Chap 2 Graph Routes

Yih-Lang Li (李毅郎)

Computer Science Department

National Yang-Ming Chiao-Tung University, Taiwan

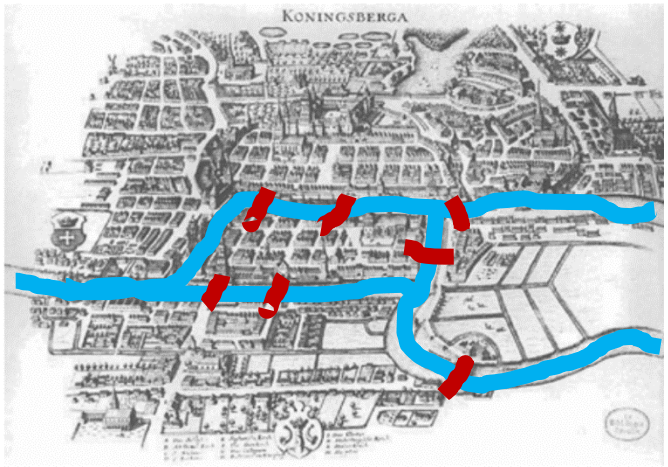
The sources of most figure images are from the textbook

Outline

- Eulerian Circuits
- Hamiltonian Cycles
- Shortest Paths

2.1 Eulerian Circuits – Königsberg Bridge Problem

Map of Königsberg

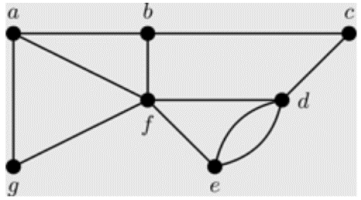


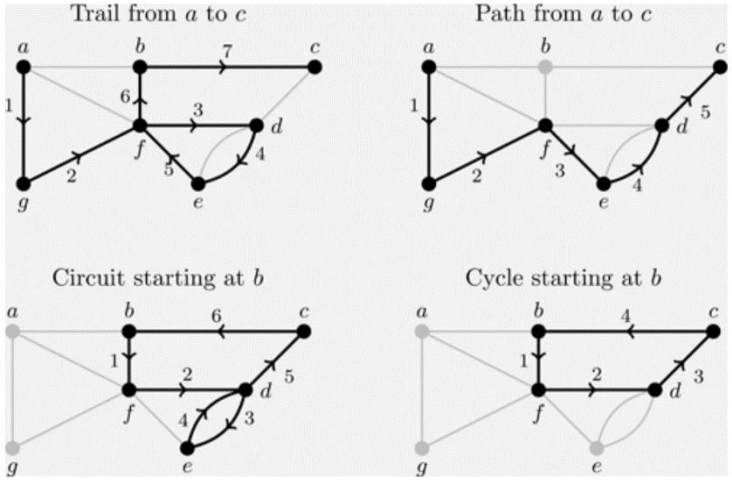
Map of Königsberg

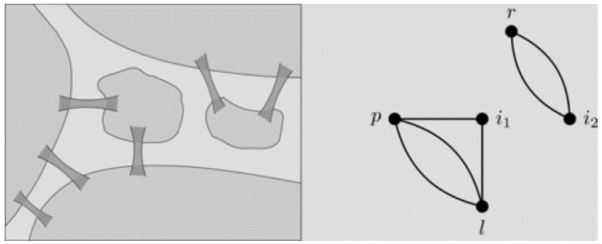
- **Definition 2.1** Let G be a graph.
  - ✓ A *walk* is a sequence of vertices so that there is an edge between consecutive vertices. A walk can repeat vertices and edges.
  - ✓ A *trail* is a walk with no repeated edges. A trail can repeat vertices but not edges.
  - ✓ A *path* is a trail with no repeated vertex (or edges). A path on n vertices is denoted P_n.
  - ✓ A *closed walk* is a walk that starts and ends at the same vertex.
  - ✓ A *circuit* is a closed trail; that is, a trail that starts and ends at the same vertex with no repeated edges though vertices may be repeated.
  - ✓ A *cycle* is a closed path; that is, a path that starts and ends at the same vertex. Thus cycles cannot repeat edges or vertices. Note: we do not consider the starting and ending vertex as being repeated since each vertex is entered and exited exactly once. A cycle on n vertices is denoted C_n.

Königsberg Bridge Problem

- The length of any of these tours is defined in terms of the number of edges. For example, P_n has length $n-1$ and C_n has length n.
- Example 2.1 Given the graph below, find a trail (that is not a path) from a to c, a path from a to c, a circuit (that is not a cycle) starting at b, and a cycle starting at b.



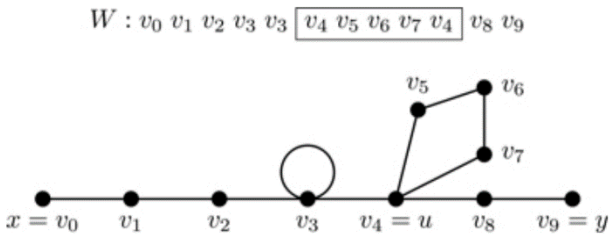


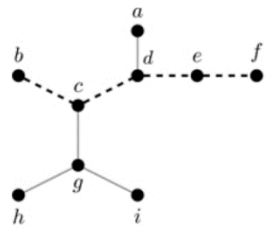


- **Definition 2.2** Let G be a graph. Two vertices x and y are connected if there exists a path from x to y in G. The graph G is connected if every pair of distinct vertices is connected.
- Example 2.2 Island City has two islands, a peninsula, and left and right banks, as shown on the left below. Modeling the relationship between the landmasses and bridges of Island City gives us the graph below on the right.

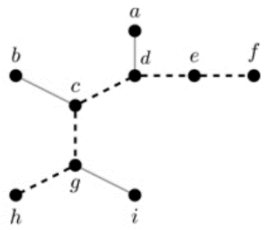
Königsberg Bridge Problem

- **Theorem 2.3** Every $x-y$ walk contains an $x-y$ path.
  - ✓ Prove by induction
  - ✓ No repeated vertex → no repeated edge
  - ✓ For each repeated vertex v, remove all vertices and edges from the first v to the edge in front of the second v such that only one v appears in the walk.



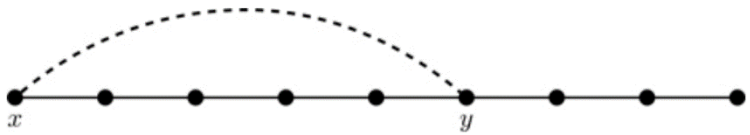


Maximal $b-f$ path



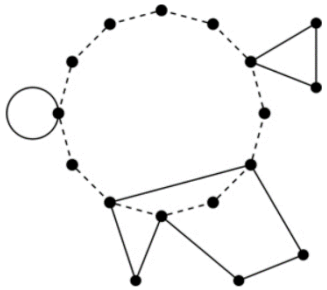
Maximum path in G

- **Definition 2.4** An object X is *maximum* if it is the largest among all objects under consideration; that is, $|X| \geq |A|$ for all $A \in U$.
- An object X is *maximal* if it cannot be made larger.
- **Theorem 2.5** If every vertex of a graph has degree at least 2 then G contains a cycle.

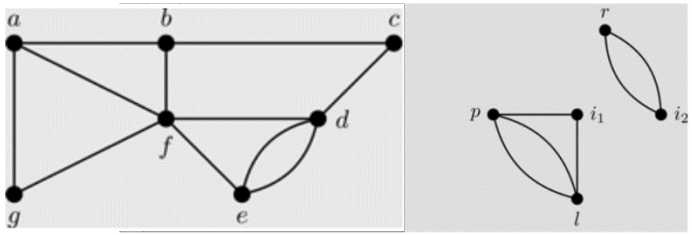


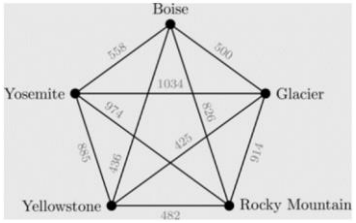
Eulerian Graphs

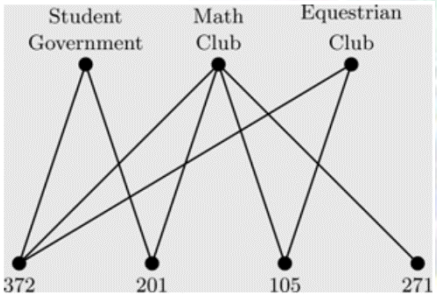
- **Definition 2.6** Let *G* be a graph. An *eulerian circuit* (or trail) is a circuit (or trail) that contains every edge and every vertex of *G*.
- If G contains an eulerian circuit it is called *eulerian* and if G contains an eulerian trail but not an eulerian circuit it is called *semi-eulerian*.
- **Theorem 2.7** A graph *G* is eulerian if and only if *G* is connected and every vertex has even degree.
  - ✓ →
  - ✓ ← By induction and by Theorem 2.5



- **Corollary 2.8** A graph *G* is semi-eulerian if and only if G is connected and exactly two vertices have odd degree.
- Example 2.3 Consider the graphs appearing in the examples from this and the previous chapter. Which ones are eulerian? semi-eulerian? neither?

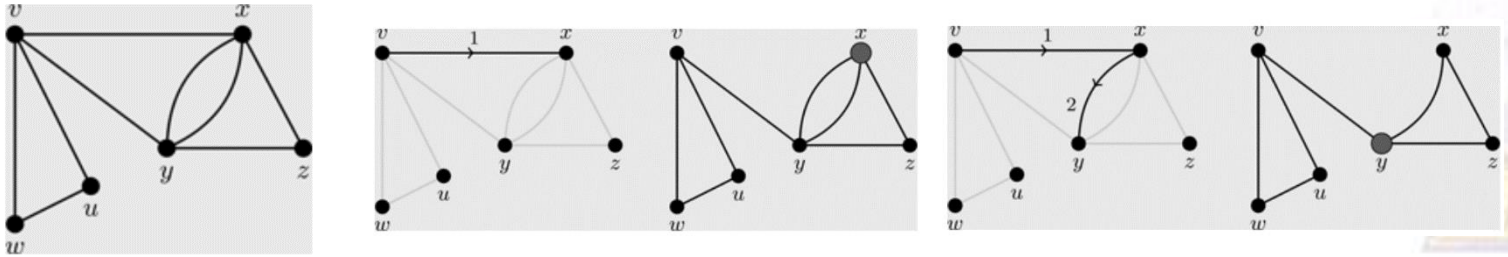




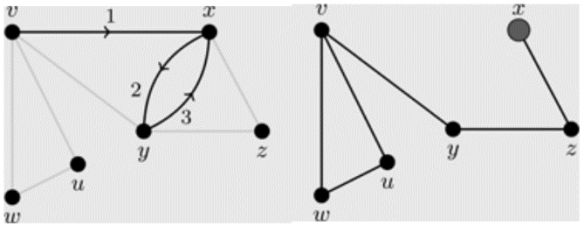
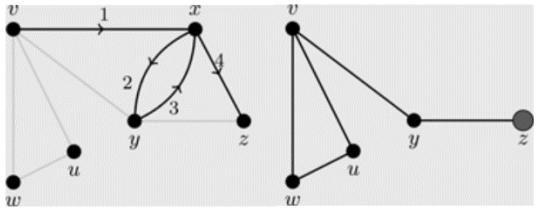
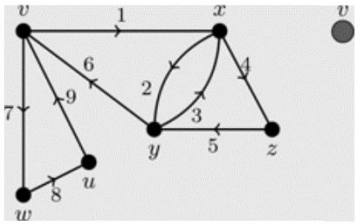


Algorithms

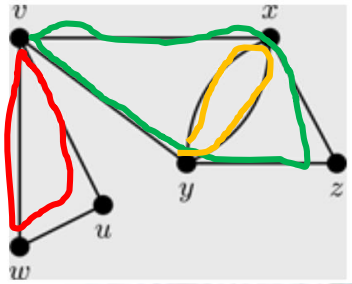
- **Fleury's Algorithm**
  - ✓ *Input*: Connected graph *G* where zero or two vertices are odd.
  - ✓ *Steps*:
  - ✓ (1) Choose a starting vertex, call it *v*. If *G* has no odd vertices, then any vertex can be the starting point. If *G* has exactly two odd vertices, then *v* must be one of the odd vertices.
  - ✓ (2) Choose *an edge incident to v that is unlabeled* and label it with the number in which it was chosen, *ensuring that the graph consisting of unlabeled edges remains connected*.
  - ✓ (3) Travel along the edge to its other endpoint.
  - ✓ (4) Repeat Steps (2) and (3) until all edges have been labeled.
  - ✓ Output: Labeled eulerian circuit or trail.
- Example 2.4 Input: A connected graph (shown below) where every vertex has even degree. We are looking for an eulerian circuit.



Algorithms

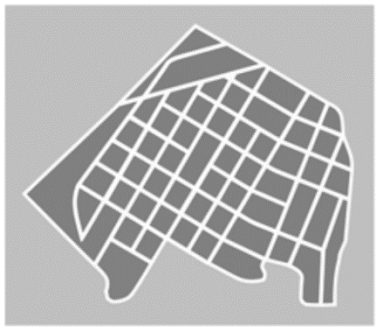
- **Hierholzer's Algorithm**
  - ✓ *Input*: Connected graph *G* where all vertices are even.
  - ✓ Steps:
  - ✓ (1) Choose a starting vertex, call it *v*. Find a circuit *C* originating at *v*.
  - ✓ (2) If any vertex *x* on *C* has edges not appearing in *C*, find a circuit *C′* originating at *x* that uses two of these edges.
  - ✓ (3) Combine *C* and *C′* into a single circuit *C**.
  - ✓ (4) Repeat Steps (2) and (3) until all edges of *G* are used.
  - ✓ Output: Labeled eulerian circuit.



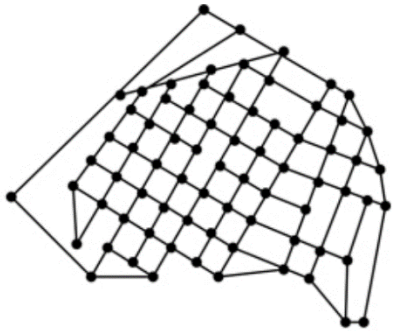
- Example 2.5 Input: A connected graph where every vertex has even degree. We are looking for an eulerian circuit.

Applications

- Snowplow routes

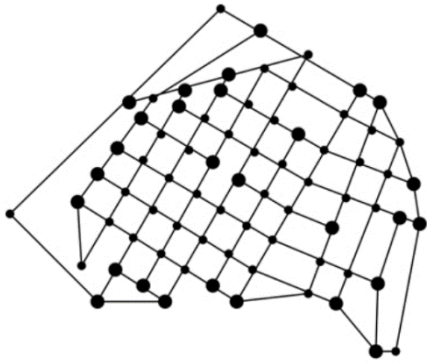


Crystal Spring Map

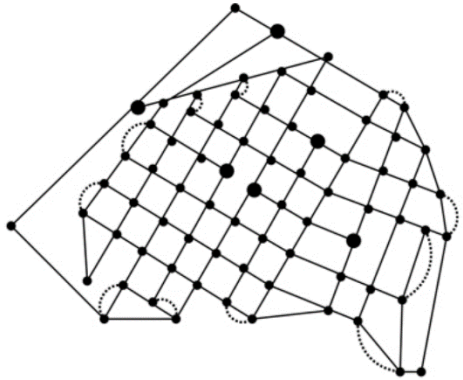


Crystal Spring Graph G_1

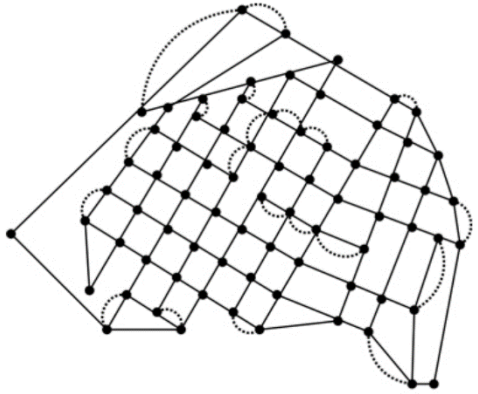
- **Definition 2.9** Given a connected graph $G=(V,E)$, an *eulerization* of G is the graph $G'=(V,E')$ so that (i) G' is obtained by duplicating edges of G, and (ii) every vertex of G' is even. A *semi-eulerization* of G results in a graph G' so that (i) G' is obtained by duplicating edges of G, and (ii) exactly two vertices of G' are odd.



G_1 with odd vertices in bold



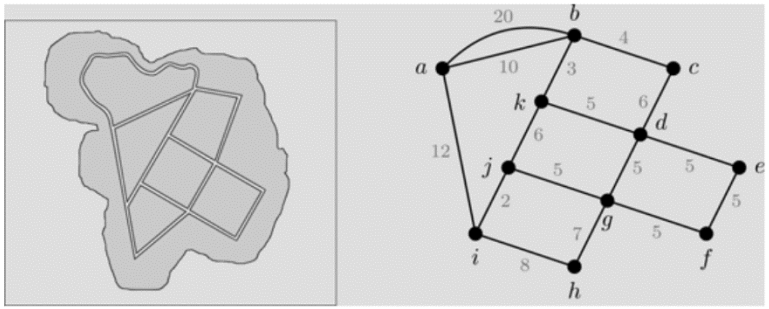
G_1 with edge duplications between adjacent odd vertices

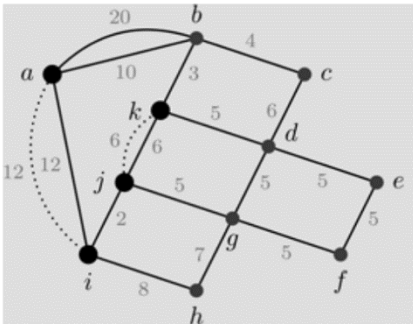


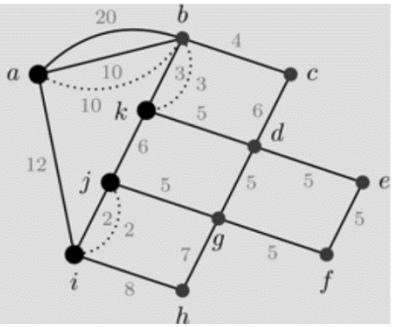
Optimal eulerization of G_1

Applications

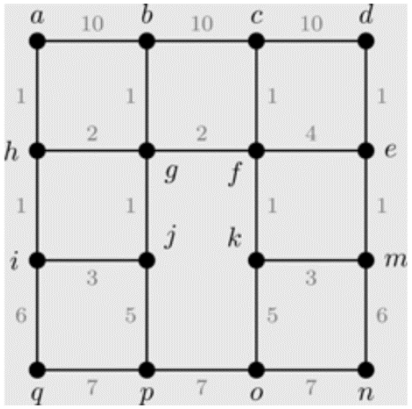
- Chinese postman problem – by Guan Mei-Gu
- Example 2.6 the patrolman on Sunset Island

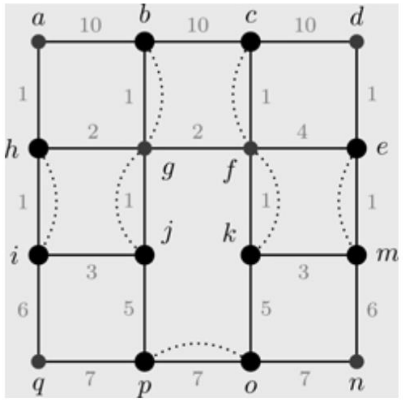






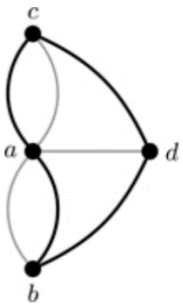
- Example 2.7 Find an optimal Eulerization for the graph below where the weights given are in terms of cost.

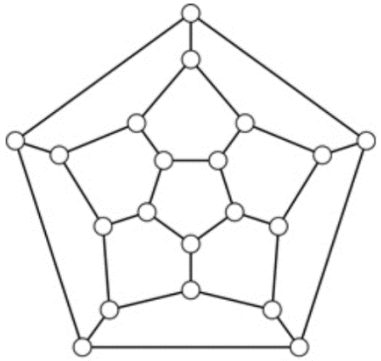




2.2 Hamiltonian Cycles

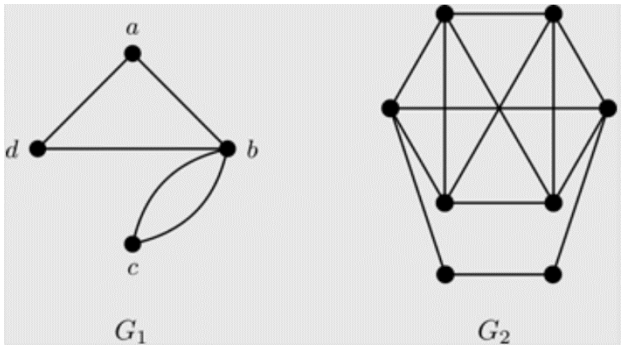
- **Definition 2.10** A cycle in a graph G that contains every vertex of G is called a *hamiltonian cycle*. A path that contains every vertex is called a *hamiltonian path*. A graph that contains a hamiltonian cycle is called *hamiltonian*.

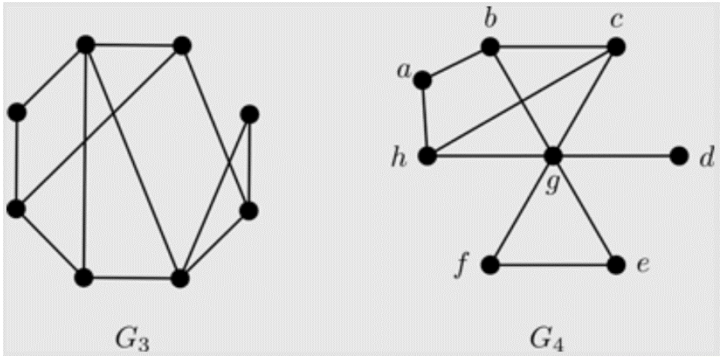




Icosian Game

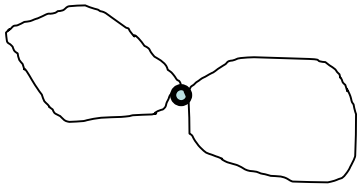
- Eulerian vs. Hamiltonian
- Example 2.8 For each of the graphs below, determine if they have hamiltonian cycles (and paths) and eulerian circuits (and trails).





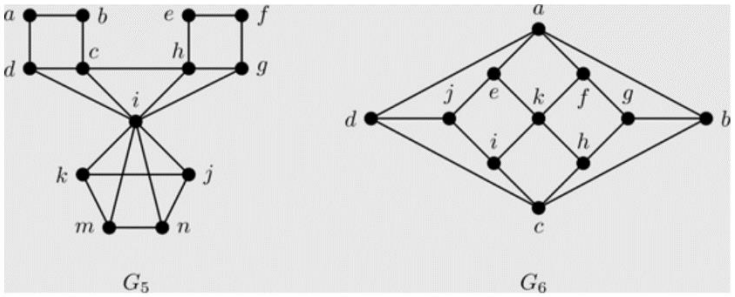
Hamiltonian Cycles

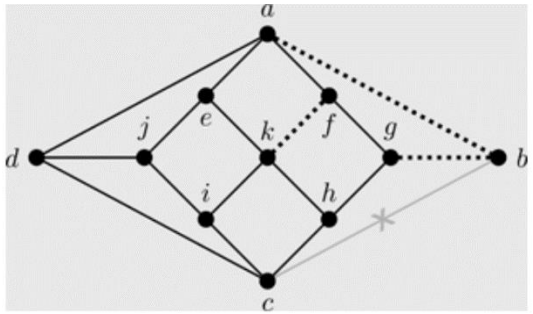
- Recall: if a graph has an eulerian circuit, then it cannot have an eulerian trail, and vice versa.
  - ✓ How about Hamiltonian cycle and path?
- **Properties** of Hamiltonian Graphs

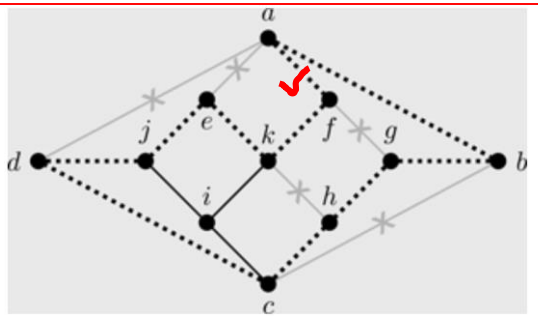


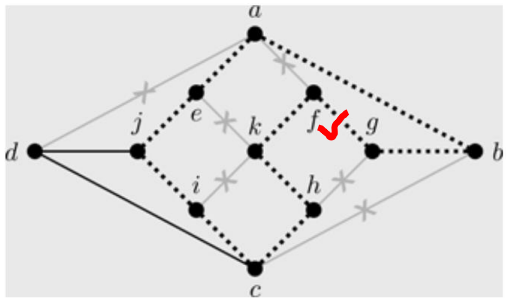
  - ✓ *G* must be connected.
  - ✓ No vertex of *G* can have degree less than 2.
  - ✓ *G* cannot contain a cut-vertex, that is a vertex whose removal disconnects the graph.
  - ✓ If *G* contains a vertex *x* of degree 2 then both edges incident to *x* must be included in the cycle.
  - ✓ If two edges incident to a vertex *x* must be included in the cycle, then all other edges incident to *x* cannot be used in the cycle.
  - ✓ If in the process of attempting to build a hamiltonian cycle, a cycle is formed that does not span *G*, then *G* cannot be hamiltonian.
- Example 2.9 Use the properties listed above to show that the graphs below are not hamiltonian.

Hamiltonian Cycles

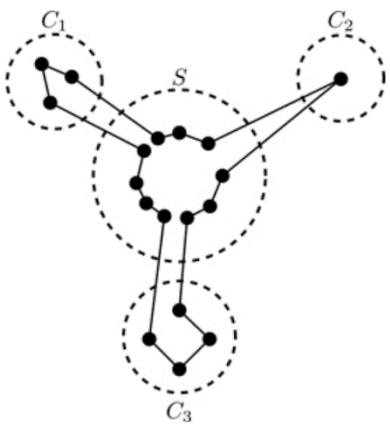






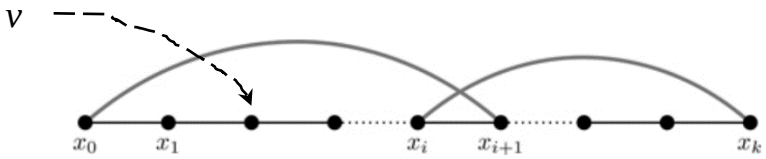


- **Proposition 2.11** If G is a graph with a hamiltonian cycle, then $G-S$ has at most $|S|$ components for every nonempty set $S \subseteq V$.



Hamiltonian Cycles

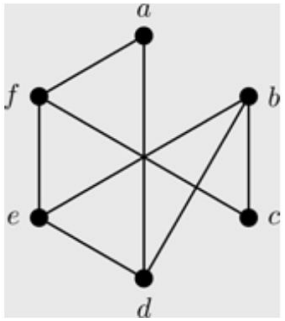
- **Theorem 2.12** (*Dirac's Theorem*) Let G be a graph with $n \geq 3$ vertices. If every vertex of G satisfies $deg(v) \geq n/2$, then G has a hamiltonian cycle.
  - ✓ *Claim* 1: G is connected
  - ✓ *Claim* 2: Let $P=x_0x_1\cdots x_k$ be a longest path of G. Then every neighbor of both x_0 and x_k must be included in the path.
  - ✓ Let $S=\{x_i: x_ix_k \in E(G)\}$. $T=\{x_i: x_{i+1}x_0 \in E(G)\}$.
  - ✓ $|S|+|T| \geq n$. In addition, $x_k \notin S$ and $x_k \notin T$, $|S \cup T| < n$.

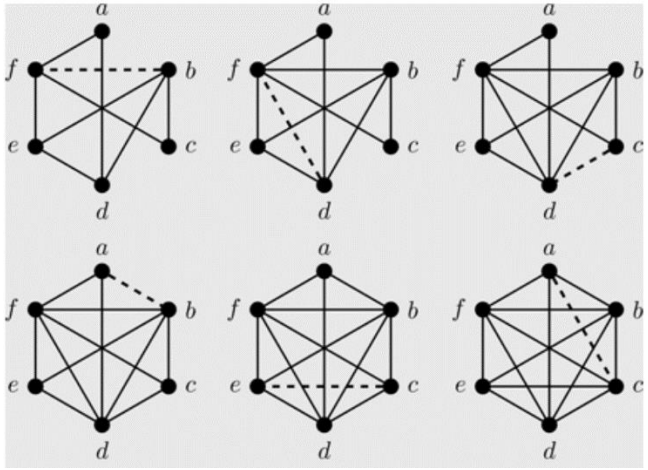


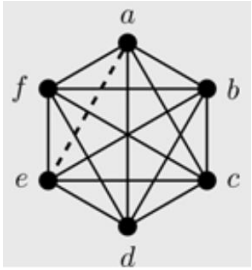
  - ✓ But as $|S \cup T| = |S| + |T| - |S \cap T|$, we have that $|S \cap T| \geq 1$, and so there exists some vertex x_i such that $x_0x_{i+1}, x_ix_k \in E(G)$.
  - ✓ Consider the cycle $C = x_0 \underset{P}{\rightarrow} x_i\, x_k \underset{P}{\rightarrow} x_{i+1}\, x_0$, C spans G.
- **Theorem 2.13** (*Ore's Theorem*) Let G be a graph with $n \geq 3$ vertices. If $deg(x)+deg(y) \geq n$ for all distinct nonadjacent vertices, then G has a hamiltonian cycle.
- **Definition 2.14** The hamiltonian closure of a graph G, denoted $cl(G)$, is the graph obtained from G by iteratively adding edges between pairs of nonadjacent vertices whose degree sum is at least n, that is we add an edge between x and y if $deg(x)+deg(y) \geq n$, until no such pair remains.

Hamiltonian Cycles

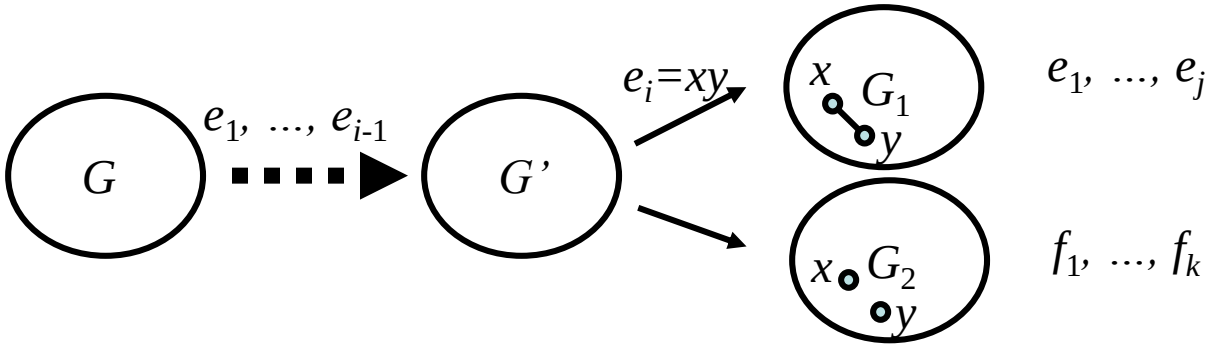
- Example 2.10 Find a hamiltonian closure for the graph below.





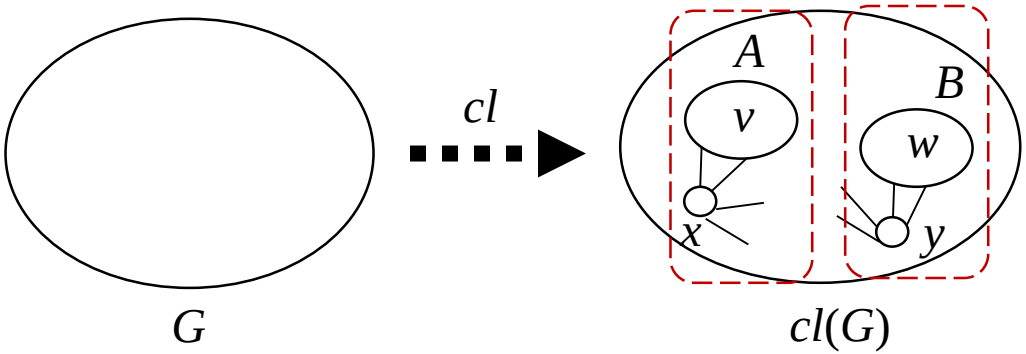


- **Theorem 2.15** The closure of G is well-defined.
  - ✓ Suppose for a contradiction that this is not true.



Hamiltonian Cycles

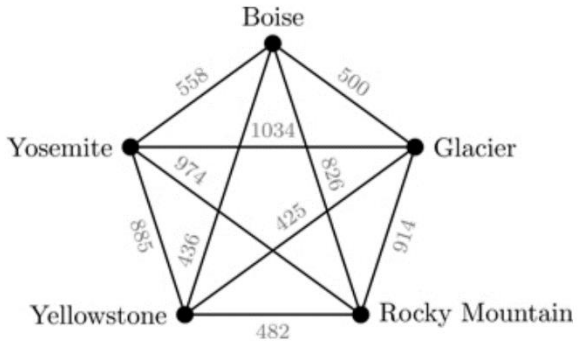
- **Theorem 2.16** A graph G is hamiltonian if and only if its closure $cl(G)$ is hamiltonian.
- **Lemma 2.17** If G is a graph with at least 3 vertices such that its closure $cl(G)$ is complete, then G is hamiltonian.
- **Theorem 2.18** Let G be a simple graph where the vertices have degree $d_1,d_2,\ldots,d_n$ such that $n\geq 3$ and the degrees are listed in increasing order. If for any $i<n/2$ either $d_i>i$ or $d_{n-i}\geq n-i$, then G is hamiltonian.
  - ✓ Suppose for a contradiction that $cl(G)$ is not complete.
  - ✓ Assume $k=deg_{cl(G)}(x) \leq deg_{cl(G)}(y)$,
  - ✓ $deg_{cl(G)}(x)+deg_{cl(G)}(y) \leq n\text{-}1$,
  - ✓ $deg_{cl(G)}(x)+deg_{cl(G)}(y)$ is maximum



1. $deg_{cl(G)}(y) \leq n-1-k$ and $k \leq (n-1)/2 < n/2$
2. $|A|=n-1-deg_{cl(G)}(y) \geq k$
3. $deg_G(v) \leq deg_{cl(G)}(v) \leq deg_{cl(G)}(x)=k$
4. $d_k \leq k$
5. $|B|=n-1-deg_{cl(G)}(x)=n-1-k$
6. $deg_G(w) \leq deg_{cl(G)}(w) \leq deg_{cl(G)}(y) \leq n-1-k$
7. $d_{n-k-1} \leq n\text{-}k\text{-}1$
8. $d_{n-k} \leq n\text{-}k\text{-}1$, $d_{n-k} < n\text{-}k$

The Traveling Salesman Problem

- Answering if a graph is Hamiltonian is difficult. Now we turn to answer what is the best Hamiltonian cycle on a weighted Hamiltonian graph.



- **Brute Force Algorithm**
  - ✓ *Input*: Weighted complete graph K_n.
  - ✓ *Steps*:

    1. Choose a starting vertex, call it *v*.

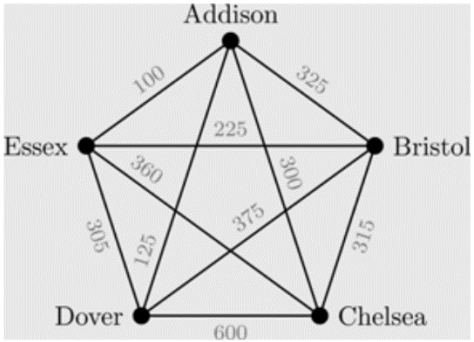
    2. Find all hamiltonian cycles starting at *v*. Calculate the total weight of each cycle.

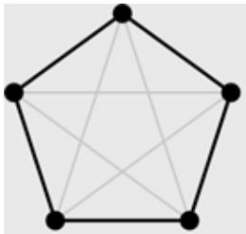
    3. Compare all $(n-1)!$ cycles. Pick one with the least total weight. (Note: there should be at least two options).

  - ✓ *Output*: Minimum hamiltonian cycle.

The Traveling Salesman Problem

- Example 2.11





- Brute-force computation

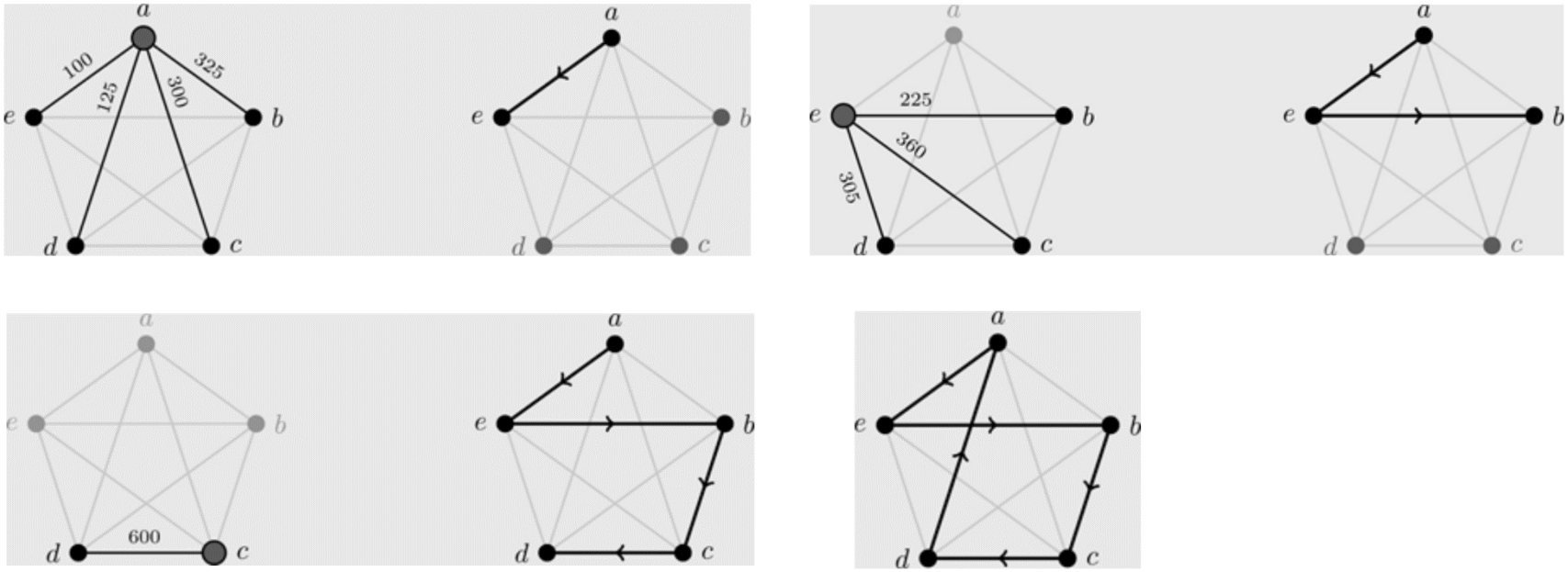
n	Best	Top 500
5	2×10^{-15} seconds	2×10^{-16} seconds
15	2×10^{-5} seconds	2×10^{-6} seconds
20	40 seconds	4 seconds
21	14 minutes	2 minutes
22	5 hours	32 minutes
23	4.5 days	12 hours
24	16 weeks	12 days
25	7.5 years	10 months
26	2 centuries	2 decades
30	132 million years	14 million years
40	4.1×10^{23} years	4.3×10^{22} years
50	1.4×10^{40} years	1.5×10^{39} years

The Traveling Salesman Problem

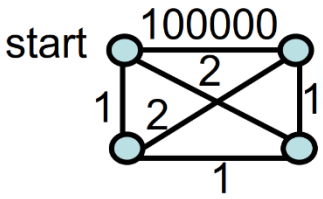
- **Nearest Neighbor Algorithm**
  - ✓ *Input*: Weighted complete graph K_n.
  - ✓ *Steps*:
    1. Choose a starting vertex, call it *v*. Highlight *v*.
    2. Among all edges incident to *v*, pick the one with the smallest weight. If two possible choices have the same weight, you may randomly pick one.
    3. Highlight the edge and move to its other endpoint *u*. Highlight *u*.
    4. Repeat Steps (2) and (3), where only edges to unhighlighted vertices are considered.
    5. Close the cycle by adding the edge to *v* from the last vertex highlighted. Calculate the total weight.
  - ✓ *Output*: hamiltonian cycle.

The Traveling Salesman Problem

- Example 2.12 Apply the Nearest Neighbor Algorithm to the graph from Example 2.11.



- The drawback of the nearest neighbor algorithm



The Traveling Salesman Problem

- **Repetitive Nearest Neighbor Algorithm**
  - ✓ *Input*: Weighted complete graph K_n.
  - ✓ *Steps*:

    1. Choose a starting vertex, call it v.
    2. Apply the Nearest Neighbor Algorithm.
    3. Repeat Steps (1) and (2) so each vertex of K_n serves as the starting vertex.
    4. Choose the cycle of least total weight. Rewrite it with the desired reference point.

  - ✓ *Output*: hamiltonian cycle.
- Example 2.13 Apply the Repetitive Nearest Neighbor Algorithm to the graph from Example 2.11.

The Traveling Salesman Problem

- **Cheapest Link Algorithm**
  - ✓ *Input*: Weighted complete graph K_n.
  - ✓ *Steps*:

    1. Among all edges in the graph pick the one with the smallest weight. If two possible choices have the same weight, you may randomly pick one. Highlight the edge.

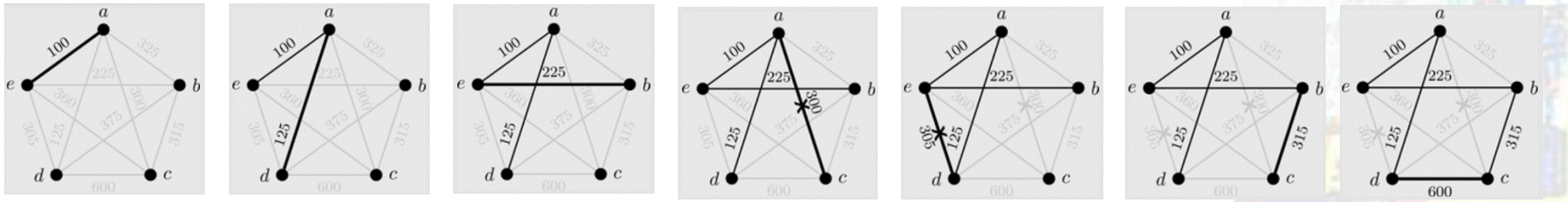
    2. Repeat Step (1) with the added conditions:

    (*a*) no vertex has three highlighted edges incident to it; and

    (*b*) no edge is chosen so that a cycle closes before hitting all the vertices.

    3. Calculate the total weight.

  - ✓ *Output*: hamiltonian cycle.
- Example 2.14 Apply the Cheapest Link Algorithm to the graph from Example 2.11.



The Traveling Salesman Problem

- **Nearest Insertion Algorithm**
  - ✓ *Input*: Weighted complete graph K_n.
  - ✓ *Steps*:

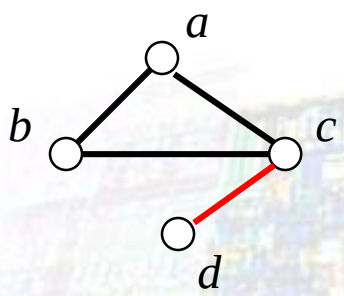
1. Among all edges in the graph, pick the one with the smallest weight. If two possible choices have the same weight, you may randomly pick one. Highlight the edge and its endpoints.

2. Pick a vertex that is closest to one of the two already chosen vertices. Highlight the new vertex and its edges to both of the previously chosen vertices.

3. Pick a vertex that is closest to one of the three already chosen vertices. Calculate the increase in weight obtained by adding two new edges and deleting a previously chosen edge. Choose the scenario with the smallest total. For example, if the cycle obtained from Step (2) was $a-b-c-a$ and d is the new vertex that is closest to c, we calculate:

$w(dc)+w(db)-w(cb)$ and $w(dc)+w(da)-w(ca)$

and choose the option that produces the smaller total.



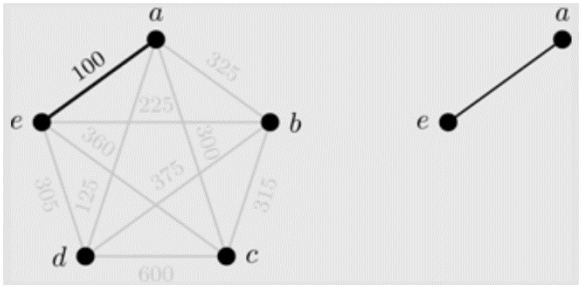
4. Repeat Step (3) until all vertices have been included in the cycle.

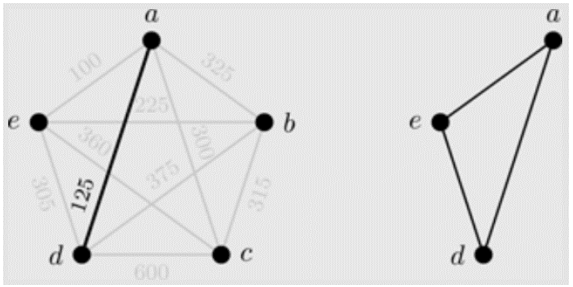
5. Calculate the total weight.

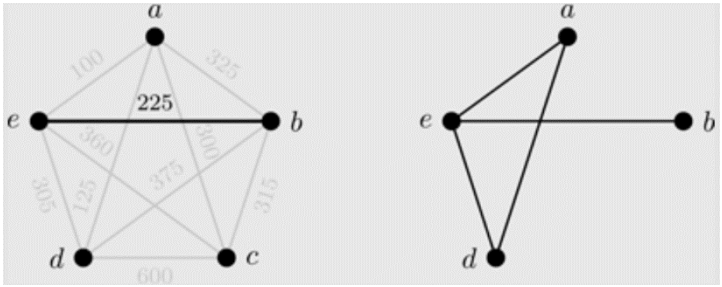
  - ✓ *Output*: hamiltonian cycle.

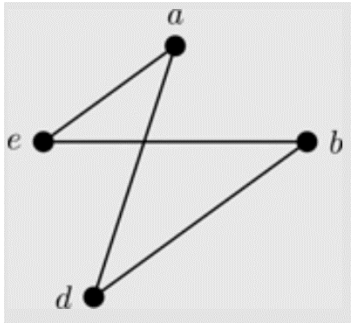
The Traveling Salesman Problem

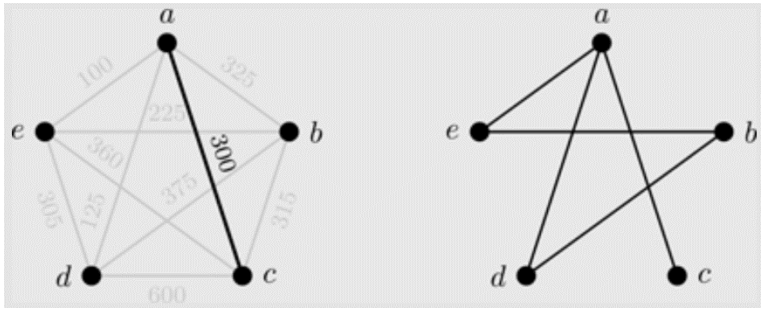
- Example 2.15 Apply the Nearest Insertion Algorithm to the graph from Example 2.11.

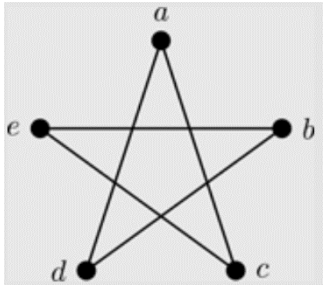






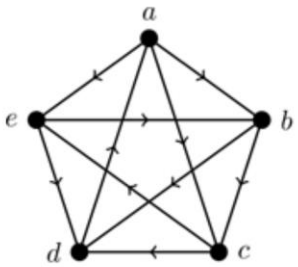




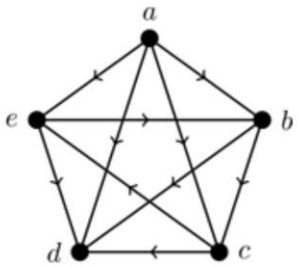


- **Definition 2.19** The *relative error* for a solution is given by ϵ_r=(*Solution*−*Optimal*)/*Optimal*
- Example 2.16 Find the relative error for each of the algorithms performed on the graph from Example 2.11.
  - ✓ Repetitive Nearest Neighbor: $\epsilon_r = (1275-1270)/1270=0.003937\approx 0.39\%$
  - ✓ Cheapest Link: $\epsilon_r = (1365-1270)/1270=0.074803\approx 7.48\%$
  - ✓ Nearest Insertion: $\epsilon_r = (1385-1270)/1270=0.090551\approx 9.05\%$

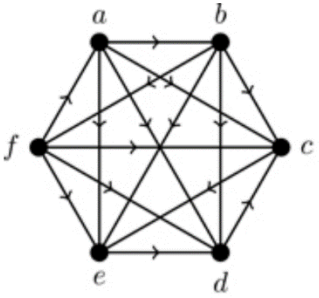
Tournament Revisited



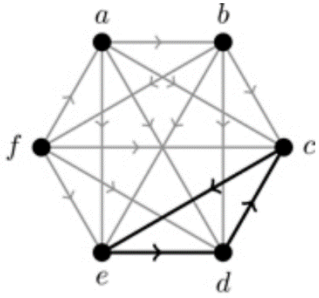
T_1



T_2

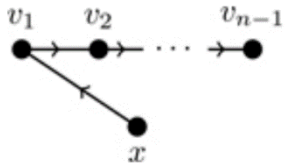


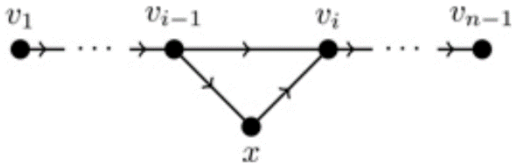
T_3



T_3 with subcycle

- **Theorem 2.20** Every tournament has a hamiltonian path.
  - ✓ prove by induction.

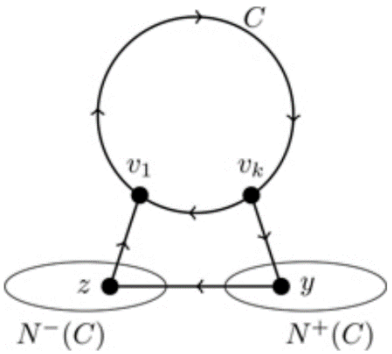




- **Definition 2.21** A digraph G is *strong* if for any pair of distinct vertices x and y there exists an $x-y$ path and a $y-x$ path in G.
- **Theorem 2.22** An increasing sequence $S: s_1, s_2, \ldots, s_n$ (for $n \geq 2$) of nonnegative integers is a score sequence of a strong tournament if and only if $s_1 + s_2 + \cdots + s_k > k(k-1)/2$ for each k with $1 \leq k \leq n-1$ and with equality holding at $k = n$.

Tournament Revisited

- **Theorem 2.23** A tournament is strong if and only if it is hamiltonian.
  - ✓ Hamiltonian → Strong
  - ✓ If T is strong, prove by contradiction. Assume C is the largest cycle.
  - ✓ For each vertex x not in C, if x has an arc to any one vertex in C, x has an arc to all vertices of C. Similar for the converse situation.
  - ✓ $N^+(C)=\{x\in V(T)|v_i\rightarrow x \text{ for all } 1\le i\le k\}$; $N^-(C)=\{x\in V(T)|x\rightarrow v_i \text{ for all } 1\le i\le k\}$
  - ✓ Note that $N^+(C)$ and $N^-(C)$ cannot both be empty.

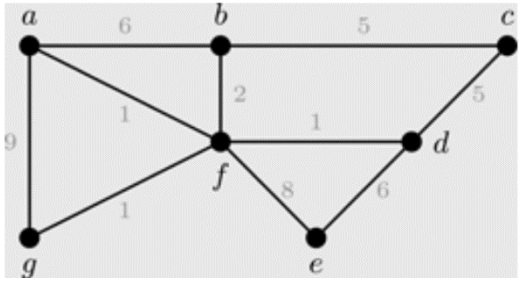


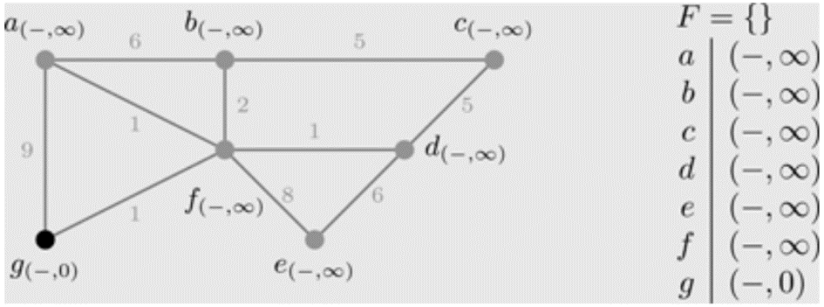
Shortest Paths

- **Dijkstra's algorithm**
  - ✓ *Input*: Weighted connected simple graph $G=(V,E,w)$ and designated *Start* vertex.
  - ✓ *Steps*:
  - ✓ (1) For each vertex x of G, assign a label $L(x)$ so that $L(x)=(-,0)$ if $x=Start$ and $L(x)=(-,\infty)$ otherwise. Highlight *Start*.
  - ✓ (2) Let $u=Start$ and define F to be the neighbors of u. Update the labels for each vertex v in F as follows: if $w(u)+w(uv)<w(v)$,then redefine $L(v)=(u,w(u)+w(uv))$ otherwise do not change $L(v)$
  - ✓ (3) Highlight the vertex with lowest weight as well as the edge uv used to update the label. Redefine $u=v$.
  - ✓ (4) Repeat Steps (2) and (3) until each vertex has been reached. In all future iterations, F consists of the un-highlighted neighbors of all previously highlighted vertices and the labels are updated only for those vertices that are adjacent to the last vertex that was highlighted.
  - ✓ (5) The shortest path from *Start* to any other vertex is found by tracing back using the first component of the labels. The total weight of the path is the weight given in the second component of the ending vertex.
  - ✓ *Output*: Highlighted path from *Start* to any vertex x of weight $w(x)$.

Shortest Paths

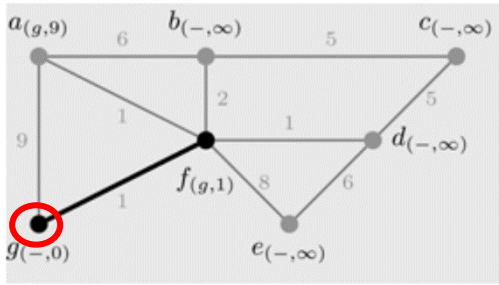
- **Example 2.17** Apply Dijkstra's Algorithm to the graph below where *Start*=g.





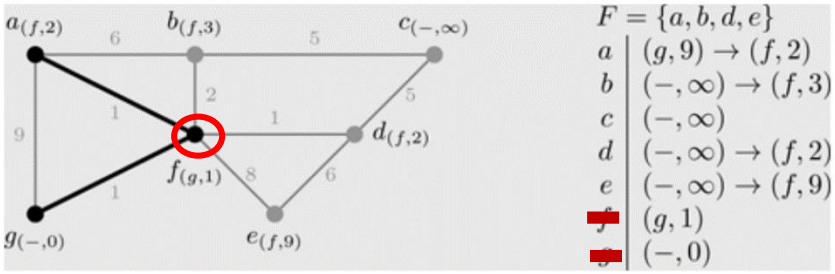
$F = \{\}$

a	$(-,\infty)$
b	$(-,\infty)$
c	$(-,\infty)$
d	$(-,\infty)$
e	$(-,\infty)$
f	$(-,\infty)$
g	$(-,0)$



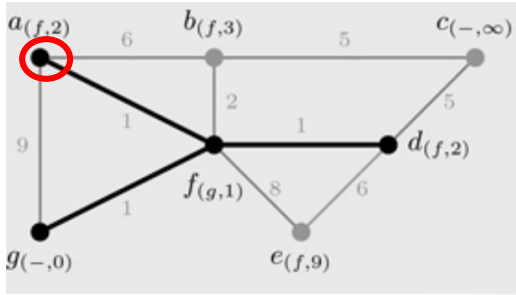
$F = \{a, f\}$

a	$(-,\infty) \to (g,9)$
b	$(-,\infty)$
c	$(-,\infty)$
d	$(-,\infty)$
e	$(-,\infty)$
f	$(-,\infty) \to (g,1)$
[redacted]	$(-,0)$



$F = \{a, b, d, e\}$

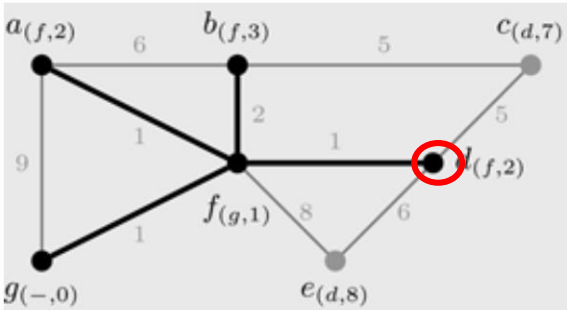
a	$(g,9) \to (f,2)$
b	$(-,\infty) \to (f,3)$
c	$(-,\infty)$
d	$(-,\infty) \to (f,2)$
e	$(-,\infty) \to (f,9)$
[redacted]	$(g,1)$
[redacted]	$(-,0)$



$F = \{b, d, e\}$

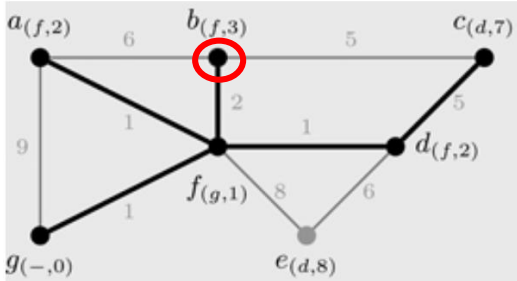
[redacted]	$(f,2)$
b	$(f,3)$
c	$(-,\infty)$
d	$(f,2)$
e	$(f,9)$
[redacted]	$(g,1)$
[redacted]	$(-,0)$

Shortest Paths



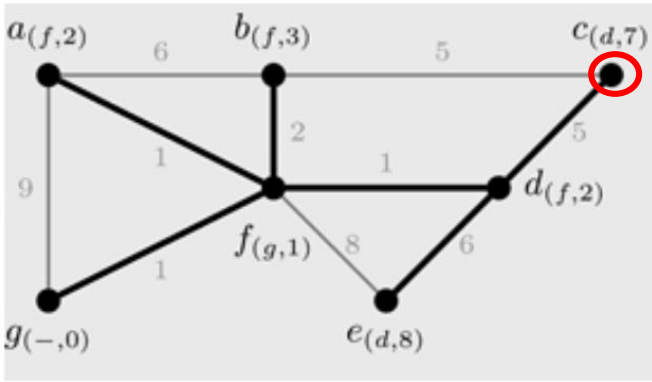
$F = \{b, c, e\}$

~~a~~	$(f, 2)$
b	$(f, 3)$
c	$(-, \infty) \to (d, 7)$
~~d~~	$(f, 2)$
e	$(f, 9) \to (d, 8)$
~~f~~	$(g, 1)$
~~g~~	$(-, 0)$



$F = \{c, e\}$

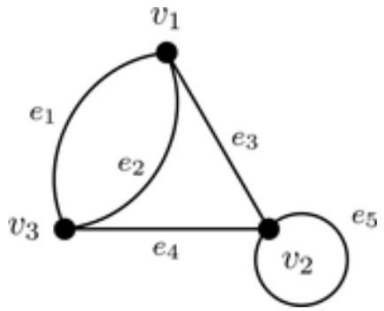
~~a~~	$(f, 2)$
~~b~~	$(f, 3)$
c	$(d, 7)$
~~d~~	$(f, 2)$
e	$(d, 8)$
~~f~~	$(g, 1)$
~~g~~	$(-, 0)$



$F = \{e\}$

~~a~~	$(f, 2)$
~~b~~	$(f, 3)$
~~c~~	$(d, 7)$
~~d~~	$(f, 2)$
e	$(d, 8)$
~~f~~	$(g, 1)$
~~g~~	$(-, 0)$

Walks Using Matrices



$$\begin{bmatrix} 0 & 1 & 2 \\ 1 & 1 & 1 \\ 2 & 1 & 0 \end{bmatrix}$$

- walk of length 2 from v_1 to v_3

$$v_1 \underset{e_3}{\rightarrow} v_2 \underset{e_4}{\rightarrow} v_3$$

- walk of length 2 from v_1 to v_2

$$v_1 \underset{e_3}{\rightarrow} v_2 \underset{e_5}{\rightarrow} v_2$$

$$v_1 \underset{e_1}{\rightarrow} v_3 \underset{e_4}{\rightarrow} v_2$$

$$v_1 \underset{e_2}{\rightarrow} v_3 \underset{e_4}{\rightarrow} v_2$$

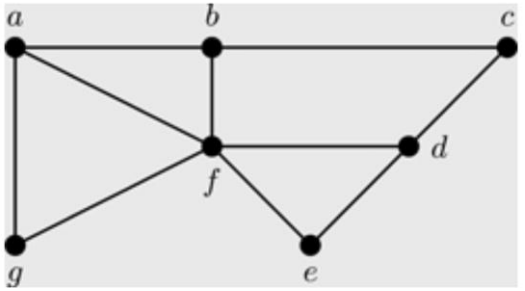
$$[0 \quad 1 \quad 2] \cdot \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = 0*1 + 1*1 + 2*1 = 3 \qquad A^2 = \begin{bmatrix} 5 & 3 & 1 \\ 3 & 3 & 3 \\ 1 & 3 & 5 \end{bmatrix}$$

- **Theorem 2.42** Let G be a graph with adjacency matrix A. Then for any integer $n>0$ the entry a_{ij} in A^n counts the number of walks from v_i to v_j.

Distance, Diameter, and Radius

- **Definition 2.25** Given two vertices x, y in a graph G, define the distance $d(x,y)$ as the length of the shortest path from x to y. The *eccentricity* of a vertex x is the maximum distance from x to any other vertex in G; that is $\epsilon(x)= \max_{y \in V(G)} d(x,y)$.The *diameter* of G is the maximum eccentricity among all vertices, and so measures the maximum distance between any two vertices; that is $diam(G)= \max_{x,y \in V(G)} d(x,y)$. The *radius* of a graph is the minimum eccentricity among all vertices; that is $rad(G)= \min_{x \in V(G)} \epsilon(x)$.

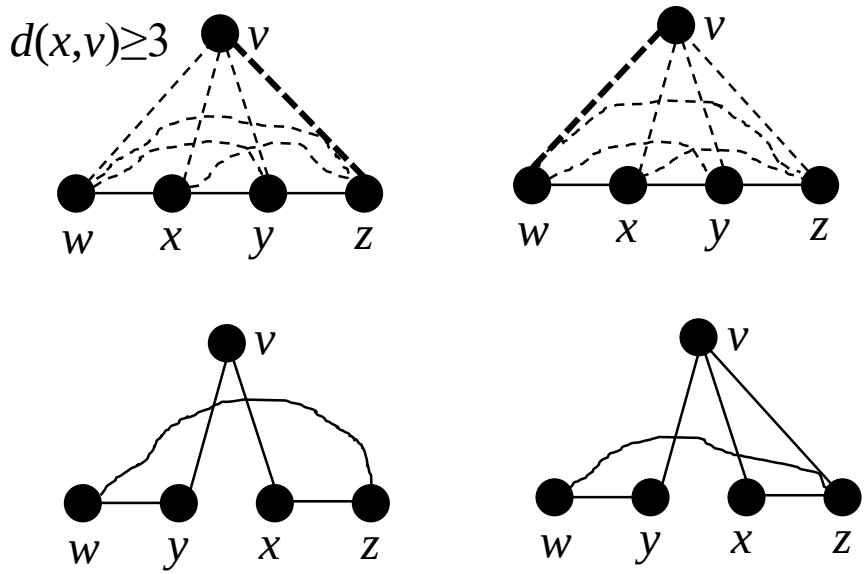
- **Example 2.18** Find the diameter and radius for the graph below.



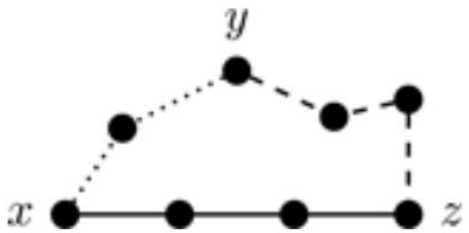
- **Theorem 2.26** If G is disconnected then $\bar{G}$ is connected and $diam(\bar{G}) \leq 2$.

Distance, Diameter, and Radius

- **Theorem 2.27** For a simple graph G if $rad(G) \geq 3$ then $rad(\bar{G}) \leq 2$.



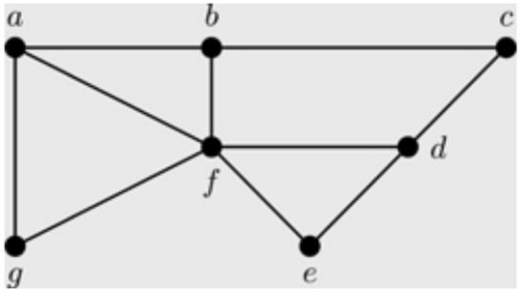
- **Theorem 2.28** For any simple graph G, $rad(G) \leq diam(G) \leq 2rad(G)$.
  - ✓ $d(x,z) = diam(G)$;
  - ✓ $\epsilon(y) = rad(G) = r$.

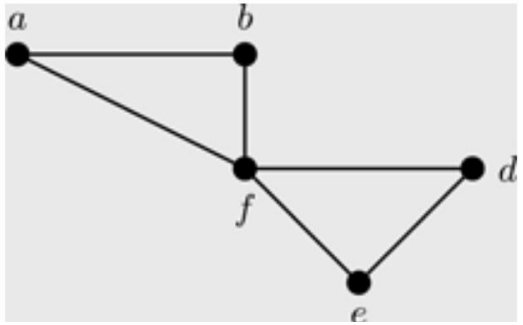


Shortest path from x to z

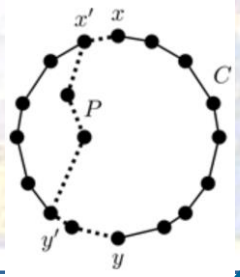
Distance, Diameter, and Radius

- **Definition 2.29** Let G be a graph with $rad(G)=r$. Then x is a *central vertex* if $\epsilon(x)=r$. Moreover, the center of G is the graph $C(G)$ that is induced by the central vertices of G.
- **Example 2.19** Find the center of the graph from Example 2.18.



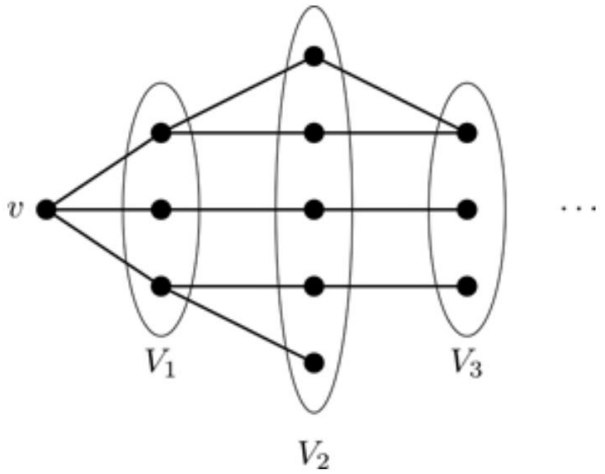


- **Definition 2.30** Given a graph G, the *girth* of G, denoted $g(G)$, is the minimum length of a cycle in G. The *circumference* of G is the maximum length of a cycle.
- **Example 2.20** Find the girth and circumference for the graph from Example 2.18.
- **Theorem 2.31** If G is a graph with at least one cycle then $g(G) \leq 2diam(G)+1$.
  - ✓ C is a minimum-length cycle of length $g(G)$
  - ✓ Prove by contradiction, assume $g(G) > 2diam(G)+1 \geq 2diam(G)+2$
  - ✓ Find x and y on C such that their distance along C is at least $diam(G)+1$



Distance, Diameter, and Radius

- **Theorem 2.32** Let G be a graph with n vertices, radius at most k, and maximum degree at most d, with $d \geq 3$. Then $n < \frac{d}{d-2}(d-1)^k$.



  - ✓ v is a central vertex with maximum degree at most d
  - ✓ Thus

$$\begin{aligned} |V_{i+1}| &\leq (d-1)|V_i| \\ &\leq (d-1)(d-1)|V_{i-1}| \\ &\vdots \\ &\leq \underbrace{(d-1)\cdots(d-1)}_{i\ times}|V_1| \\ &\leq (d-1)^i d \end{aligned}$$

$$\begin{aligned} n &= |V_0| + |V_1| + \cdots + |V_k| \\ &\leq 1 + d + d(d-1) + \cdots + d(d-1)^{k-1} \\ &\leq 1 + d\sum_{i=0}^{k-1}(d-1)^i \\ &\leq 1 + d\left(\tfrac{(d-1)^k - 1}{d-2}\right) \\ &\leq 1 + \tfrac{d}{d-2}((d-1)^k - 1) \\ &< \tfrac{d}{d-2}(d-1)^k. \end{aligned}$$