**Chapter 2 B*usiness Efficiency-part I***

Concepts in these notes:

• Hamiltonian circuit

• Weighted graph

• Complete graph

• Algorithm

• Method of tree diagram

• Fundamental Principle of counting

• Nearest-Neighbor (Greedy) algorithm

• Sorted-edges algorithm

1. **Traveling Salesman Problem (TSP) —**

Determine the trip of minimum cost that a salesperson can make to visit the cities in a sales territory, starting and ending the trip in the same city.

Example 1. A college student from Chicago wants to visit friends on her spring break. She will need to drive to Minneapolis, Cleveland, and St. Louis then back home to Chicago. The number of miles between each city is labeled.

The problem is how to **visit each vertex exactly one time with minimum distance traveled.**

figure-02-03.JPG                                               000264EE
production                     B8414D3D:

A graph with every edge labeled with a weight is called a **weighted graph**, where weight may represent distance, time, cost, etc.

The traveling salesman problem has many applications:

• Saturday morning activities.

(a) mail a package at post office

(b) pick up dry cleaning

(c) check out some books at county library

(d) go to a car wash

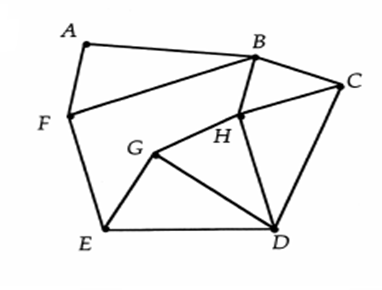
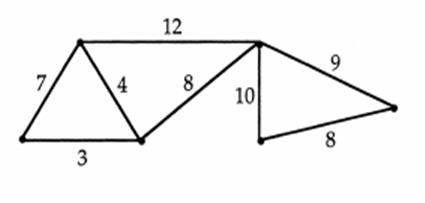
• Collecting mail from each of the large blue collection boxes on streets in an area.

• Picking up 6 people from their home to your home.

• Visiting three countries in a business trip.

A **Hamiltonian circuit** is a circuit that visits each vertex on the graph once and only once. (It starts and ends at the same vertex.)

Example 3. Find a Hamiltonian circuit in each graph, starting from Vertex A then find different Hamilton circuit starting from vertex C.

**Comment:**  There is no simple criterion (no Euler-like Theorem) to tell whether or not a graph has a Hamiltonian circuit.

In some special situations, where graphs are known to have Hamiltonian circuits, people developed several algorithms to find optimal or near optimal solution.

**Optimal solution**: Identify a Hamiltonian circuit in the graph which has the minimum distance, cost or time.

**Near optimal solution**: Identify a Hamiltonian circuit in the graph which does not has the minimum distance/cost, however, its distance/cost is not off the minimum much.

Why should we use a near optimal solution?

(a) It is usually much easier to find.

(b) Finding a solution also costs time and/or money.

(c) Finding an optimal solution might cost much more than finding a near optimal solution.

1. **An Algorithm Using Tree Diagram**

An **algorithm** is a step-by-step description of how to solve a problem.

The following is an algorithm to determine which **Hamiltonian circuit** has minimum cost that is to find an optimal solution.

**Step 1.** Generate and list all possible Hamiltonian circuits.

**Step 2**. Add up the weights (distances, costs) on the edges of each circuit.

**Step 3**. Choose the Hamiltonian circuit of minimum weight (distance, cost).

Example 4. Use a tree diagram to list all Hamiltonian circuits in the following diagram, and then find the optimal solution.

figure-02-03.JPG                                               000264EE
production                     B8414D3D:

• To work on a tree diagram, a graph should be complete. That is, every pair of vertices is joined by an edge.

Example 5: If the number of cities is not 4, but 5 as the following graph, how many “branch tips”? (That is number of all possible Hamiltonian circuits in the graph.)



If the number of cities is not 4, but *n*, how many “branch tips”?

**The fundamental principle of counting**

**— The multiplication rule:**

If there are *a* ways of choosing one thing, b ways of choosing a second thing after the first is chosen, ..., and z ways to choosing the last thing, then the total number of choices is *a × b × … × z.*

Example 6. (a) If you have 2 hats and 4 pairs of shoes, how many different ways to wear hat-shoes outfits?

(b) If you have 2 hats, 3 shirts and 6 pairs of shoes, how many different ways to wear hat-shirt-shoes outfits?

Example 7. How many ways to arrange your 5 Saturday morning activities?

Example 5. (Continue) If the number of cities is not 4, but *n*, How many ways to arrange the travel?

(n − 1)! = (n − 1) x (n - 2) × …. × 1

“!” is read as “factorial”. For instance,

(4 − 1)! = 3! = 3 × 2 × 1 = 6

(5 − 1)! = 4! = 4 × 3 × 2 × 1 = 24

(6 − 1)! = 5! = 5 × 4 × 3 × 2 × 1 = 120

(9 − 1)! = 8! = 8 × 7 × 6 × · · · × 1 = 40, 320

(11 − 1)! = 10! = 10 × 9 × 8 × · · · × 1 = 3, 628, 800

• The **method of trees** is a visual method of carrying out the counting and listing all Hamiltonian circuits in a tree diagram.

**3. Nearest-neighbor (Greedy) Algorithm**

The nearest-neighbor algorithm is an approach that we always choose the nearest unvisited vertex as the next stop.

1. Start at a specified vertex (home city). Travel to the vertex (another city) that you haven’t been to yet whose path has the smallest weight (If there is a tie, pick one at random.)

3. Continue until you travel to all vertices (cities except home).

4. Travel back to your starting vertex (home city). The resulting path is a Hamilton Circuit.

• It is called the nearest-neighbor algorithm when the weights are distances.

• It is called a **greedy algorithm** when the best (cheapest) paths are chosen.

Example 8. Find the Hamiltonian circuit and the corresponding distance traveled in the following graph using nearest-neighbor algorithm. Is it optimal?

figure-02-03.JPG                                               000264EE
production                     B8414D3D:

Example 9. Find the Hamiltonian circuit of the following graph using nearest neighbor algorithm, where weights are costs (in $100). Start from A. Also find the total cost.

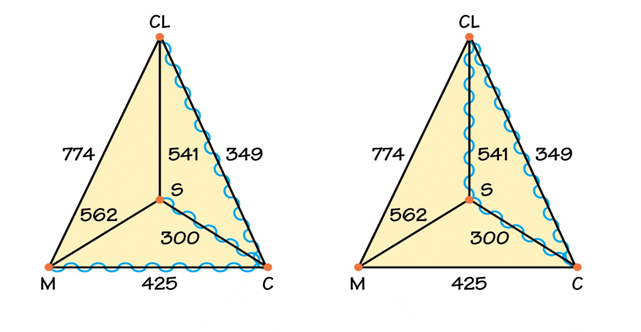


**4. Sort-Edges Algorithm**

**Review**: (1) A Hamiltonian circuit uses exactly two edges meeting at a vertex.

(2) A Hamiltonian circuit does not miss any vertex.

For instance, in the each of the following two graphs, the wiggled part can NOT lead to a Hamiltonian circuit. Why?



**Definition**: **The sort-edges algorithm** is an approach to find a Hamiltonian circuit with the following steps.

**Step 1**. List the edges of a complete graph in order of increasing weights, from the smallest weight to the largest weight.

**Step 2**. Start from the cheapest edge. Keep on adding an unused edge with least weight until a circuit is complete under the requirements that:

(a) no three edges meet at a same vertex,

(b) never close up a circuit without including all vertices.

Example 10. Find the Hamiltonian circuit and the corresponding distance traveled in the following graph using sort-edges algorithm. Start from Chicago. Is it optimal?

figure-02-03.JPG                                               000264EE
production                     B8414D3D:

Example 11. The following graph has weights that represent costs(in $100)

(a) Use sort-edges algorithm to find the Hamiltonian circuit



(b) Find the route(s) following the Hamiltonian circuit in (a)

(i) Start from A (ii) Start from D.

Comments:

(a) The Hamiltonian circuit of a graph determined by sort-edges algorithm does not depend on the starting vertex, but the tours do.

(b) The Hamiltonian circuit of a graph determined by nearest- neighbor algorithm depends on the starting vertex.

• Both nearest-neighbor algorithm and sort-edges algorithm are classified as Heuristic algorithm

• A heuristic algorithm is a method of solving an optimization problem “fast”, but that does not guarantee an optimal solution to the problem.

**Chapter 2 *Spanning Tree Problem -part II***

Topics in this section:

• Spanning tree

• Minimum-cost Spanning tree.

• Kruskal’s Algorithm

In a traveling salesman problem, it costs money every time the salesman travels to another city, and he has to go back to his home city at the end.

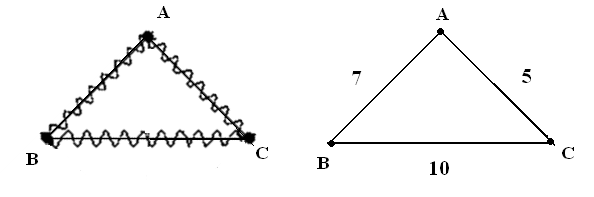
If establishing the connections between cities is **one-time investment** such as

• install power line when building an apartment complex;

• install TV cable in a new developing area;

• link 5 cities with septic service;

having a circuit link is a **big** waste. For instance, in the following 3- city graph, if AB and AC are linked, BC is linked through AB-BC.



The goal is to link all the cities with minimum cost.

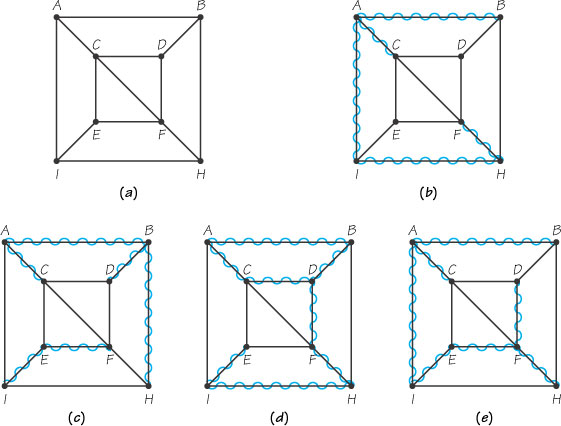
**Definitions:**

• A **subgraph** is a part of the original graph.

• A **tree** is a subgraph which is connected and has no circuit.

• A **spanning tree** is a tree which contains all the vertices in the original graph.

Example 1. Identify the wiggled part of each graph as a subgraph, or a tree, or a spanning tree.



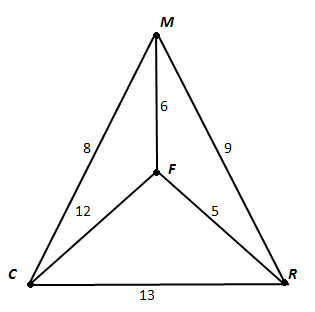
**Definition**: A minimum-cost spanning tree is a spanning tree with minimum sum of edge weights, which is the cost.

**Definition**: The Kruskal’s algorithm is an approach to find a minimum-cost spanning tree with the following steps.

**Step 1**. List the edges of a weighted graph in order of increasing weight.

**Step 2**. Start from the cheapest edge. Keep on adding an unused edge with least weight until all vertices are linked without a circuit.

Example 2. Find the minimum-cost spanning tree using Kruskal’s algorithm. The weights are costs in $100.

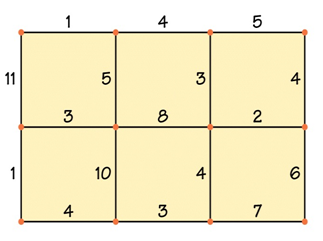


• **Kruskal’s algorithm** is somewhat similar to sort-edges algorithm, but with the different criteria about what to avoid.

• **Kruskal’s algorithm** always gives an optimal solution while sort- edges may not.

Example 3. Find the minimum-cost spanning tree for the each of the following graphs using Kruskal’s algorithm. The weights are costs in $1000. Also find the total cost.

figure-02-12.JPG                                               000264EE
production                     B8414D3D:



We have studied three different problems:

**(a) Chinese postman problem**

Find a circuit on the graph that covers every edge of the graph at least once and has the shortest possible length (finding an Euler circuit or eulerizing a graph).

**(b) Traveling salesman problem**

Find a minimum cost Hamiltonian circuit in a complete and weighted graph (every pair of vertices is joined by an edge and every edge has been assigned a weight).

Tree diagram will help to list all possible Hamiltonian circuits for small n. Then we may find the optimal solution by finding the minimum sum of costs. This is also called a **brute force** method.

Nearest-neighbor algorithm and sort-edges algorithm provide two approaches to find a Hamiltonian circuit, but that might not be an optimal (minimum cost) solution.

**(c) Minimum-cost spanning tree**

Find a spanning tree (all vertices are linked without any circuit) with minimum sum of costs.

Kruskal’s algorithm provides an approach to find the optimal (minimum cost) solution.

**Chapter 2 Order-requirement Digraph*-part III***

Topics: • Scheduling problems

• Order-requirement digraph

• Sum of task times

• Critical path

Example 1. You and your family have a kitchen remodeling project. You wrote down all the tasks in your mind: (a) clean the kitchen (f) install wallpaper on walls (b) replace old floor with new floor tiles (g) scrape paint on ceiling (c) install new stove (h) paint ceiling (d) scrape walls to remove old paint (i) install new sink (e) prime walls

However, you can not do these things in a random order, not Hamiltonian circuit, not spanning tree.

**Definitions**: **An order-requirement digraph** shows which tasks precede other tasks among the collection of tasks making up a job. (A graph to be given in class)

Example 2. An airplane must have its passengers and freight unloaded and new passengers and cargo loaded before it can take off again. The airline has the regulation that the cargo should be unloaded and cabin should be cleaned before loading new passengers. Thus the job turning the plane around requires completion of five tasks:

Task A Unload passengers 13 minutes

Task B Unload cargo 25 minutes

Task C Clean cabin 15 minutes

Task D Load new cargo 22 minutes

Task E Load new passengers 27 minutes

Construct an order-requirement digraph.

figure-02-17.JPG                                               000264EE
production                     B8414D3D:

**Definition**:

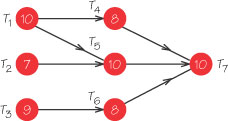
A **critical path** in an order-requirement digraph is a longest path. The length is measured in terms of summing the task times of the tasks making up the path.

The time a critical path takes is the least/fastest time in which all tasks can be completed.

Example 2. (Continue)

Find the critical path and the least time to complete all tasks. How do you apply it in managing an airline?

Example 3. Find the critical path in the following order-requirement digraph. Also find the least time to complete all tasks.



Example 1. (Continue) You and your family have a kitchen remodeling project. You wrote down all the tasks and reasonably estimated the time for each task.

Task: Time:

(a) clean the kitchen 4 Hr (f) install wallpaper on walls 5 Hr

(b) replace old floor with new floor tiles 7 Hr (g) scrape paint on ceiling 6 Hr

(c) install new stove 2 Hr (h) paint ceiling 4 Hr

(d) scrape walls to remove old paint 8 Hr (i) install new sink 5 Hr

(e) prime walls 6 Hr

Construct a reasonable order-requirement digraph. Find the critical path. What is the fastest time in which these tasks can be completed so you can enjoy your new kitchen?