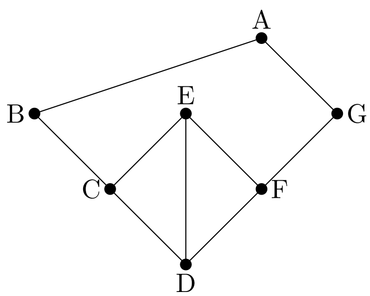
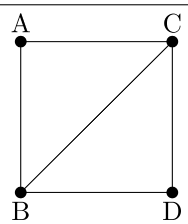
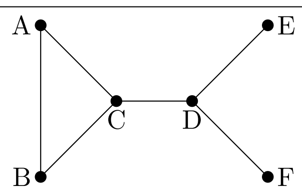
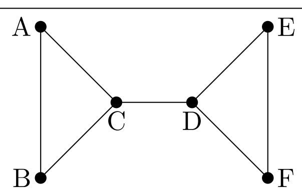
## 1. Hamiltonian Circuit

**Definition 1.1.**
A path that visits each vertex exactly once is a ________________ path.
A circuit that visits each vertex exactly once (except the beginning point will be visited again) is a ________________ circuit.

**Example 1.2.** Determine if the graph below have a Hamiltonian path or circuit.

| Graph | Hamiltonian Circuit? | Hamiltonian Path (but not circuit)? |
|---|---|---|
|  | | |
|  | | |
|  | | |
|  | | |

A ________________ can be used to count all possible Hamiltonian circuits.

**Algorithm 1.3** (Method of tree)**.**

(1) Choose a vertex $v$ and draw a point called $v'$.

(2) Find all vertices $w_1, w_2, \cdots, w_n$ which are neighbor to $v$. In other words, find all vertices in the edge containing $v$.

(3) Draw $n$ branches from $v'$, and call each new end of the branch as $w_i'$ for $i = 1, 2, \cdots, n$.

(4) For each $i = 1, 2, \cdots, n$

  (a) Find all vertices $x_1, \cdots, x_m$ which are neighbor to $w_i$, except $v$.

  (b) Draw $m$ branches from $w_i'$, and call each new end of the branch as $x_j'$ for $j = 1, 2, \cdots, m$.

(5) For each $j = 1, 2, \cdots, m$

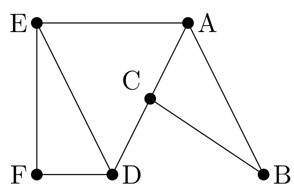
  (a) $\cdots$ (Repeat a process similar to (4) (a))

  (b) $\cdots$ (Repeat (4)(b))

(6) Repeat these process until every new branch has no neighbors unused for previous step.

**Do not try to remember the Algorithm! Instead, just be familiar with it by your hands from examples.**

**Example 1.4.** Use the method of trees to find all Hamiltonian circuits starting at A.



**Definition 1.5.** An ________________ Hamiltonian circuit is the circuit that has the least weight.

**Example 1.6.** From the weighed graph below, determine the minimum cost (travel time, in hours) to visit Miami, San Juan (Puerto Rico), Barbados and Belize City (Belize) if you must start and end in Miami. In what order do you visit the cities?

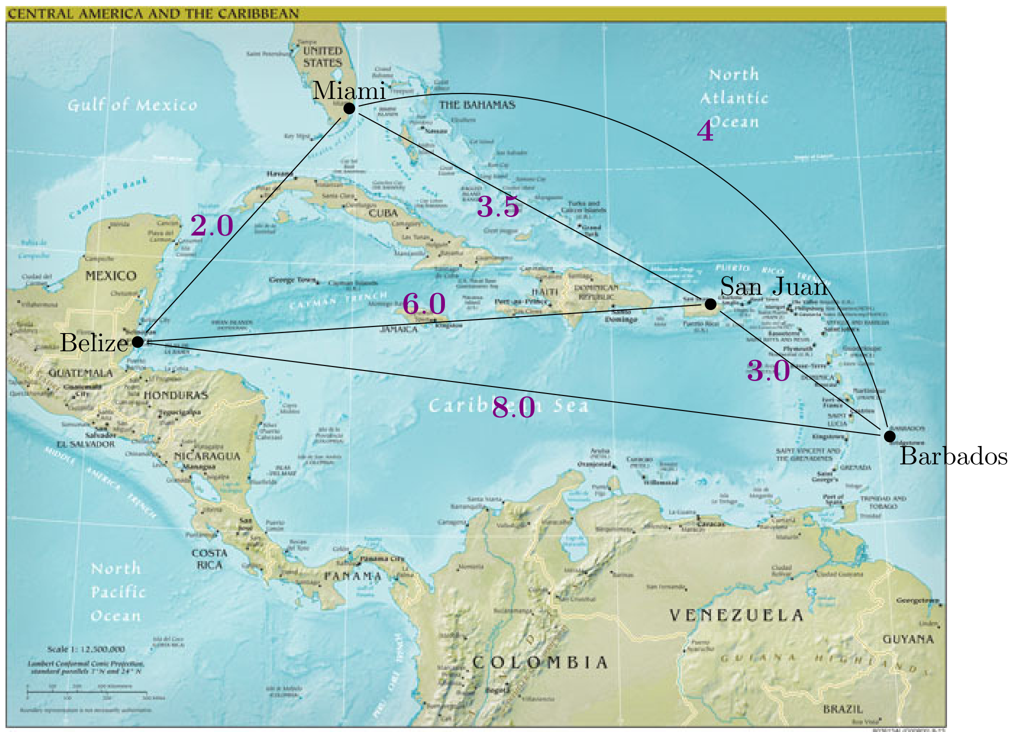
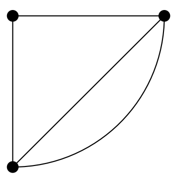
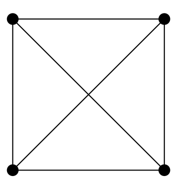
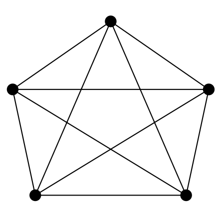
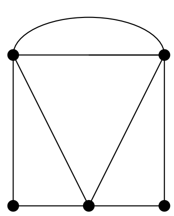



FIGURE 1. Physical Central America map, The World Factbook 2020. Washington, DC: Central Intelligence Agency, 2020.

**Definition 1.7.** A ________________graph is a graph in which every pair of vertices is connected by exactly one edge.

**Example 1.8.** Given the graphs below, figure out $V :=$ the number of vertices, $E :=$ the number of edges, and whether it is complete graph or not.

| Graph | $V$ | $E$ | Is it complete? |
|---|---|---|---|
|  | | | Y/N |
|  | | | Y/N |
|  | | | Y/N |
|  | | | Y/N |
|  | | | Y/N |

**Example 1.9.** Draw a complete graph with 3 vertices. Find all possible Hamiltonian circuits.

How many edges does a complete graph with $n$ vertices have? Can you explain why the formula holds?

**Theorem 1.10** (Fundamental Theorem of Counting)**.** *Suppose you have $k$ tasks to be performed. The first task can be completed $n_1$ ways, the second task $n_2$ ways, and $i$-th task can be completed in $n_i$ ways. The total number of ways that these $k$ tasks can be performed is*

$$\underline{\hspace{4cm}}$$

**Example 1.11.**

(1) How many outfits can be made from 2 shirts and 5 pairs of pants, if an outfit consists of a shirt and a pair of pants?

(2) How many different ways can 3 people stand in a line?

(3) How does this relate to the number of Hamiltonian circuits in a complete graph with 3 vertices?

Notation: $n! := n \times (n-1) \times \cdots \times 2 \times 1$. For example,

$$1! = 1 \quad 2! = 2 \times 1 = 2, \quad 3! = 3 \times 2 \times 1 = 6, \text{ and so on.}$$

**Theorem 1.12** (Number of Hamiltonian Circuits in a Complete Graph with $n$ vertices)**.**

- *There are __________________Hamiltonian circuits in a complete graph.*
- *There are __________________Hamiltonian circuits in a complete graph if a circuit and its mirror image are not counted as separates circuits.*
- *If a starting point is specified (fixed), then there are __________________Hamiltonian circuits in a complete graph.*
- *If a starting point is specified (fixed), then there are __________________Hamiltonian circuits in a complete graph if a circuit and its mirror image are not counted as separates circuits.*

**Example 1.13.**
For a complete graph with 6 vertices,

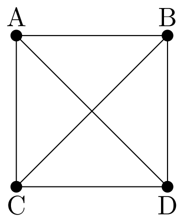
(1) How many Hamiltonian circuits are there?

(2) How many different non-mirror image Hamiltonian circuits are there?

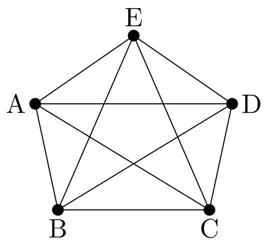
(3) How many Hamiltonian circuits are there starting at the vertex A?

(4) How many different non-mirror image Hamiltonian circuits are there starting at the vertex A?

**Example 1.14.** Use the brute force method to find all different (non-mirror image) Hamiltonian circuits for the complete graphs below starting at $A$.



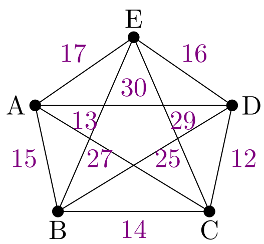
**Example 1.15.** Use the brute force method to find all different (non-mirror image) Hamiltonian circuits for the complete graphs below starting at $A$.



## 2. Traveling Salesman Problem

The **traveling salesman problem** is the problem of finding a least cost Hamiltonian circuit in a complete graph where each edge has been assigned a cost (or weight).

**Example 2.1.** Use the brute force method to find the least cost Hamiltonian circuit, starting at A, in the graph below. The values are distances between points, in miles.




Can you figure out the least cost Hamiltonian circuit without brute force? How can you figure out? Does your method work for other weighted complete graphs?

**Definition 2.2.** ________________is an algorithm which are faster to solve, however the result may not be an optimal.

The TSP is an important and common problem to solve, however since a company wants to solve it as fast as possible, so we need **heuristic algorithm**.

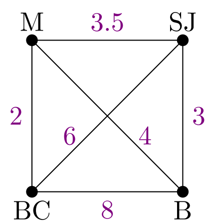
## 3. Helping Traveling Salesmen

**Algorithm 3.1** (Nearest Neighbor (NN) Algorithm (for finding low-cost Hamiltonian circuits))**.**

(1) Starting from the home city.
(2) Visit the nearest city first.
(3) Visit the nearest city that has not already been visited.
$\cdots$ Repeat until no other choices remain.
(4) Return to the home city when no other choices remain.

This is an example of ________________algorithm, which the choices are made by what is best at the next step.

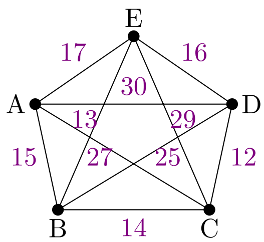
**Example 3.2.** Use the NN algorithm to solve the TSP of finding a low-cost trip starting at Miami (M) and traveling to the other three places. The weights given are the travel times between places, in hours.)



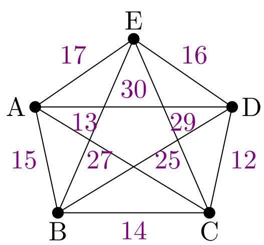

Is the solution you found the same as the optimal solution found when we used brute force?

**Example 3.3.** Use the NN algorithm to find a low-cost Hamiltonian circuit (solve the TSP) given the graph below, starting at D and also starting at A. The values are distances between points, in miles.

Starting at D:




Starting at A:




Is the solution you found starting at A the same as the optimal solution found when we used brute force?

**Example 3.4.** The following table gives the distances between the different cities, in miles.

| | Afton | Bar Nunn | Casper | Laramie | Newcastle | Pine Bluffs | Rock Springs | Sleepy Hollow |
|---|---|---|---|---|---|---|---|---|
| Afton | 0 | 355 | 356 | 387 | 537 | 477 | 180 | 477 |
| Bar Nunn | 355 | 0 | 7 | 146 | 189 | 227 | 228 | 122 |
| Casper | 356 | 7 | 0 | 142 | 185 | 223 | 228 | 128 |
| Laramie | 387 | 146 | 142 | 0 | 246 | 89 | 207 | 249 |
| Newcastle | 537 | 189 | 185 | 246 | 0 | 217 | 409 | 78 |
| Pine Bluffs | 477 | 227 | 223 | 89 | 217 | 0 | 296 | 285 |
| Rock Springs | 180 | 228 | 228 | 207 | 409 | 296 | 0 | 351 |
| Sleepy Hollow | 477 | 122 | 128 | 249 | 78 | 285 | 351 | 0 |

If the brute force method was used to find the least cost Hamiltonian circuit (solve the TSP) starting at Afton, how many different (non-mirror image) circuits would need to be checked?

Use the NN algorithm to solve the TSP, starting at Afton.

Use the NN algorithm to solve the TSP, starting at Bar Nunn.

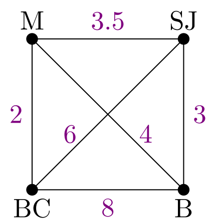
**Algorithm 3.5** (Sorted Edges (SE) Algorithm (for finding low-cost Hamiltonian circuits))**.**

(1) Arrange edges of the complete graph in order of increasing cost

(2) Select the lowest cost edge that has not already been selected that
- Does not cause a vertex to have 3 edges
- Does not close the circuit unless all vertices have been included.

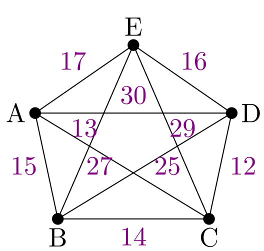
(3) Repeat (2) until we have no possible edges to select.

**Example 3.6.** Solve the following TSP using the SE algorithm. (Remember this graph represents places and the travel times between them, in hours.)




Is the solution you found the same as the optimal solution found when we used brute force?

**Example 3.7.** Solve the following TSP using the SE algorithm. (Remember the weights given are distances, in miles.)




Is the solution you found the same as the optimal solution found when we used brute force?

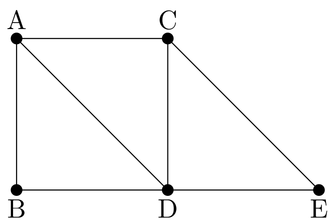
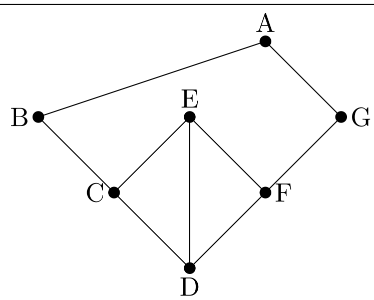
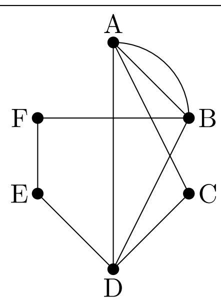
## 4. Minimum Cost Spanning Tree

**Definition 4.1.** A connected graph that has no circuits is a ________________. A ________________is a tree that has all the vertices of the original graph.

**Algorithm 4.2** (Creat a spanning tree of the given graph)**.**

(1) Copy the vertices with no edges
(2) Add edges back one by one until you have a connected graph that uses all vertices and contains no circuits

**Example 4.3.** For the following graphs, form a subgraph that is a spanning tree.

<table>
<tr><th>Graph</th><th>Spanning Tree</th></tr>
<tr><td>
</td><td></td></tr>
<tr><td>
</td><td></td></tr>
<tr><td>
</td><td></td></tr>
</table>

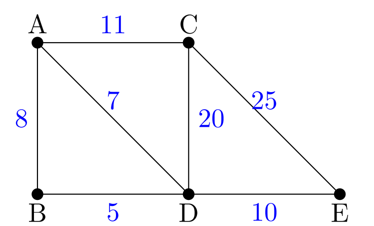
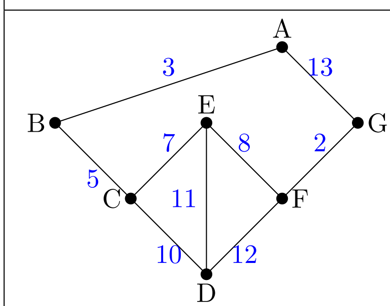
**Definition 4.4.**
A ________________ spanning tree is a spanning tree with the smallest possible weight.

To find a minimal spanning tree of the graph, we use the ________________ algorithm

**Algorithm 4.5** (Kruskal's algorithm)**.**
Suppose that the given graph has $n$ vertices.

(1) Let $M = \{\}$ be an empty set.
(2) Pick the edge with the smallest weight, say $e_0$. Put $e_0$ in $M$; now $M = \{e_0\}$.
(3) While # of elements in $M < n - 1$,
  (a) Pick the edges with the smallest weight among all edges outside of $M$, say $e$.
  (b) Check $M \cup \{e\}$ as a graph has a cycle;
    (i) If $M \cup \{e\}$ has a cycle, then delete $e$ completely from the edges, and go back to (3)
    (ii) Otherwise, put $e$ in $M$, and go back to (3).
(4) Now $M$ has $n - 1$ edges, which forms a minimal spanning tree.

**Example 4.6.** Use Kruskal's algorithm to find a minimum-cost spanning tree from the graph below (in each, the weights given are times, in minutes):

| Graph | Spanning Tree | Total cost of the minimum spanning tree |
|---|---|---|
|   | | |
|   | | |

## 5. Critical Path Analysis

**Definition 5.1.**
________________graph or ________________is a list of vertices connected by arrows. If the tasks cannot be completed in a random order, then the order can be specified in an ________________digraph. If the time to complete task is shown on the digraph, it is a ________________digraph.

**Example 5.2.** Suppose that the first task $T_1$ takes 4 minutes and a second task $T_2$ takes 6 minutes. Assume that $T_2$ cannot be started until $T_1$ is done. This would be represented in a weighted digraph as

$$\underbrace{4}_{T_1} \to \underbrace{6}_{T_2}$$

**Definition 5.3.** An ________________task is one that can be done independently of any of the other tasks.

So if task $T_3$ takes 1 minutes and is an independent task, the weighed order-requirement digraph is as below.

$$\underbrace{4}_{T_1} \to \underbrace{6}_{T_2}$$

$$\underbrace{1}_{T_3}$$

**Example 5.4.** Break down the fajita recipe below into a series of tasks. Show these tasks in a weighted order-requirement digraph.

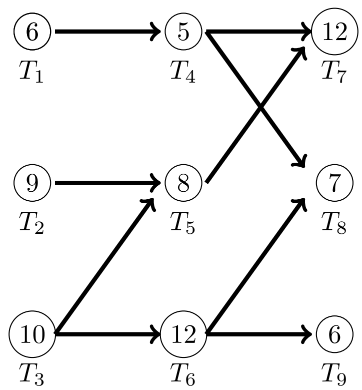
Ingredients: Tortillas, Chicken, 1 onion, Seasoning, Tomatoes, and Cilantro.

(1) Slice the onion and chicken into strips. It will take 2 minutes and 5 minutes respectively.
(2) Cook chicken 10 minutes then add onions and cook for 5 more minutes.
(3) Add seasoning (1 minute)
(4) Chop tomatoes (2 minutes) and mix with Cilantro. (1 minute)
(5) Warm tortillas (1 minute)
(6) Serve it (5 minutes)

**Definition 5.5.**
A __________________on the digraph is the *longest path* and it determines the *earliest completion time* (the earliest possible time for the completion of all of the tasks making up the job in the digraph.)

**Example 5.6.** What is the critical path for making fajitas?

**Example 5.7.** Determine the critical path in the digraph below.




What is the earliest possible time for completion of all the tasks in the digraph?