Announcements

- Reminder: HW1 due tomorrow at 6 pm
- This lecture will be shorter, can answer questions afterwards

CS6161: Design and Analysis of Algorithms (Fall 2020)

Graphs, and Travelling Salesman Problem

Instructor: Haifeng Xu

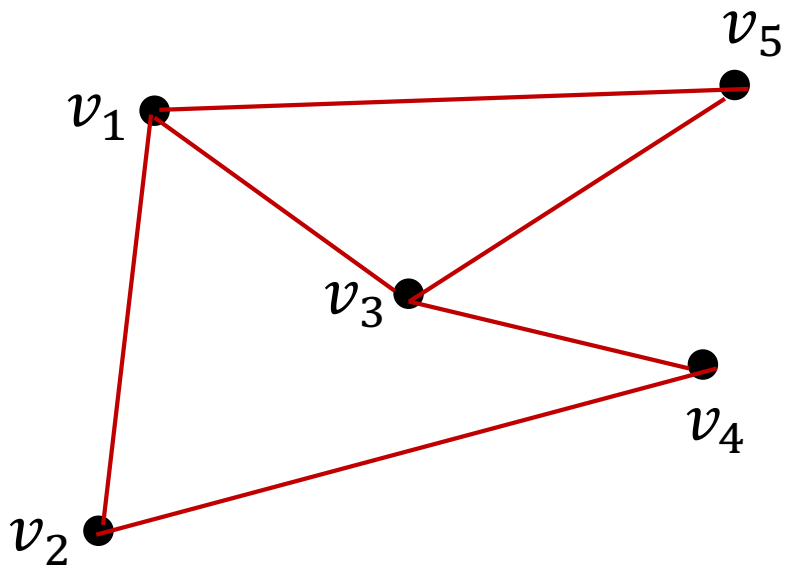
Outline

- Graphs
- Intro to Travelling Salesman Problem (TSP)
- DP for TSP

What is a Graph?

Two components:

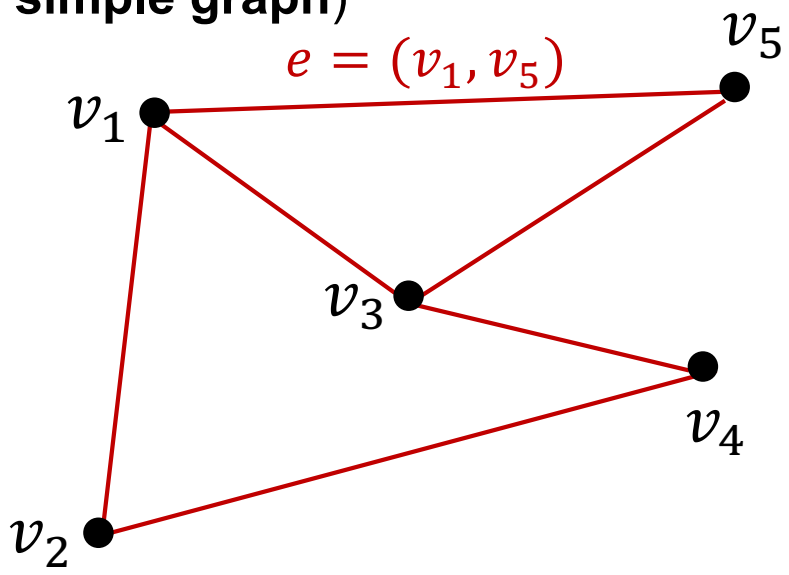
- Set of nodes V
- Set of edges E



What is a Graph?

Two components:

- Set of nodes V
- Set of edges E (can be undirected or directed)
  - For undirected graphs, typically at most one edge between each two nodes (called a **simple graph**)

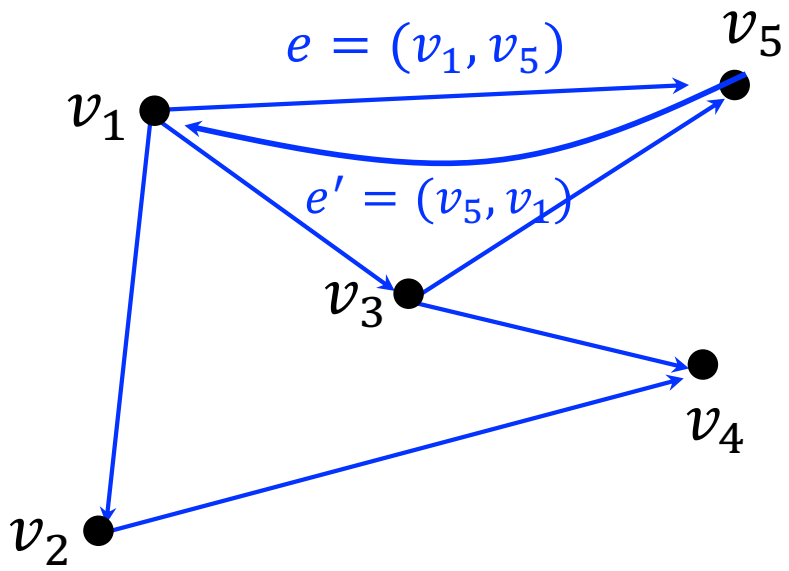


(v_1, v_5) and (v_5, v_1)
represent the same edge

What is a Graph?

Two components:

- Set of nodes V
- Set of edges E (can be undirected or directed)



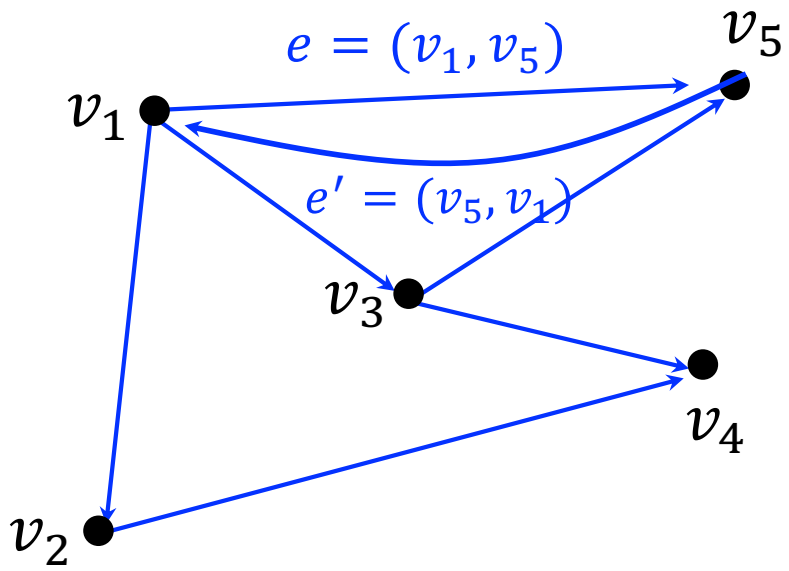
(v_1, v_5) and (v_5, v_1) are different edges

What is a Graph?

Two components:

- Set of nodes V
- Set of edges E (can be undirected or directed)

Often denoted as $G = (V, E)$

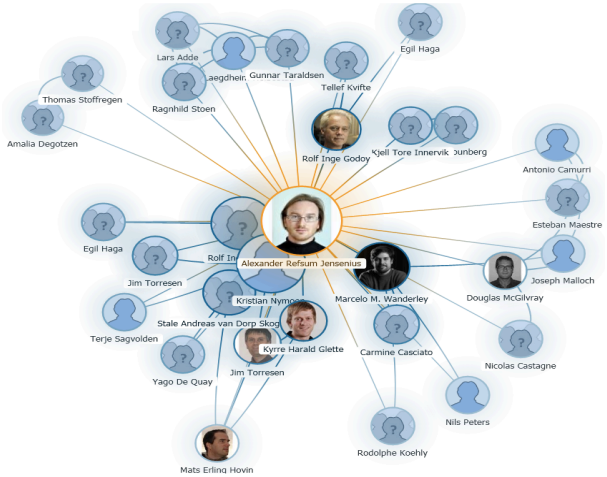


(v_1, v_5) and (v_5, v_1) are different edges

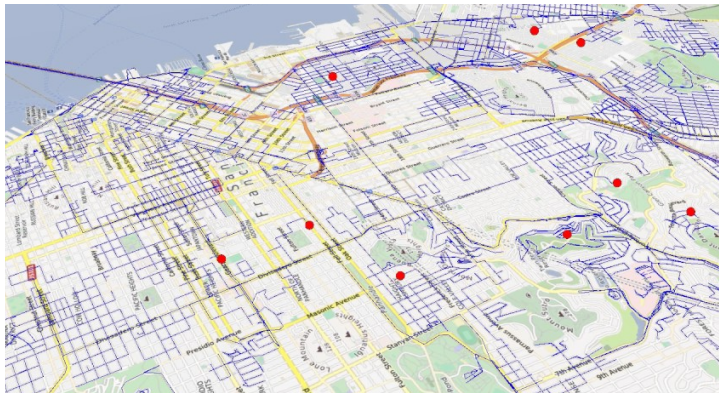
Graphs Are Ubiquitous

- Undirected graphs

Facebook



Coauthor graph

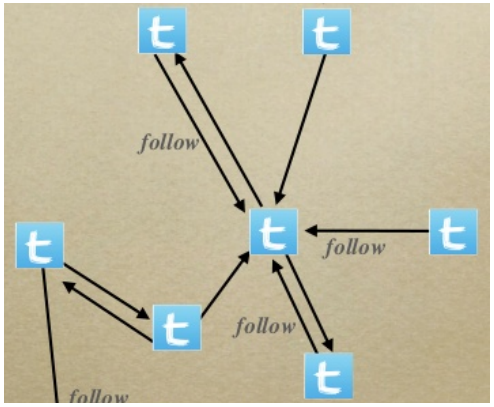


Traffic network

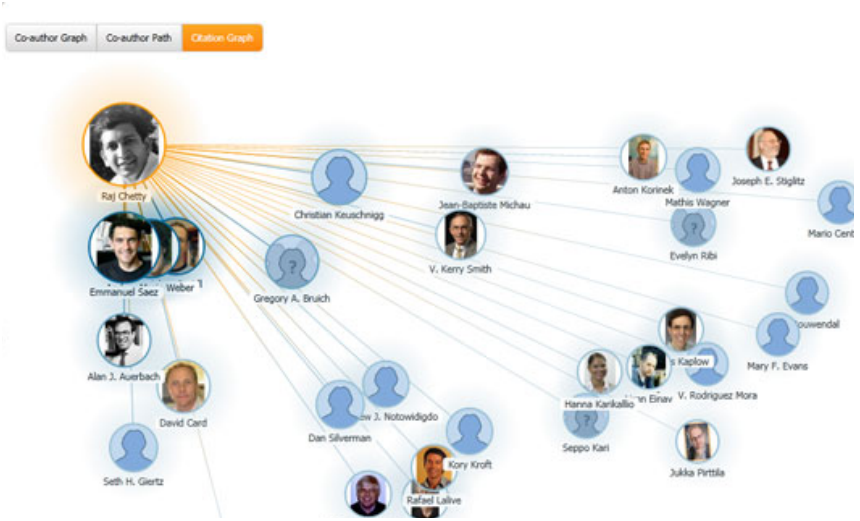
Cyber network

Graphs Are Ubiquitous

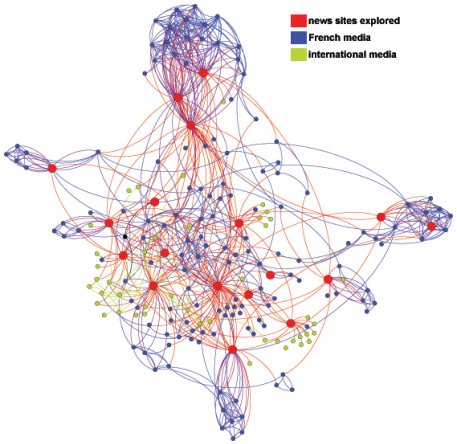
- Directed graphs



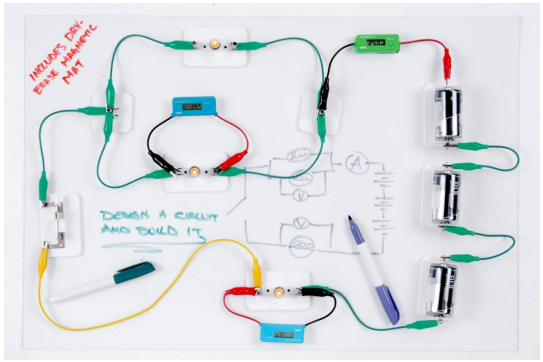
Twitter



Citation graph



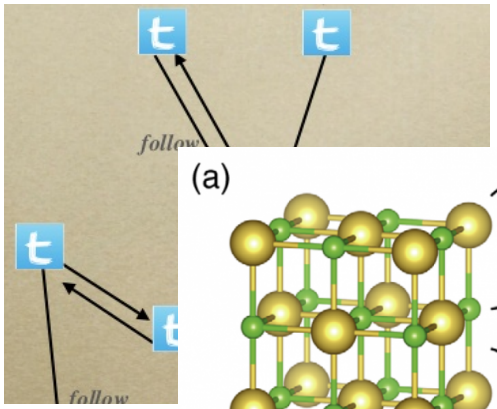
Websites link graph



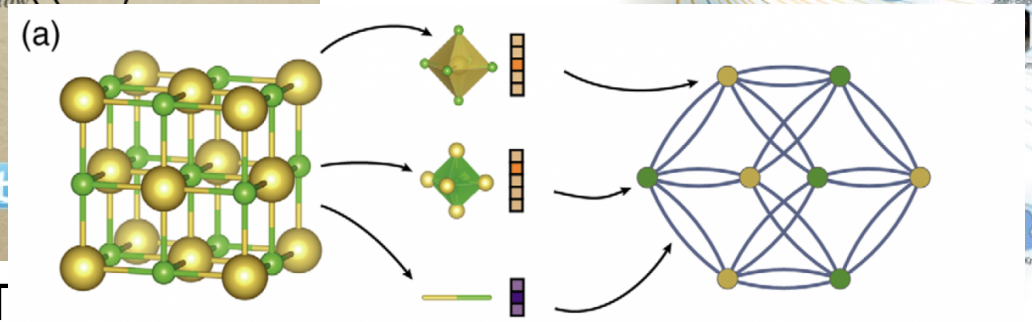
Electric circuit

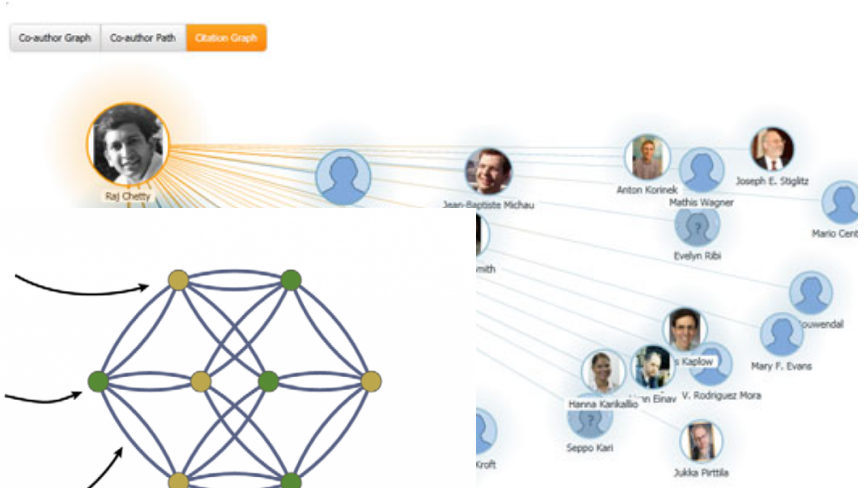
Graphs Are Ubiquitous

- Directed graphs

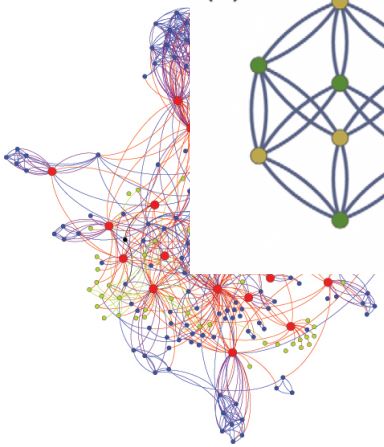


T

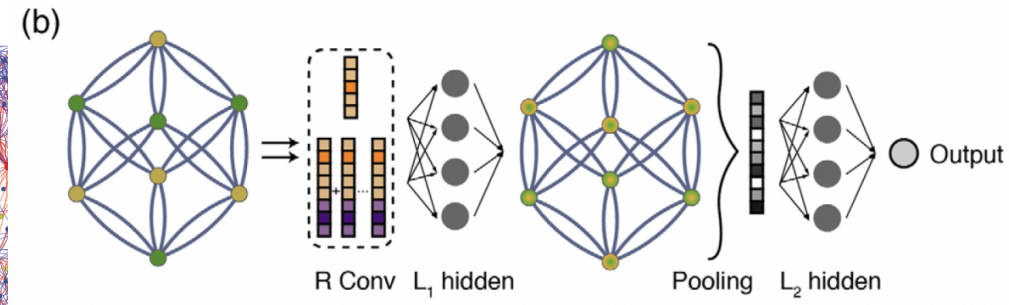


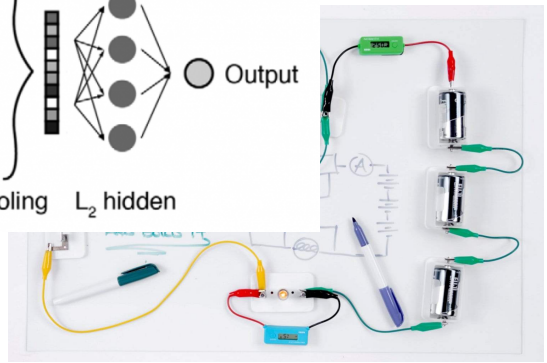


graph



Websites link graph





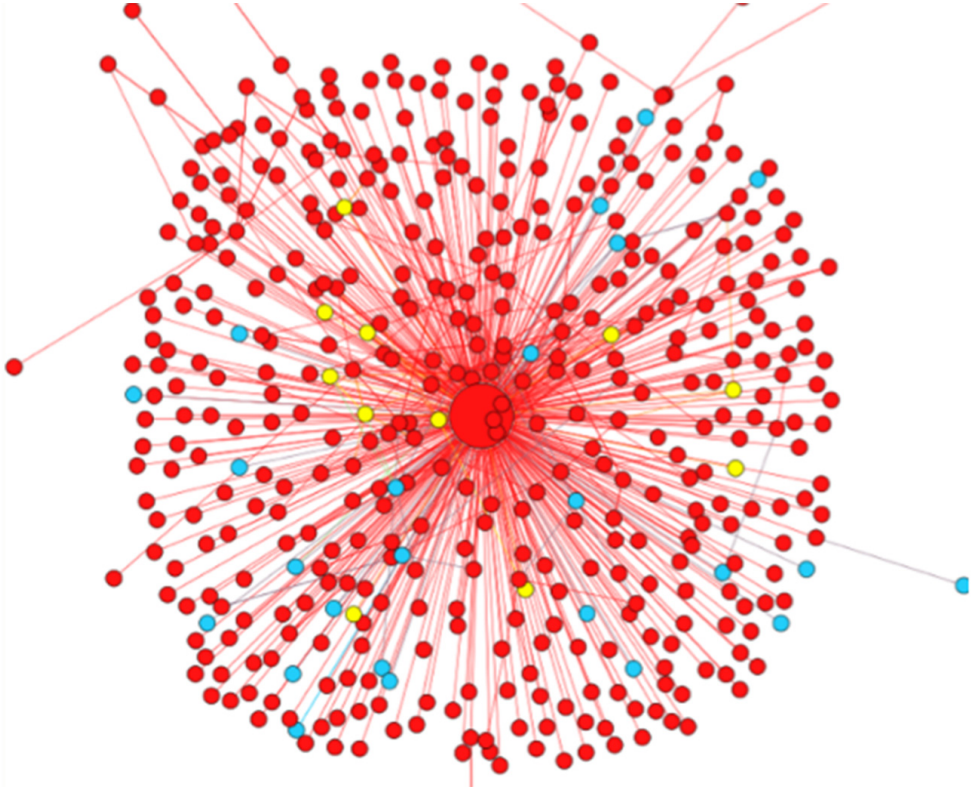
Electric circuit

Graphs Are Ubiquitous

The most important model to capture “connectivity” or “relation”

Many Properties Can Be Studied

- Connectivity: whether all components are connected
- Centrality: which nodes are the central nodes



Many Properties Can Be Studied

- Connectivity: whether all components are connected
- Centrality: which nodes are the central nodes
- Radius of a component

Many Properties Can Be Studied

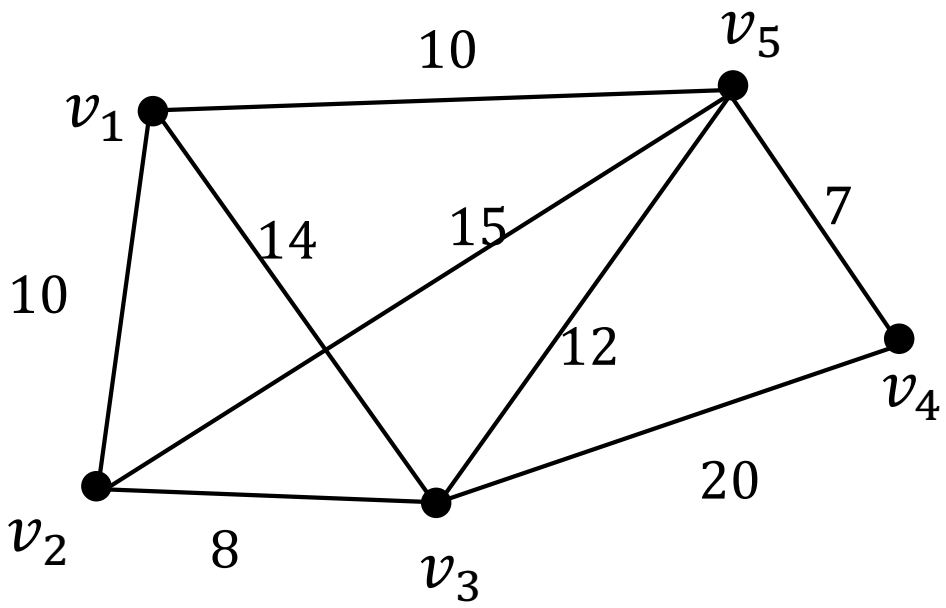
- Connectivity: whether all components are connected
- Centrality: which nodes are the central nodes
- Radius of a component
- Average degree, shortest path, minimum cut…

Outline

- Graphs
- Intro to Travelling Salesman Problem (TSP)
- DP for TSP

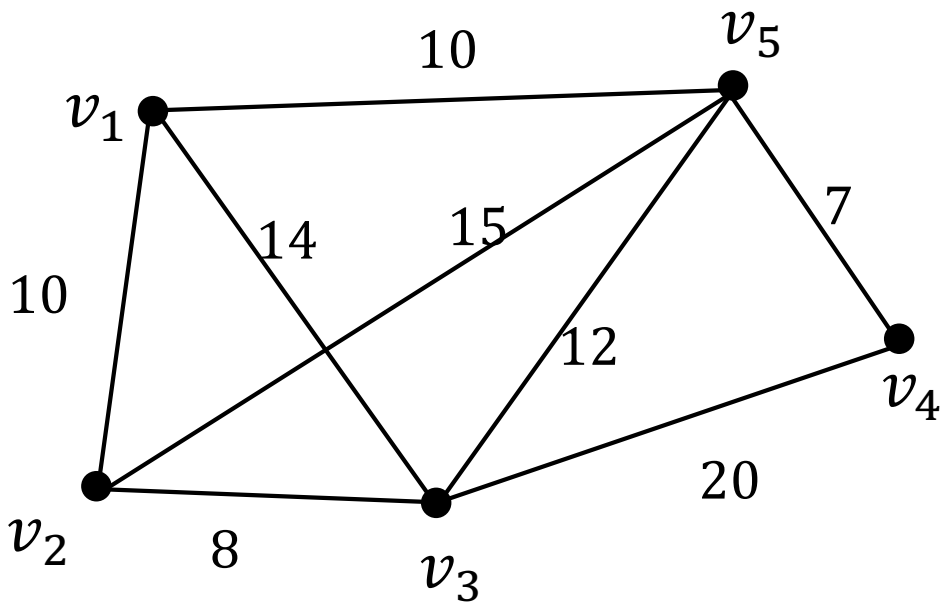
Weighted Graph

- Edges have weights
  - E.g., model the distance between two nodes
- Described by $G = (V, E, \{c_e\}_{e \in E})$



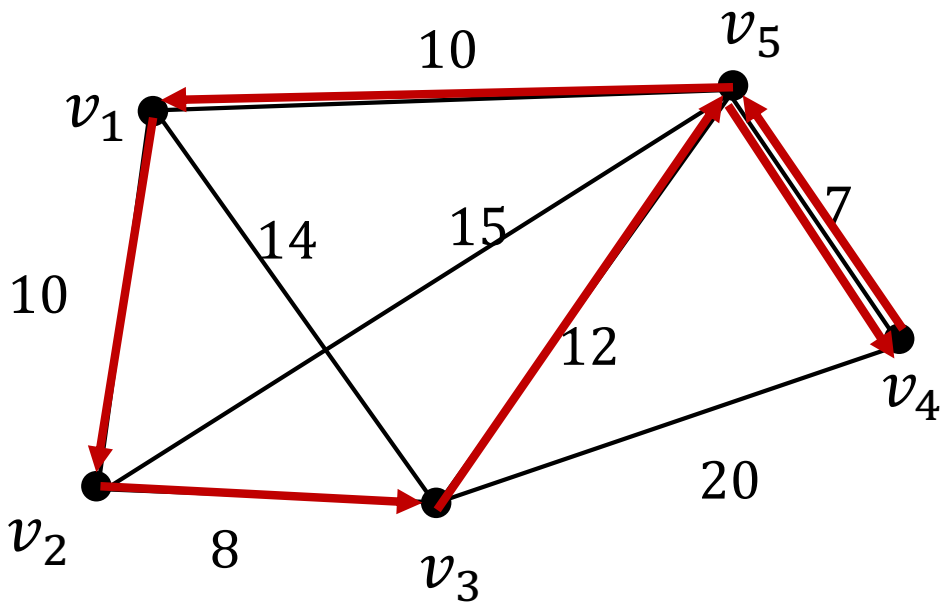
The Travelling Salesman Problem (TSP)

- Finding a route that traverse through all nodes with minimum total cost



The Travelling Salesman Problem (TSP)

- Finding a route that traverse through all nodes with minimum total cost
  - A node may be traversed for multiple times
  - The cost minimization goal to prevent you to do this, unless you really have to

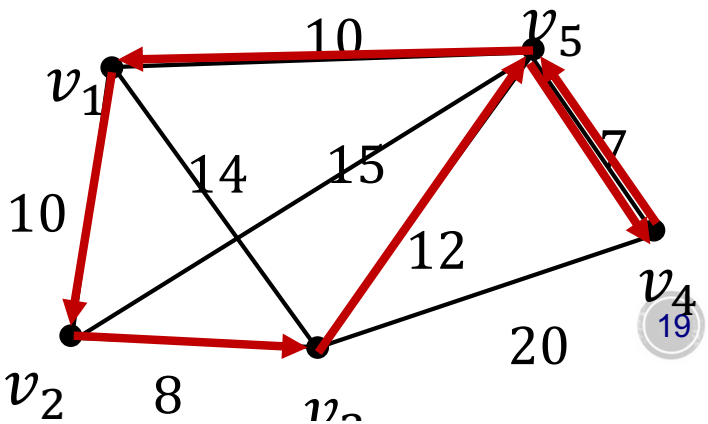


A Straightforward Algorithm

- Simply try all possible routes

How many possible routes in total?

- $O(n!)$: each feasible route = an order to nodes to be visited
- Slightly trickier since we may visit the same node twice
  - Not a big issue, can view $v_3 \rightarrow v_5 \rightarrow v_4$ as visiting v_4 through a shortest path
  - Shortest path can be computed quickly



Can We Do Better?

- Yes, but not that much…

Fact: TSP is NP-hard.

Will cover proof later in this course

Why We Still Want to Solve the Problem?

- It has many applications, and is useful
  - Shows up a lot in planning, logistics, manufacturing
  - A sub-problem of DNA sequencing
  - Testing microchips
  - Astronomy: minimize time spent moving telescope between different sources
  - …
- It is an important problem in the theory of algorithm analysis
  - Many studies on approximation algorithms
  - Its study resulted in several best papers on top Theoretical Computer Science (TCS) conferences (e.g., SODA 2010, FOCS 2011)

Next, we will give an $O(2^n)$ exact algorithm based on DP

(better than the $O(n!)$ naïve algorithm)

Later in this course, will cover approximation algorithms for TSP

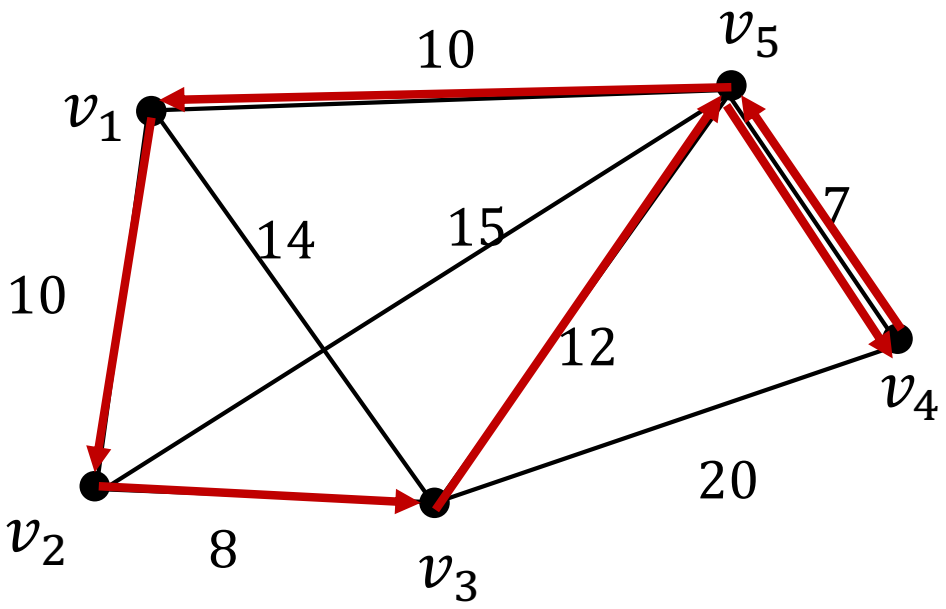
Outline

- Graphs
- Intro to Travelling Salesman Problem (TSP)
- DP for TSP

DP for TSP

Recall: two key steps

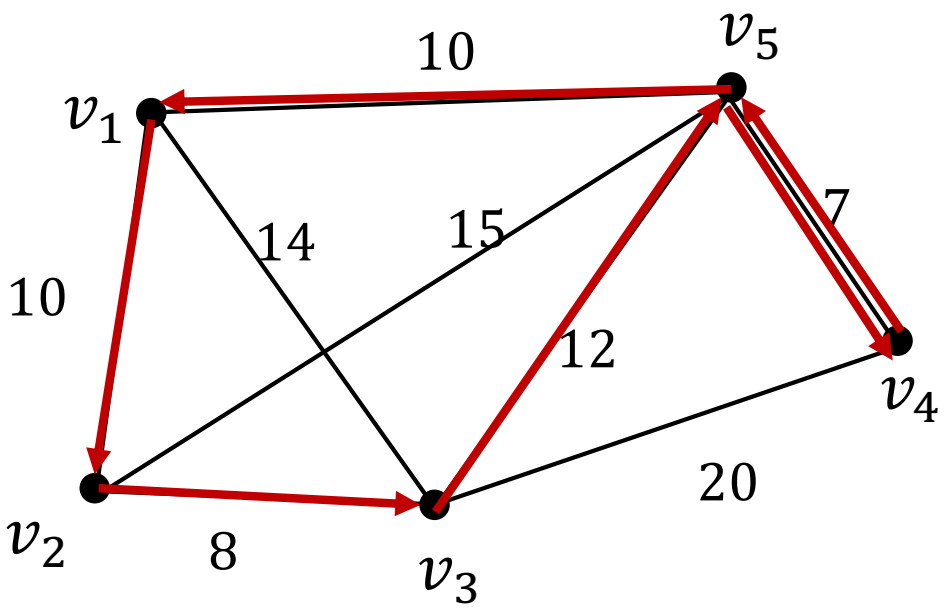
- What are the sub-problems?
- What is the order for solving sub-problems?



TSP Sub-Problems

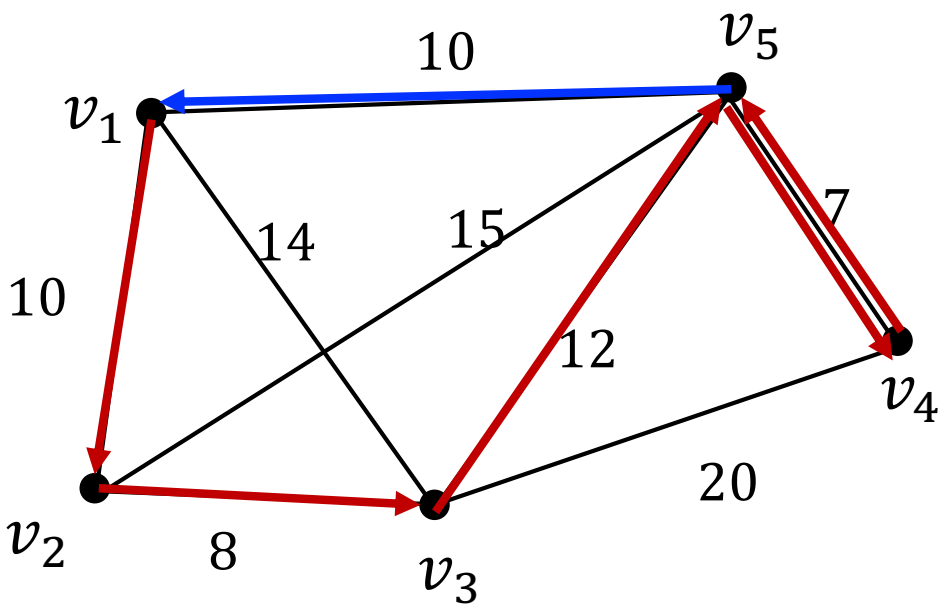
- Optimal path: $v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow v_4 \rightarrow v_5 - v_1$

By taking shortest path $v_3 \rightarrow v_5 \rightarrow v_4$



TSP Sub-Problems

- Optimal path: $v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow v_4 \rightarrow v_5 - v_1$
- If we know the last node before returning back to v_1 is v_5, then we only need to: (1) find minimum-cost route from v_1 to v_5 that traverses over all nodes; (2) find shortest path from v_5 back to v_1

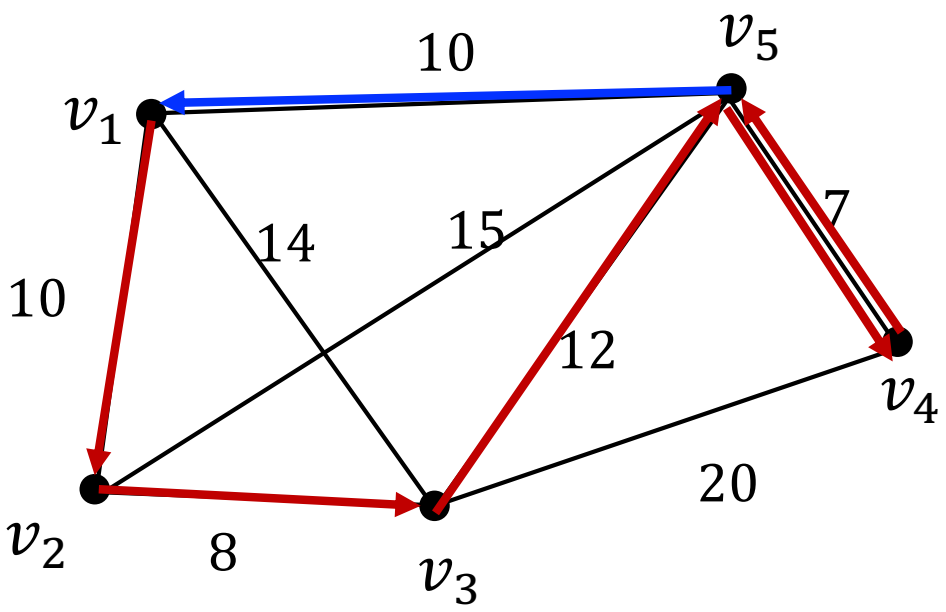


TSP Sub-Problems

- Optimal path: $v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow v_4 \rightarrow v_5 - v_1$
- If we know the last node before returning back to v_1 is v_5, then we only need to: (1) find minimum-cost route from v_1 to v_5 that traverses over all nodes; (2) find shortest path from v_5 back to v_1

An easy problem

This separates the problem into two problems (red and blue part)

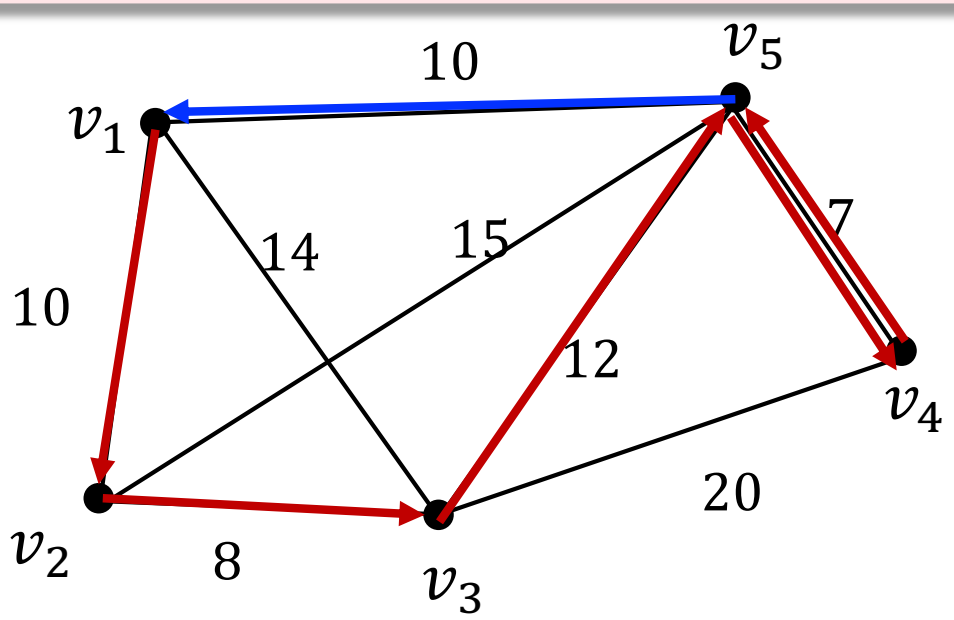


TSP Sub-Problems

- Optimal path: $v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow v_4 \rightarrow v_5 - v_1$
- If we know the last node before returning back to v_1 is v_5, then we only need to: (1) find minimum-cost route from v_1 to v_5 that traverses over all nodes; (2) find shortest path from v_5 back to v_1

Subproblem $C(v_1, v_i)$ for all i

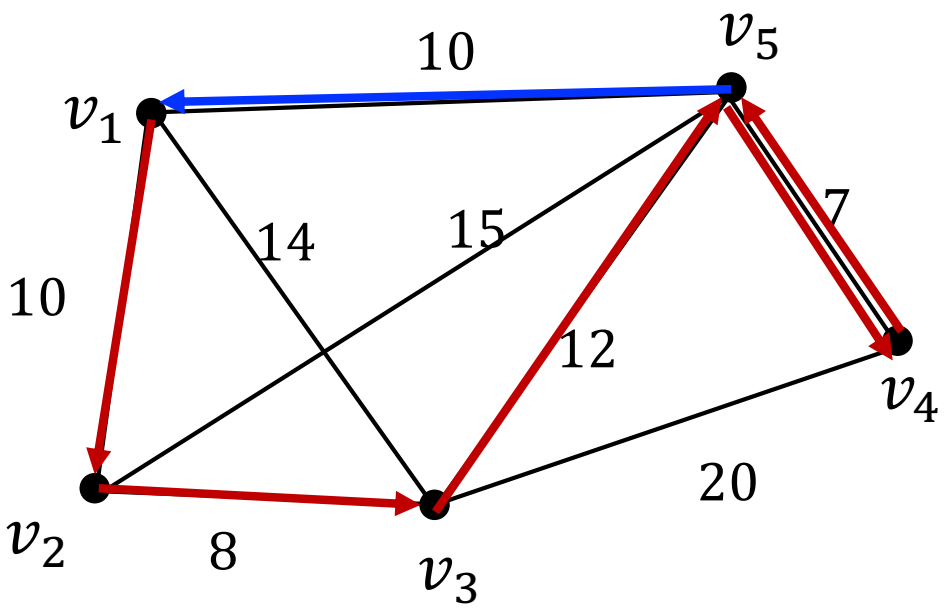
find minimum-cost route from v_1 to v_i that traverses over all nodes



Subproblem $C(v_1, v_i)$ for all i

find minimum-cost route from v_1 to v_i that traverses over all nodes

- But this problem is still difficult to resolve…
  - In fact, it is still NP-hard
  - Why? If we can solve $C(v_1, v_i)$ efficiently for all i, a solution for TSP can be computed easily from $C(v_1, v_i)$
- Need further splitting of $C(v_1, v_i)$



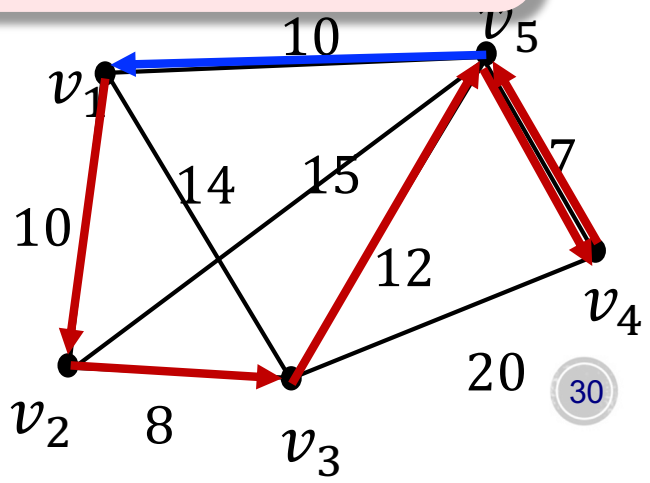
Subproblem $C(v_1, v_i)$ for all i

find minimum-cost route from v_1 to v_i that traverses over all nodes

- But this problem is still difficult to resolve…
  - In fact, it is still NP-hard
  - Why? If we can solve $C(v_1, v_i)$ efficiently for all i, a solution for TSP can be computed easily from $C(v_1, v_i)$
- Need further splitting of $C(v_1, v_i)$

Subproblem $C(v_1, v_i; S)$ for any subset $S \subseteq V$ and $v_i \in S$

find minimum-cost route from v_1 to v_i that traverses over all nodes in set S



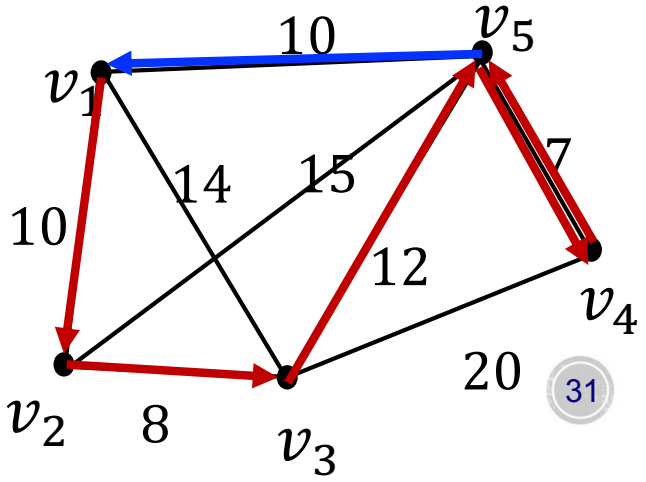
Subproblem $C(v_1, v_i; S)$ for any subset $S \subseteq V$ and $v_i \in S$
find minimum-cost route from v_1 to v_i that traverses over all nodes in set S

- How are the sub-problems related?
  - Interpret $C(v_1, v_i; S)$ as the minimum cost

$$C(v_1, v_i; S) = \min_{v_j \in S}[C(v_1, v_j; S - \{v_i\}) + dist(v_j, v_i)]$$

Has a smaller size than S

The size of S also gives a natural order to build up solutions from subproblems



DP for TSP

TSP-DP($G = (V, E, \{c_e\}_{e \in E})$)

1

2

3

4

5

6

7

DP for TSP

TSP-DP($G = (V, E, \{c_e\}_{e \in E})$)

1 Build table $C[i, S]$ for any $S \subseteq V, i \in V$

2

3

4

5

6

7

DP for TSP

TSP-DP($G = (V, E, \{c_e\}_{e \in E})$)

1 Build table $C[i, S]$ for any $S \subseteq V, i \in V$

2 **for** all S of size 2 and $v_i \in S$

3 $C[i, S] = dist(v_1, v_i)$

4

5

6

7

DP for TSP

TSP-DP($G = (V, E, \{c_e\}_{e \in E})$)

1 Build table $C[i, S]$ for any $S \subseteq V, i \in V$

2 **for** all S of size 2 and $v_i \in S$

3 $C[i, S] = dist(v_1, v_i)$

4 **for** $k = 3, \ldots, n$

5 **for** all S of size k and $v_i \in S$

6

7

DP for TSP

TSP-DP($G = (V, E, \{c_e\}_{e \in E})$)

1 Build table $C[i, S]$ for any $S \subseteq V, i \in V$

2 **for** all S of size 2 and $v_i \in S$

3 $\quad C[i, S] = dist(v_1, v_i)$

4 **for** $k = 3, \ldots, n$

5 $\quad$ **for** all S of size k and $v_i \in S$

6 $\quad\quad C[i, S] = \min_{v_j \in S}[C[j, S - \{v_i\}] + dist(v_j, v_i)]$

7

DP for TSP

TSP-DP($G = (V, E, \{c_e\}_{e \in E})$)

1 Build table $C[i, S]$ for any $S \subseteq V, i \in V$

2 **for** all S of size 2 and $v_i \in S$

3 $\quad C[i, S] = dist(v_1, v_i)$

4 **for** $k = 3, \ldots, n$

5 $\quad$ **for** all S of size k and $v_i \in S$

6 $\quad\quad C[i, S] = \min_{v_j \in S}[C[j, S - \{v_i\}] + dist(v_j, v_i)]$

7 **return** $\min_{v_i \in S} C[i, V] + dist(v_i, v_1)$

Running Time Analysis

TSP-DP($G = (V, E, \{c_e\}_{e \in E})$)

1 Build table $C[i, S]$ for any $S \subseteq V, i \in V$

2 **for** all S of size 2 and $v_i \in S$

3 $\quad C[i, S] = dist(v_1, v_i)$

4 **for** $k = 3, \ldots, n$

5 $\quad$ **for** all S of size k and $v_i \in S$

6 $\quad\quad C[i, S] = \min_{v_j \in S}[C[j, S - \{v_i\}] + dist(v_j, v_i)]$

7 **return** $\min_{v_i \in S} C[i, V] + dist(v_i, v_1)$

- Two for loops in total enumerate $\approx 2^n$ subsets S
- At most n within the loop → $O(n2^n)$ in total

Running Time Analysis

TSP-DP($G = (V, E, \{c_e\}_{e \in E})$)

1 Build table $C[i, S]$ for any $S \subseteq V, i \in V$

2 **for** all S of size 2 and $v_i \in S$

3 $\quad C[i, S] = dist(v_1, v_i)$

4 **for** $k = 3, \ldots, n$

5 $\quad$ **for** all S of size k and $v_i \in S$

6 $\quad\quad C[i, S] = \min_{v_j \in S}[C[j, S - \{v_i\}] + dist(v_j, v_i)]$

7 **return** $\min_{v_i \in S} C[i, V] + dist(v_i, v_1)$

Caveat: did not record the paths yet, but simple modification of the algorithm can help to record the optimal path as well

- End of Dynamic Programming
- Next lecture: greedy algorithms

Thank You

Haifeng Xu

University of Virginia

hx4ad@virginia.edu