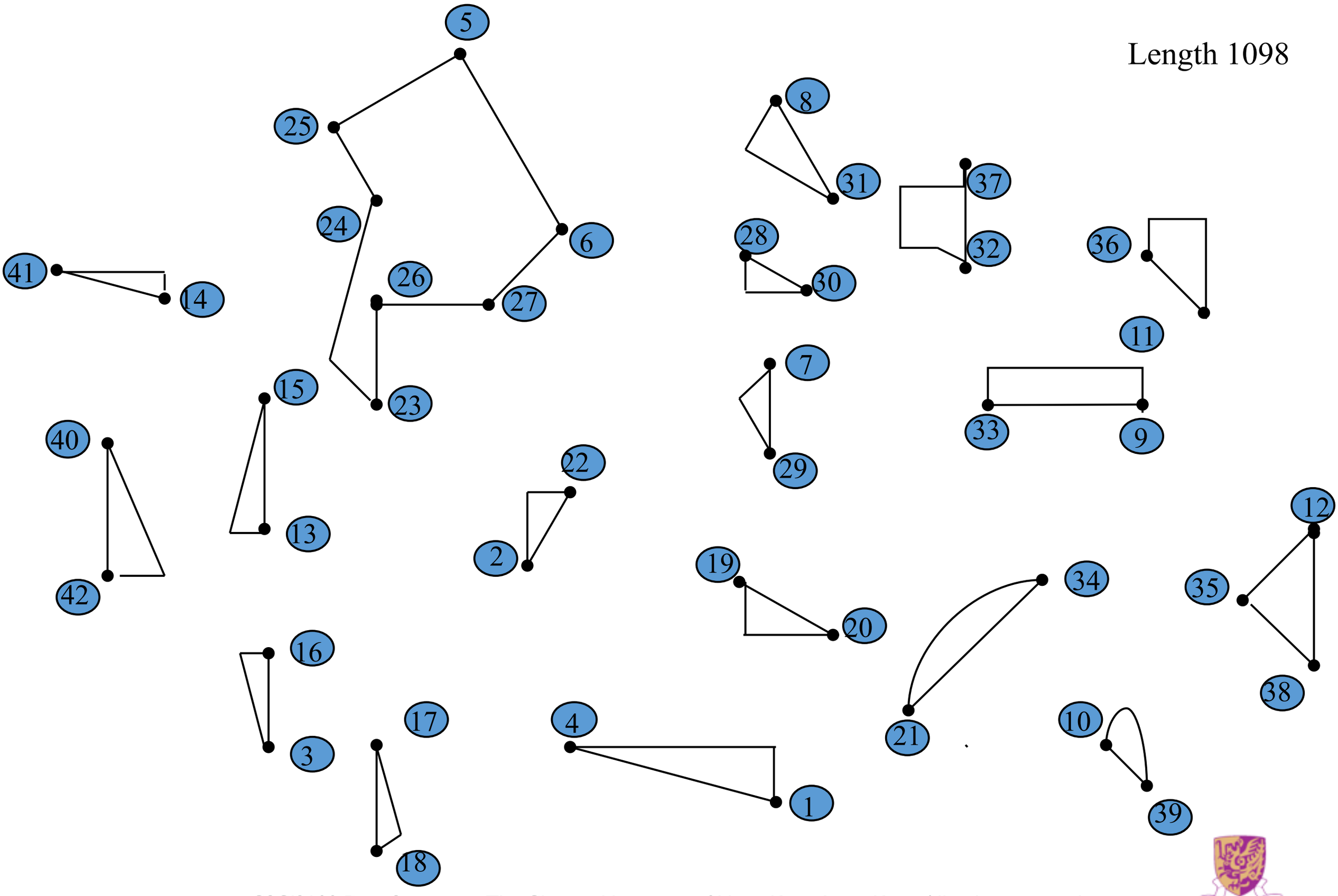
CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Optimizing Methods

1. Make sure every city visited once and left once – in cheapest way (Easy)
    - The Assignment Problem
    - Results in subtours
2. Put in extra constraints to remove subtours (More Difficult)
    - Results in new subtours
3. Remove new subtours and results in further subtours
4. Further subtours ……
5. Optimal Solution

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

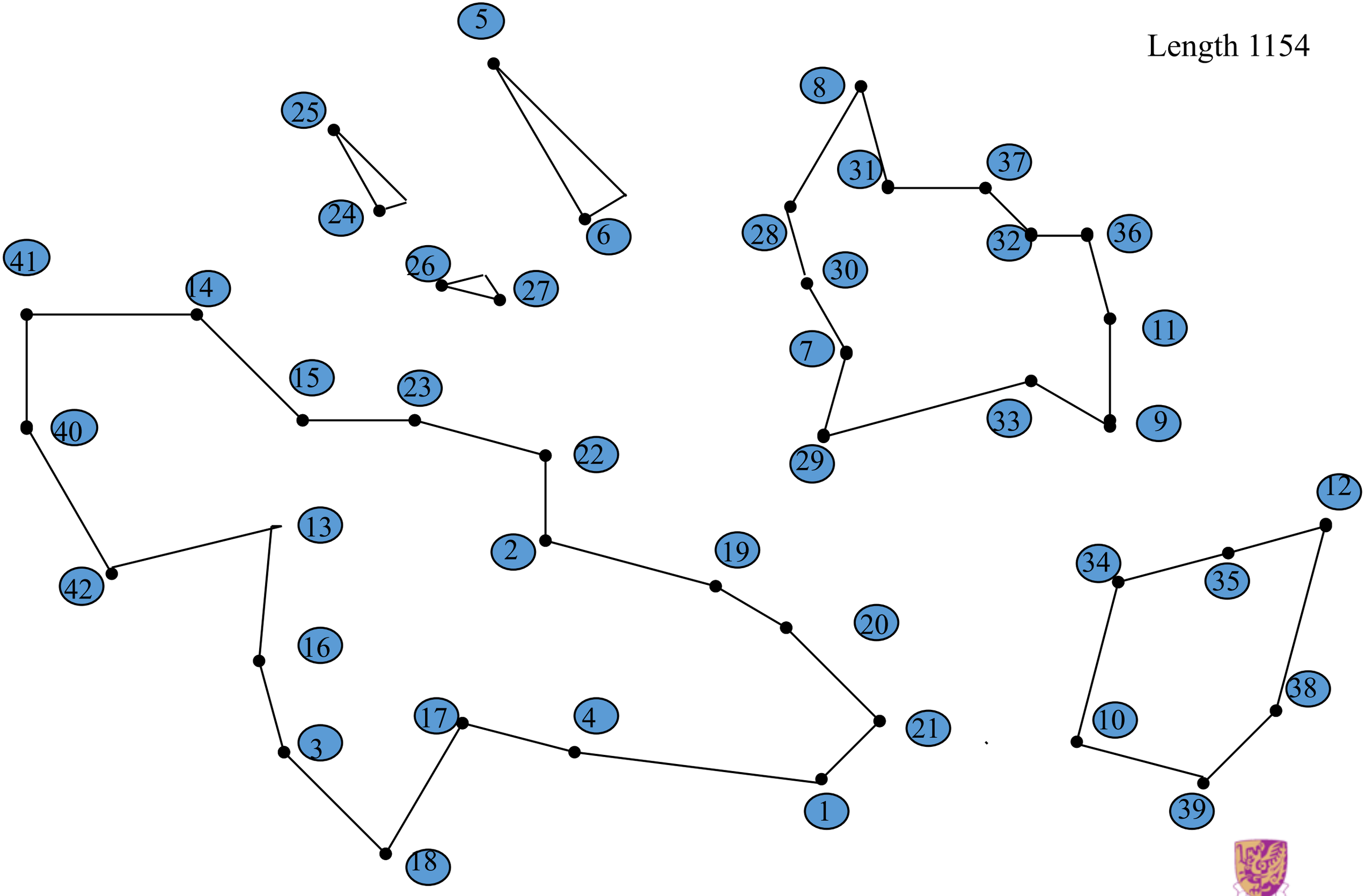
1. Make sure every city visited once and left once – in cheapest way (Easy)



CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

2. Put in extra constraints to remove subtours

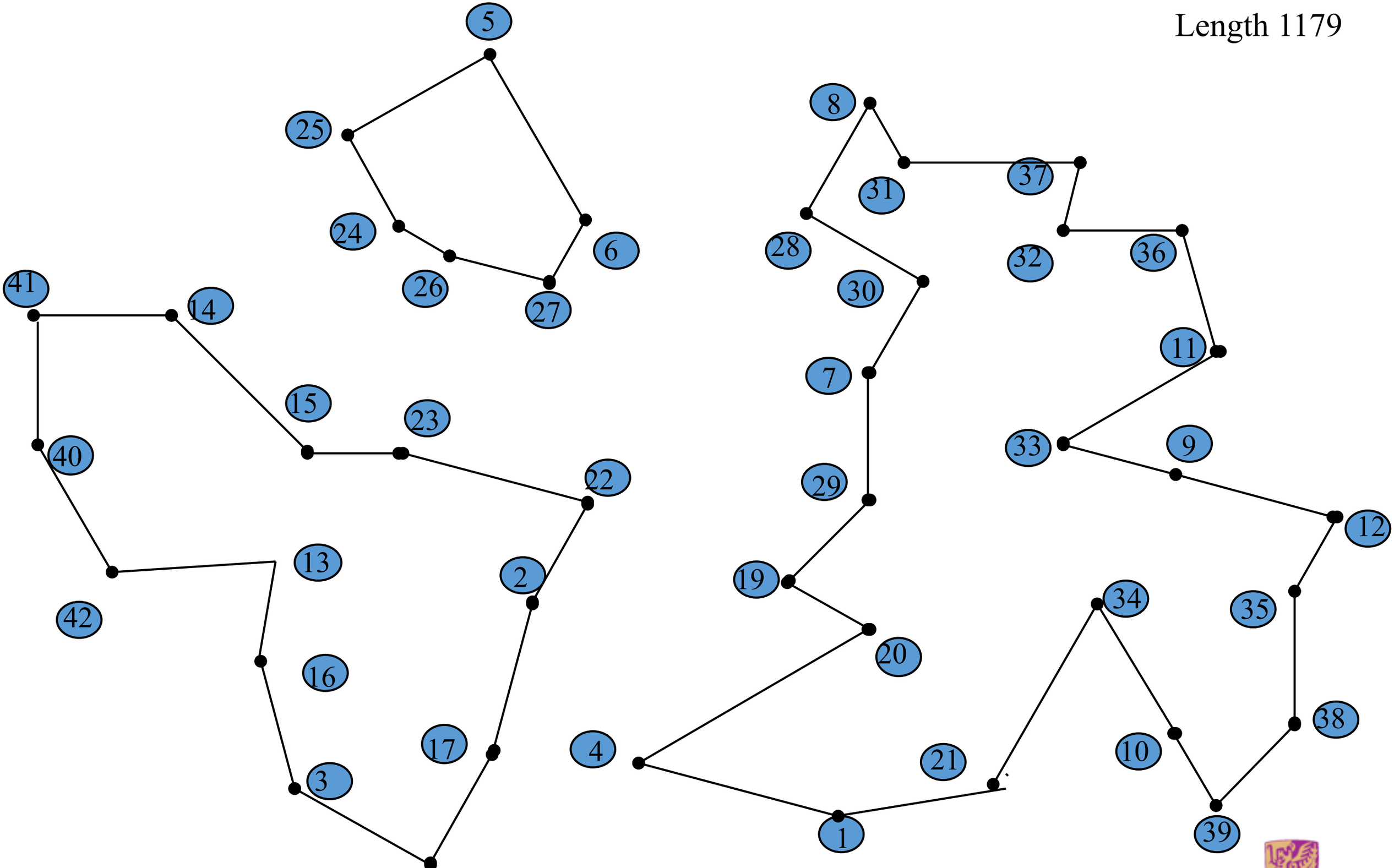
Length 1154



CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

3. Remove new subtours and results in further subtours

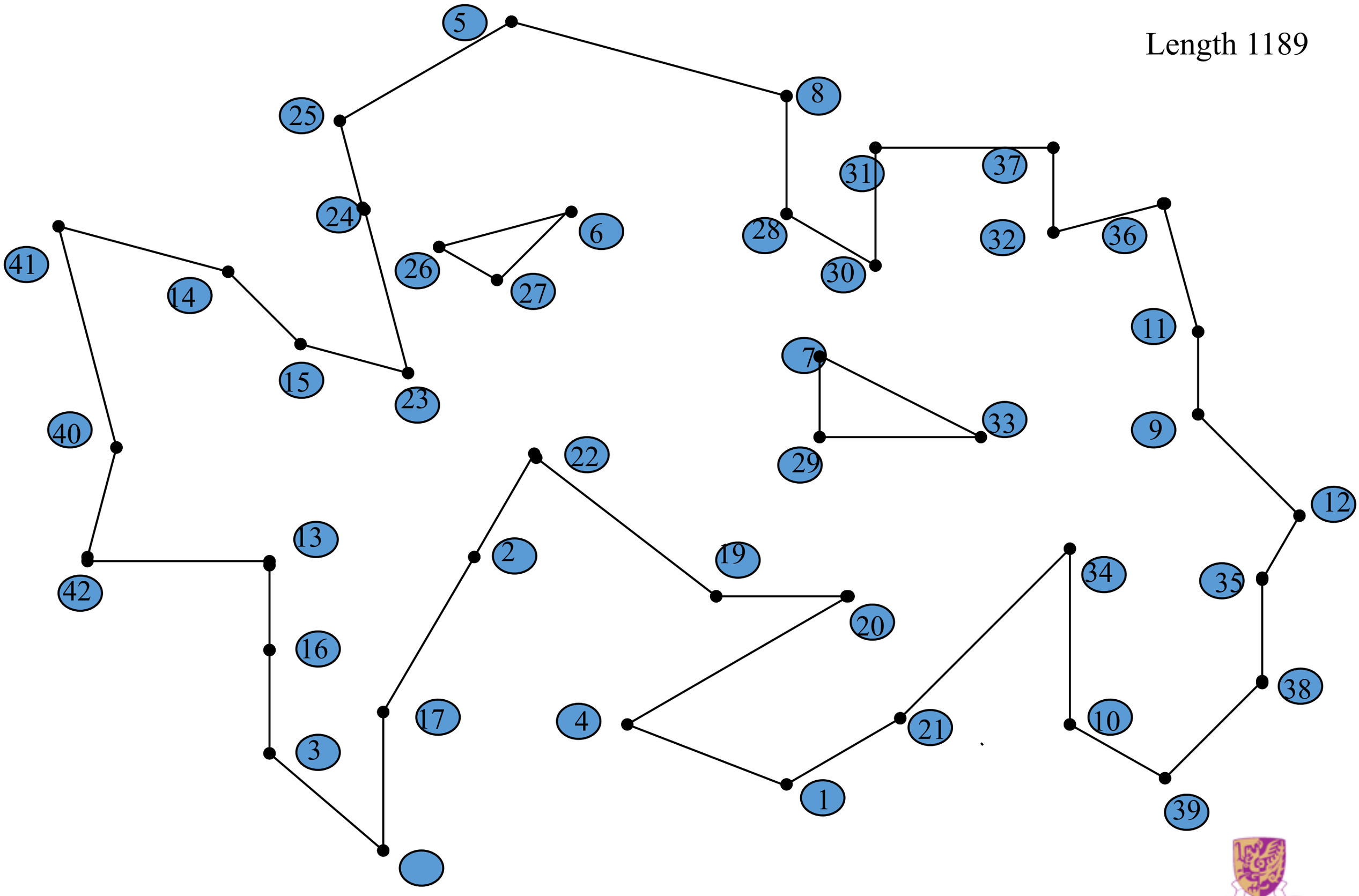
Length 1179



CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

4. Further subtours

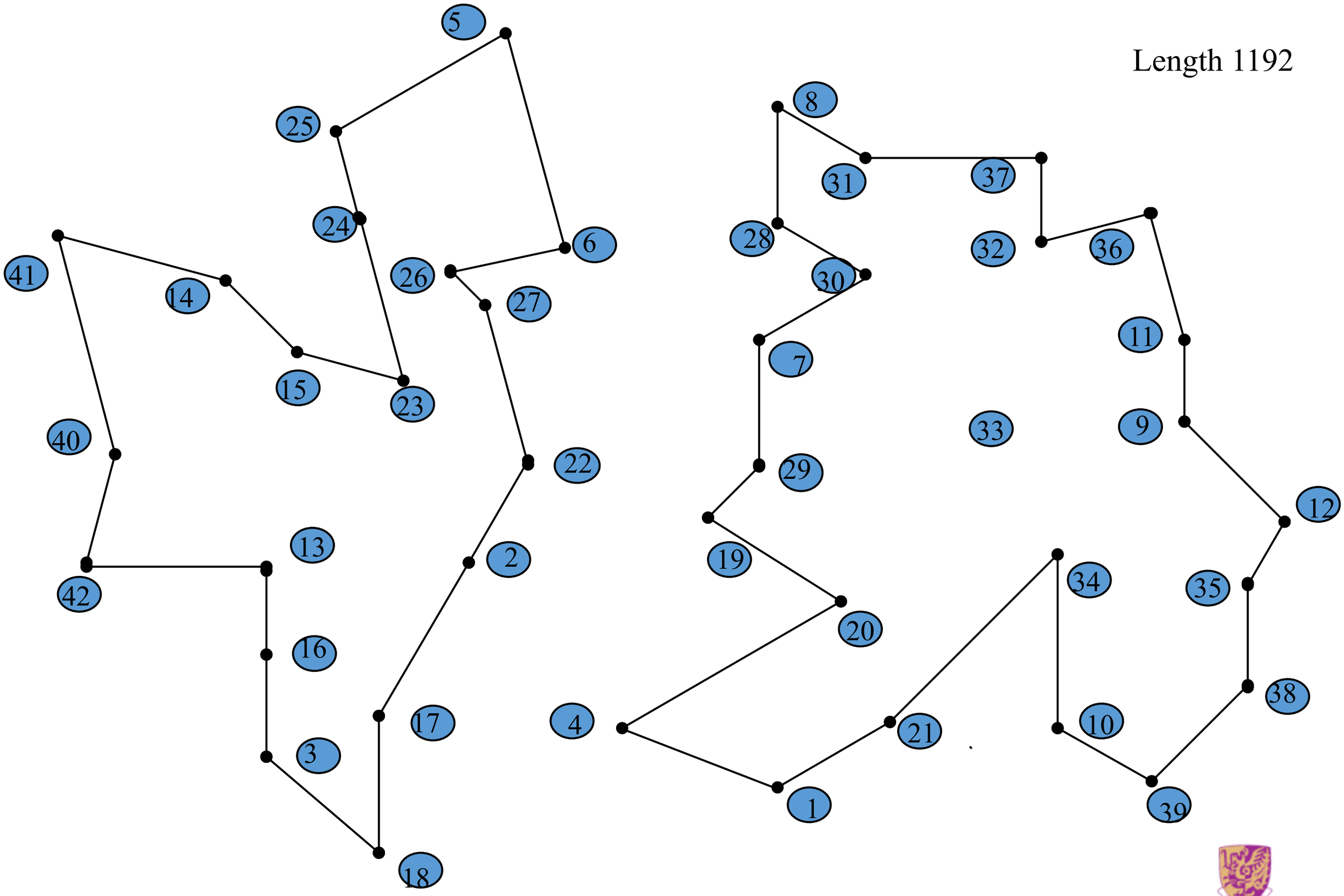
Length 1189



CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

4. Further subtours

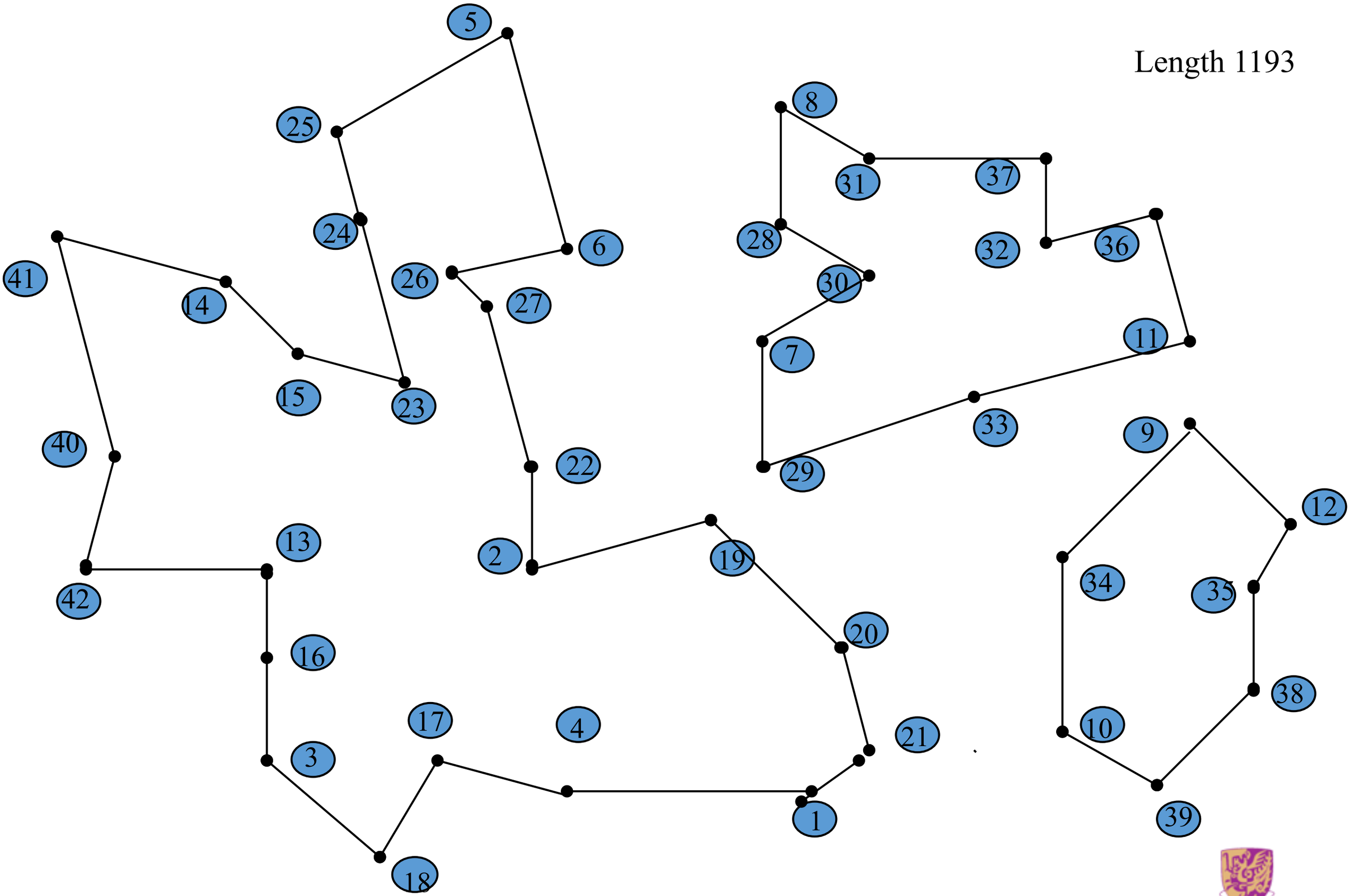
Length 1192



CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

4. Further subtours

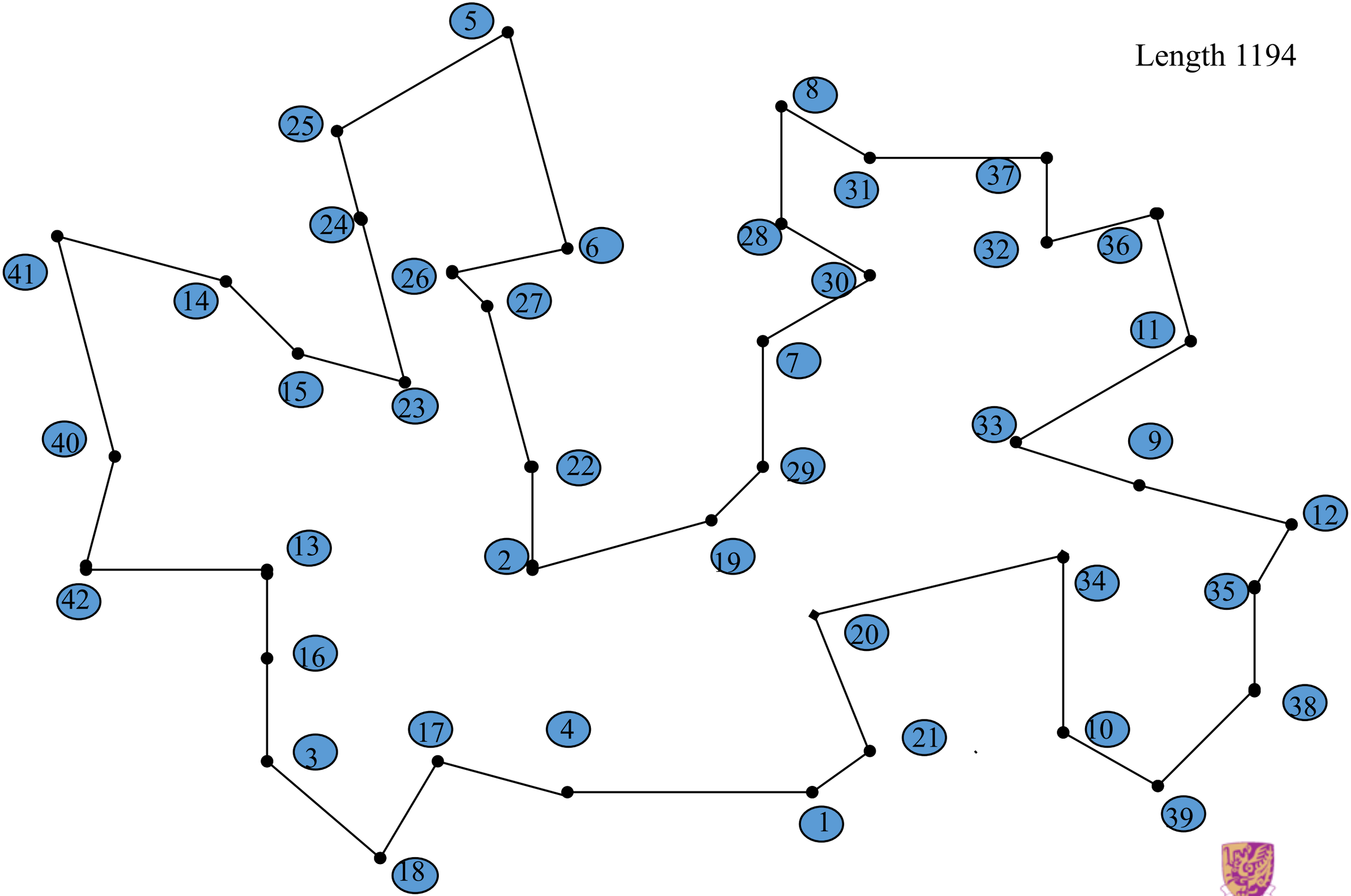
Length 1193



CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

5. Optimal Solution

Length 1194



CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

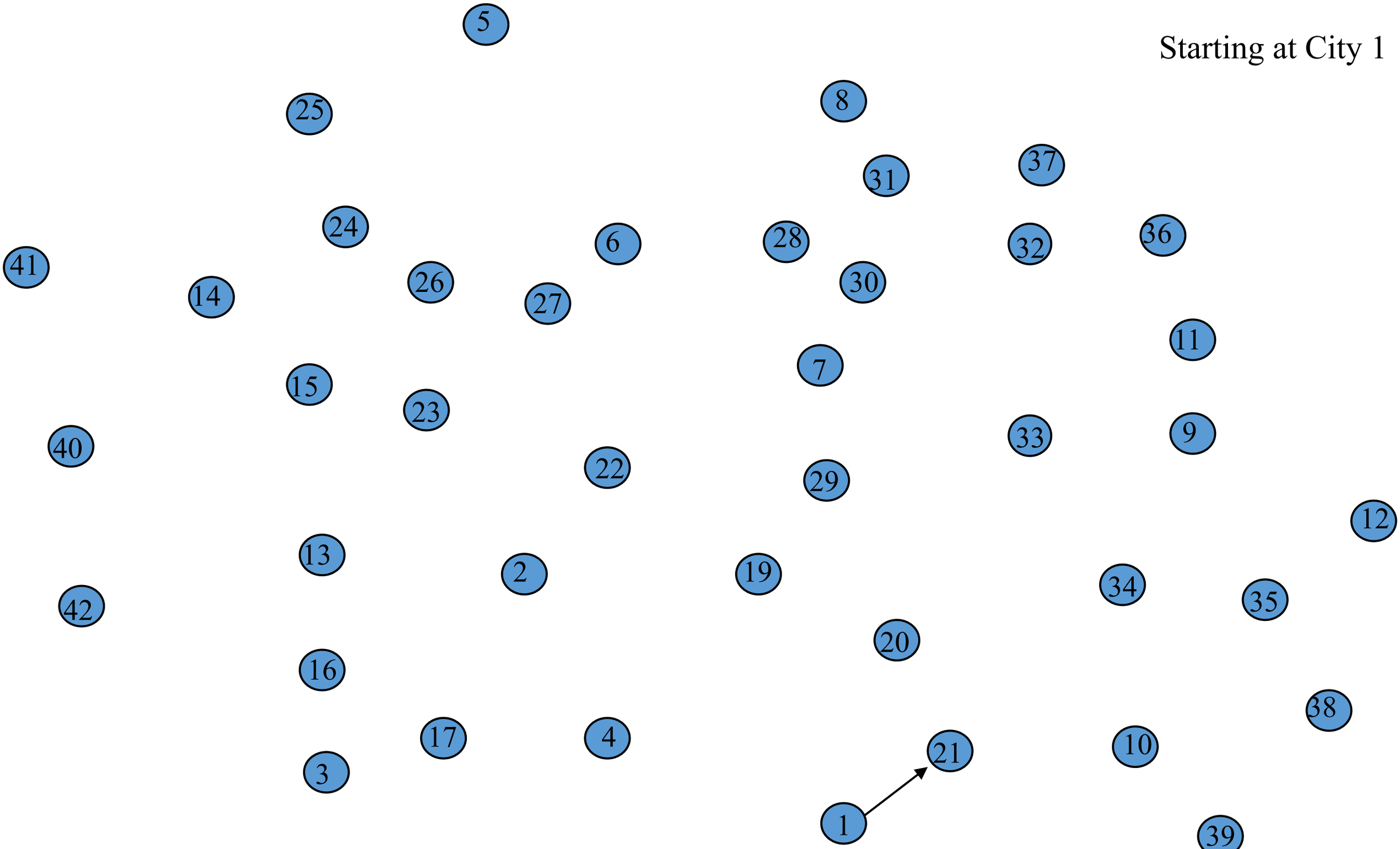
The Nearest Neighbour Method

- A Heuristic Method
- A 'Greedy' Method
  - Start Anywhere
  - Go to Nearest Unvisited City
  - Continue until all Cities visited
  - Return to Beginning

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

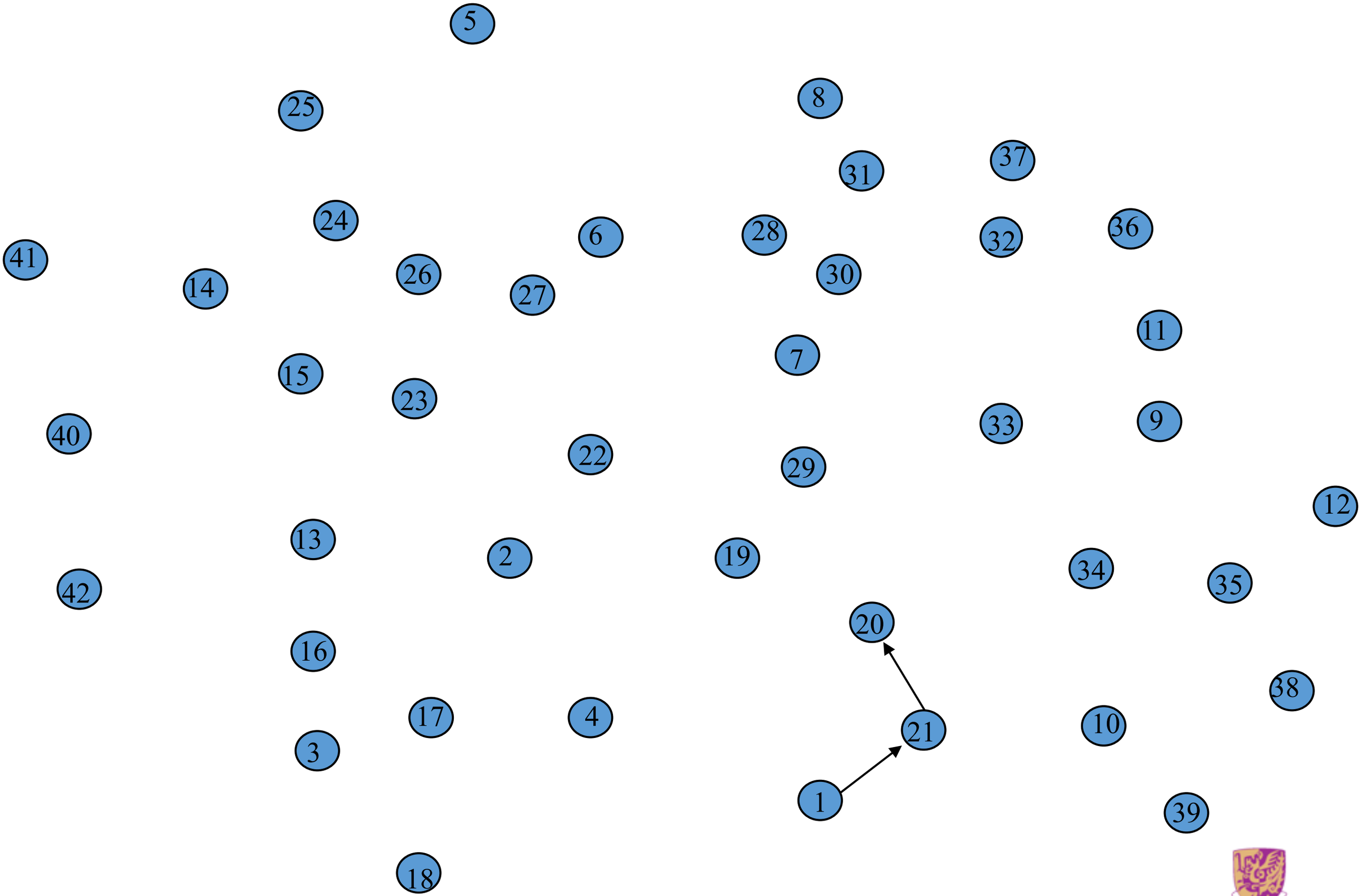
A 42-City Problem — The Nearest Neighbour Method

Starting at City 1



CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

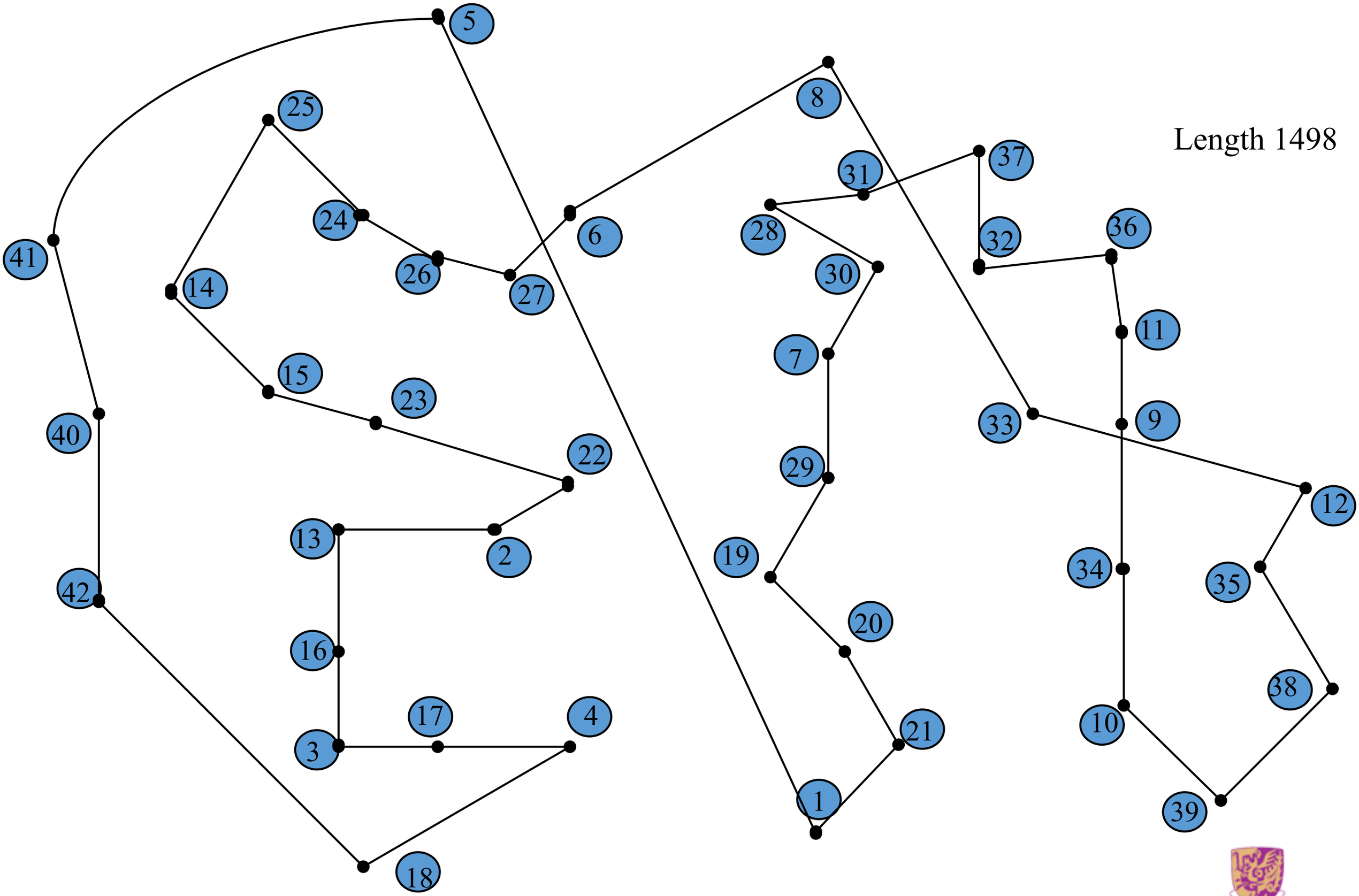
A 42-City Problem — The Nearest Neighbour Method



CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

A 42-City Problem — The Nearest Neighbour Method

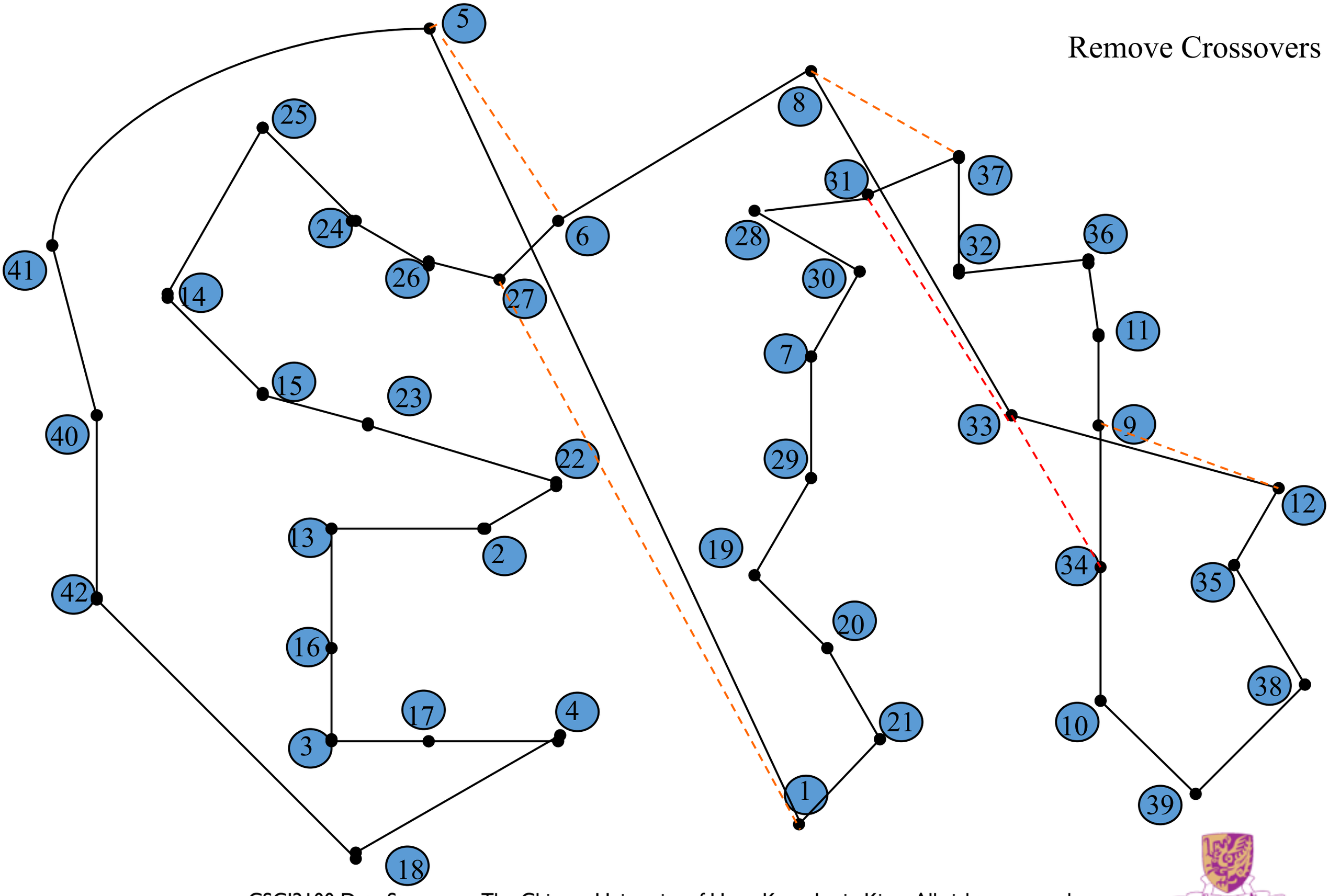
Length 1498



CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

A 42-City Problem — The Nearest Neighbour Method

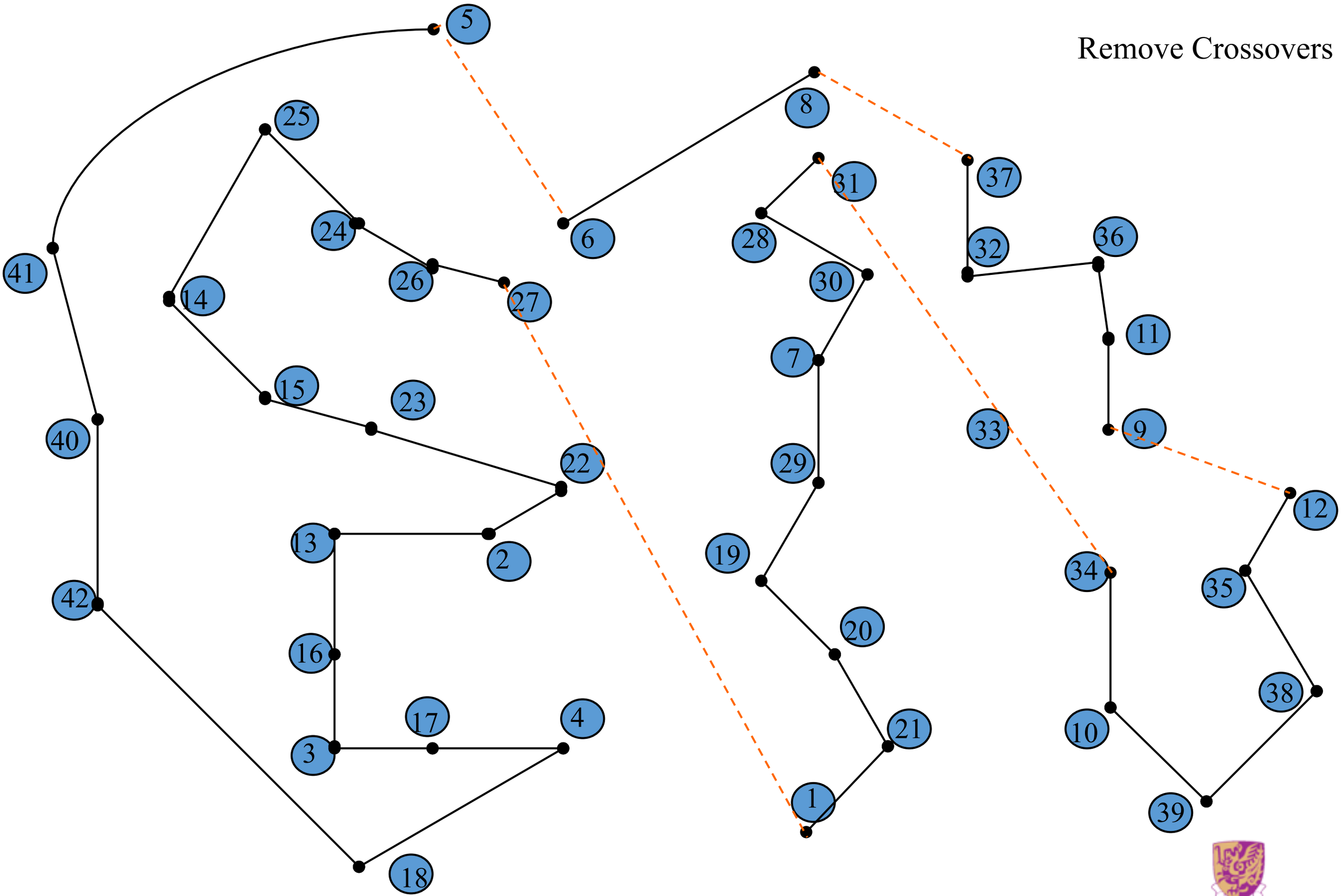
Remove Crossovers



CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

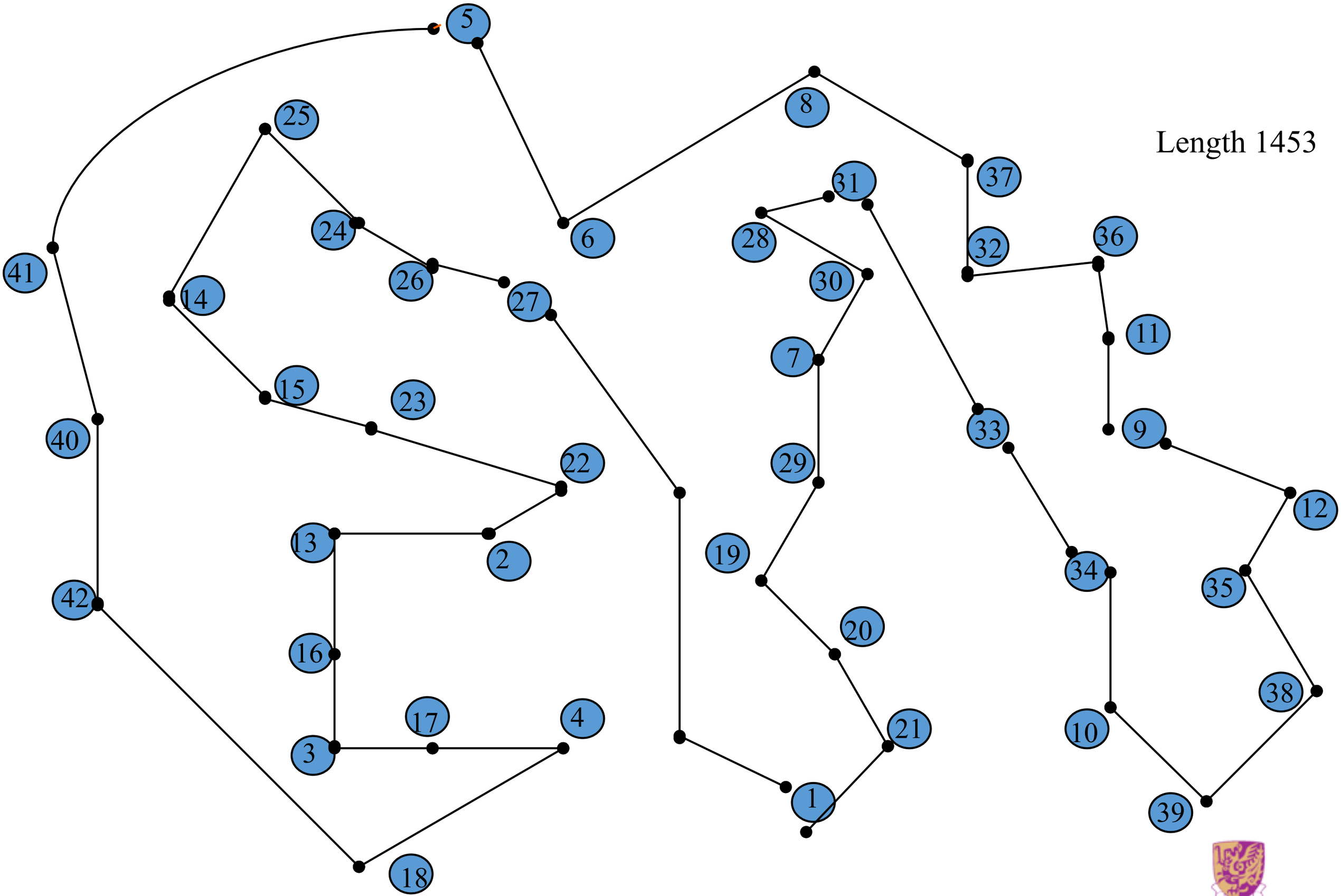
A 42-City Problem — The Nearest Neighbour Method

Remove Crossovers



CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

A 42-City Problem — The Nearest Neighbour Method



CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Christofides Method

- A heuristic method
- A Greedy Algorithm

1. Create Minimum Cost Spanning Tree
2. 'Match' Odd Degree Nodes
3. Create an Eulerian Tour

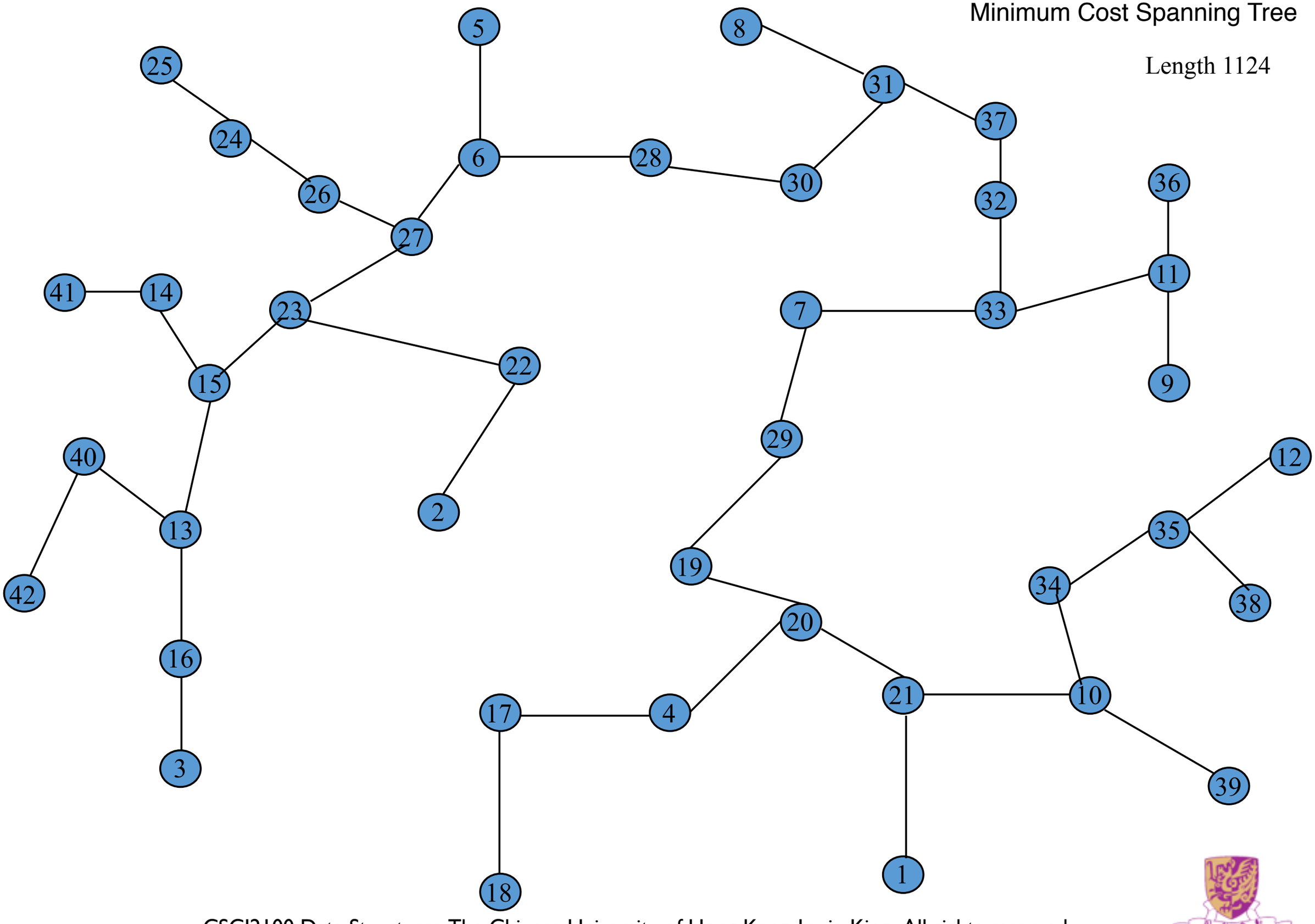
    - Short circuit cities revisited

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

A 42-City Problem — Christofides Method

Minimum Cost Spanning Tree

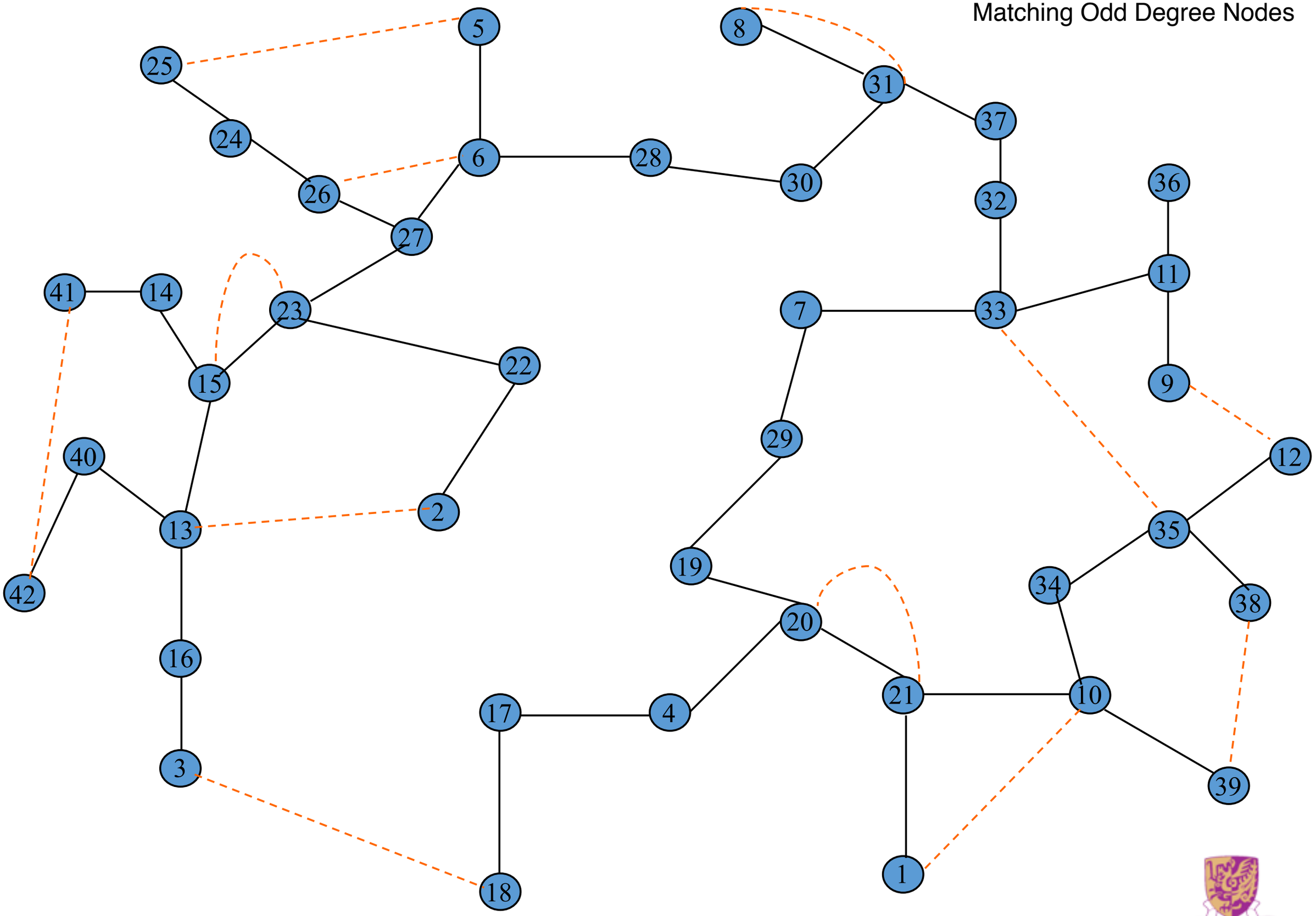
Length 1124



CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

A 42-City Problem — Christofides Method

Matching Odd Degree Nodes

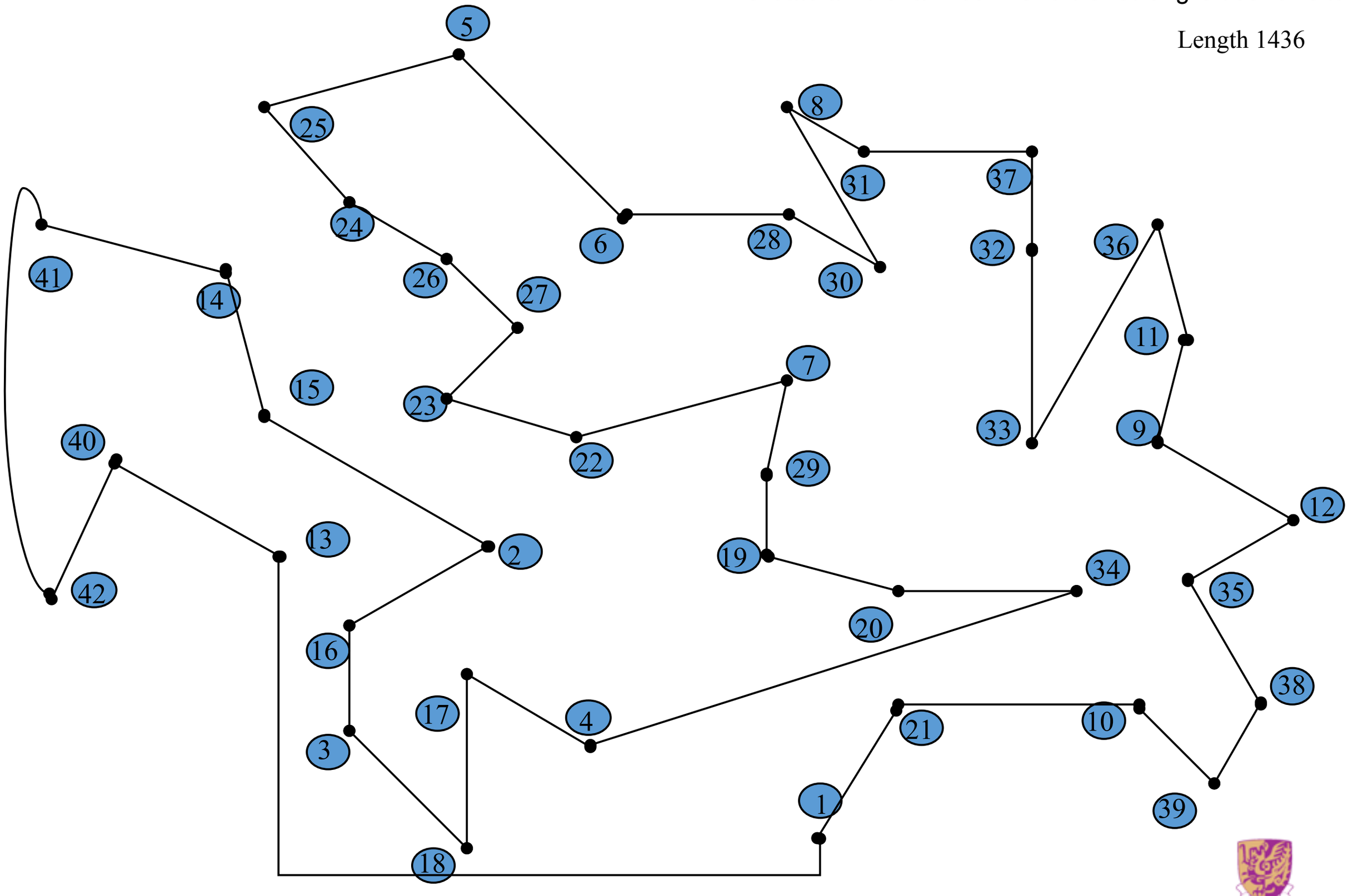


CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

A 42-City Problem — Christofides Method

Create a Eulerian Tour - Short Circuiting Cities revisited

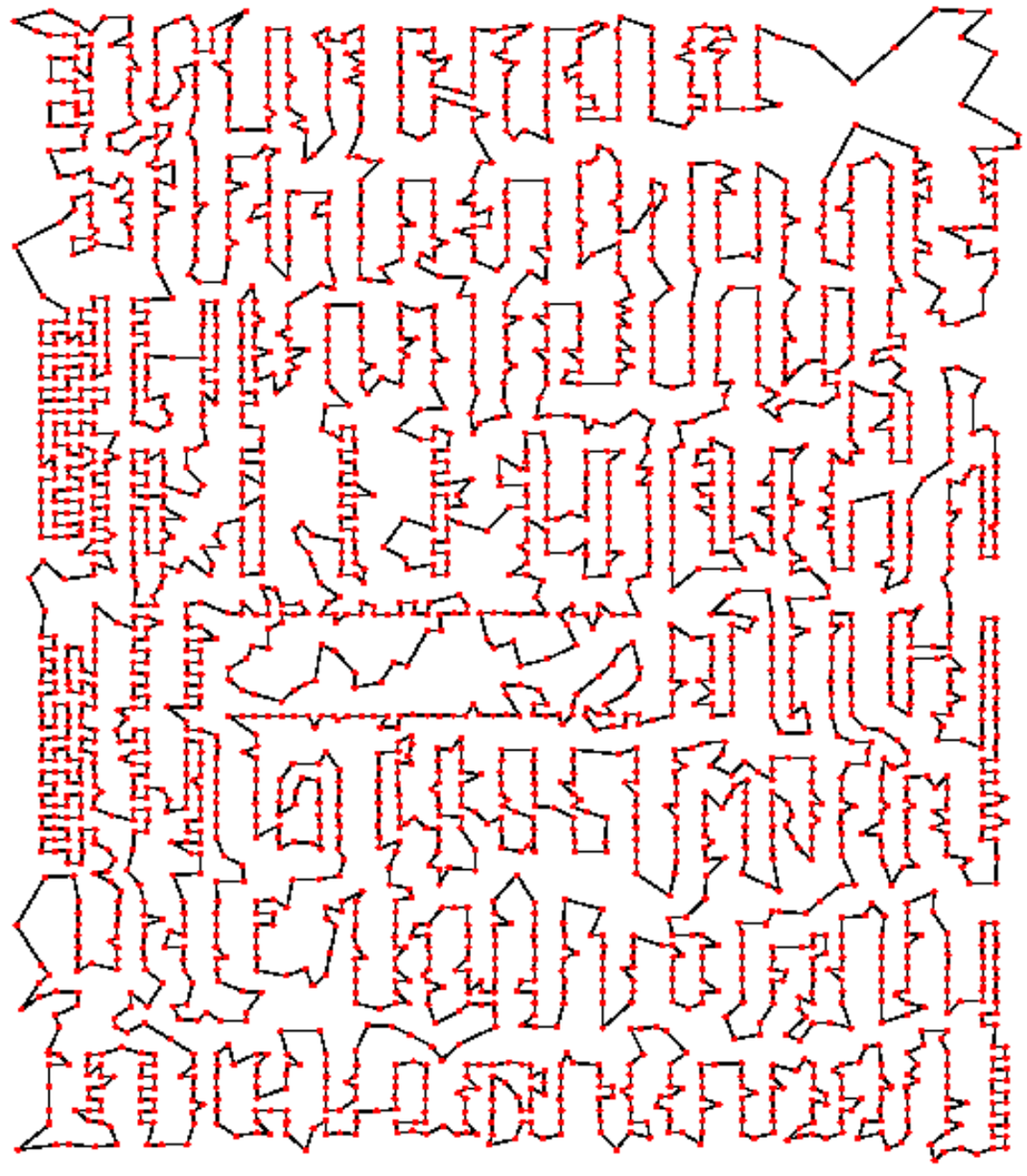
Length 1436



CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

History

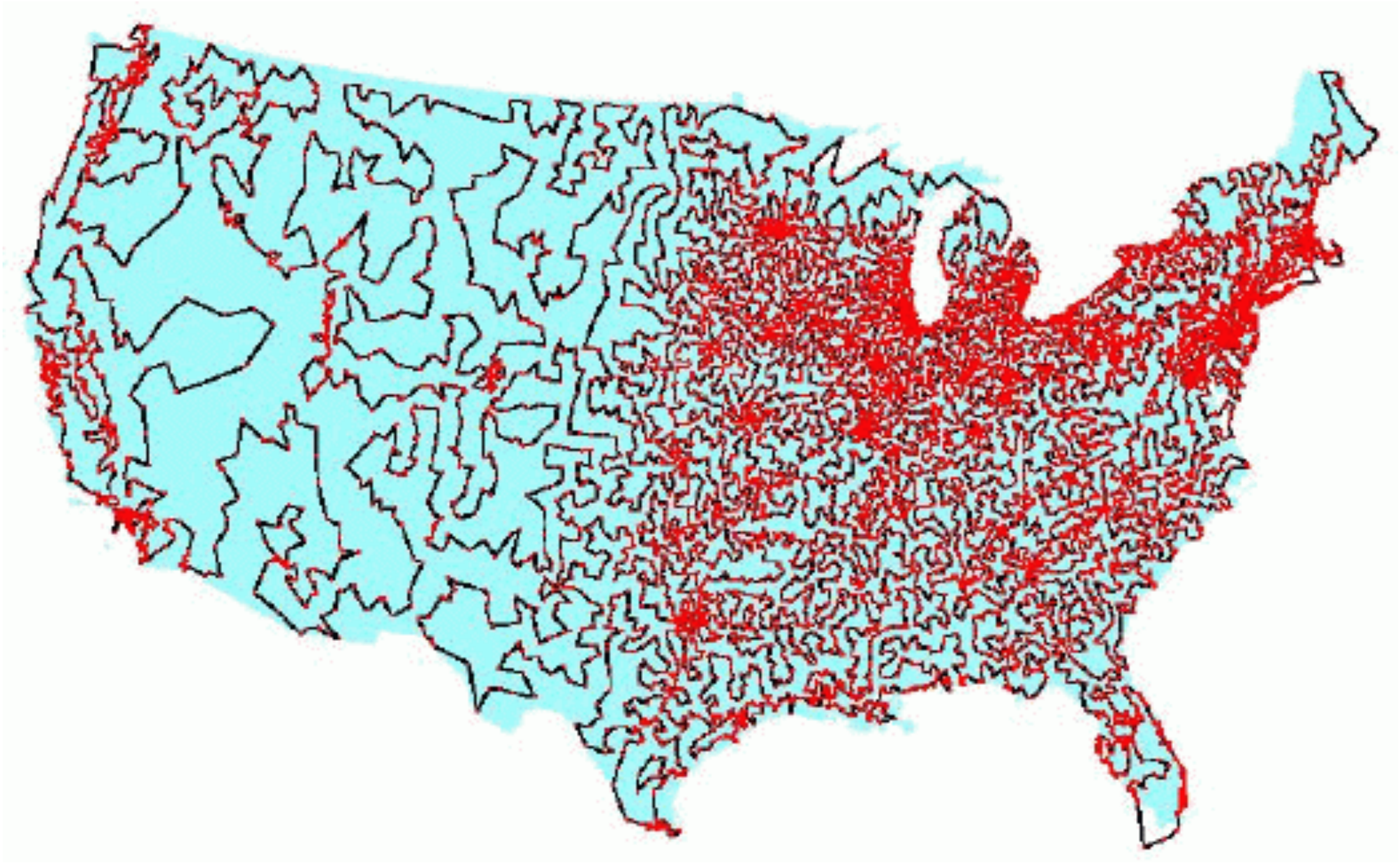
- 1987 Padberg and Rinaldi — Printed Circuit Board 2,392 cities



CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

History

- 1998 Applegate, Bixby, Chvátal and Cook — USA Towns of 500 or more population with13,509 cities



CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

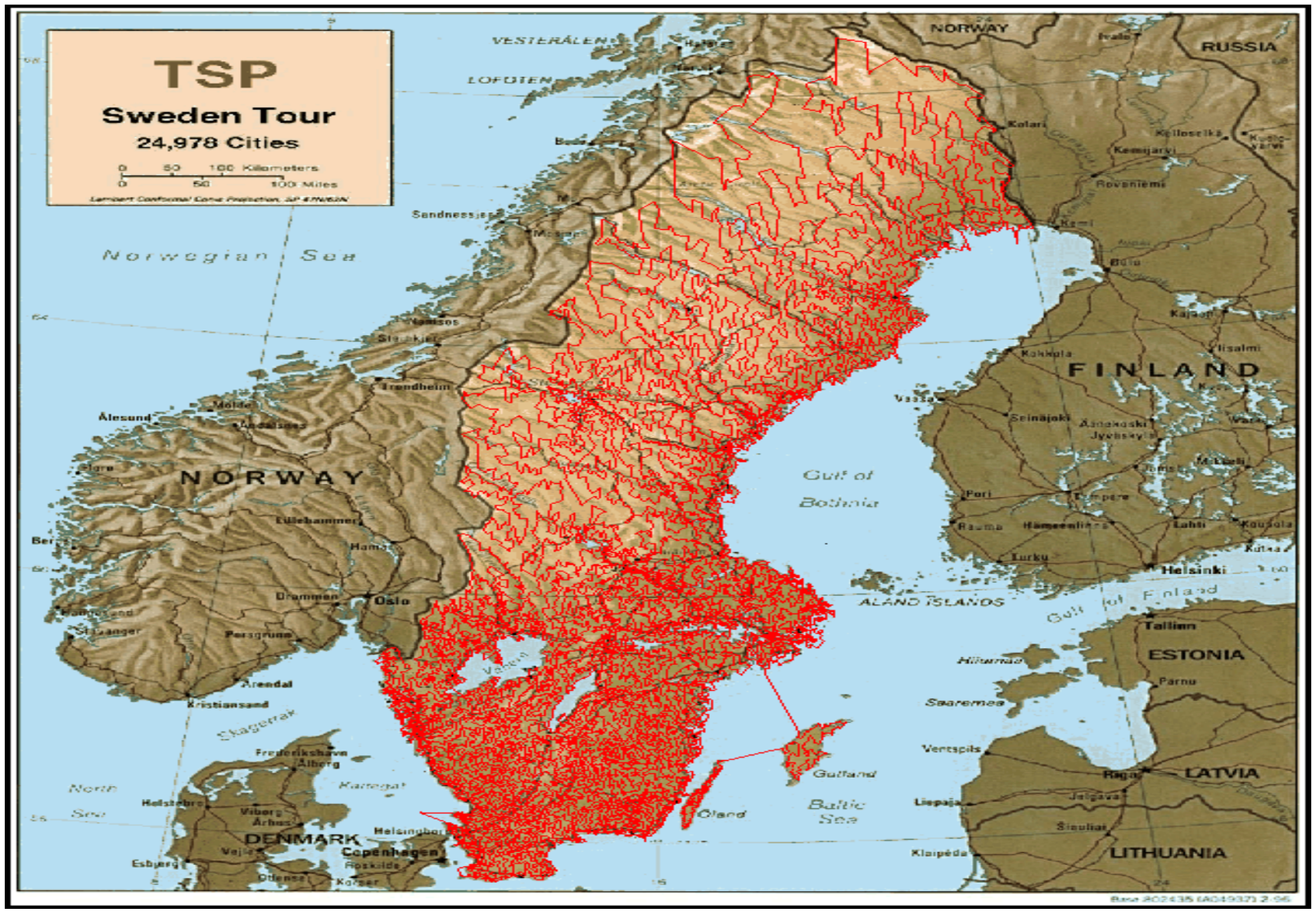
History

- 2001 Applegate, Bixby, Chvátal and Cook — Towns in Germany 15,112 Cities

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

History

- 2004 Applegate, Bixby, Chvátal, Cook and Helsgaun — Sweden 24,978 Cities



CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Reference

- https://algs4.cs.princeton.edu/home/
- Coursera 'Shortest Paths Revisited, NP-Complete Problems and What To Do About Them'
- https://www.youtube.com/watch?v=SC5CX8drAtU
- personal.lse.ac.uk/williahp/talks/The_Travelling_Salesman_Problem.ppt

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.