COSMO Lecture 0
System Modelling and Optimization Lecture 1

Function, Inequality, and Graph

Instructor: Hoi-To Wai, Dept of SEEM, CUHK

June 22, 2024

interactive demo:

What is a Function?

What is a Function?

$$f : X \to Y$$
$$x \mapsto f(x)$$

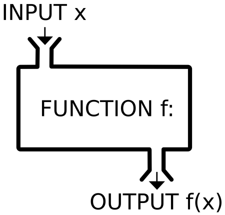
- A **mapping** from set X to Y[1] such that $x \in X$, $f(x) \in Y$.
- X(domain) = accepted input; Y(codomain) = possible output.
- This lecture will take $X \equiv \mathbb{R}$, $Y \equiv \mathbb{R}$ for most of the time.

[1] A **set** is a collection of unique **elements**.

What is a Function?

$$f: X \to Y$$
$$x \mapsto f(x)$$

- In words, a function is a *machine / black box* that relates input to its corresponding output.



(Image credits: Wikipedia)

Examples of Function (Abstract Form)

- **Example**: a *linear (affine) function*

$$f(x) = 10 + 20x$$

- **Example**: a *polynomial function*

$$f(x) = x + 2x^2 + 3x^3 + 4x^4$$

- **Example**: an *exponential function*

$$f(x) = 10^x$$

- **Example**: an *piecewise function*

$$f(x) = \begin{cases} |x| - \frac{1}{2}, & |x| \geq 1, \\ \frac{1}{2}x^2, & |x| \leq 1. \end{cases}$$

What is a Function? Why is it relevant?

- A *function* describes a **relationship** between input & output.
- **(Warm-up) Example:** It costs \$10 to buy a pen at a bookstore. What will be the total cost if I wish to buy x pen?

What is a Function? Why is it relevant?

- A *function* describes a **relationship** between input & output.

- **(Warm-up) Example:** It costs \$10 to buy a pen at a bookstore. What will be the total cost if I wish to buy x pen?

  ▷ buying 1 pen = \$10, 2 pen = \$20, 3 pen = \$30, ...

  ▷ the total cost is a **function** of x, with the **relationship**:

$$f(x) = 10 \cdot x.$$

  ▷ The above function is also a **linear function**

  — since it satisfies $f(\alpha x + \beta z) = \alpha f(x) + \beta f(z)$ for any x, y.

What is a Function? Why is it relevant?

- **Example:** In Hong Kong, the salary tax is calculated with a *progressive* rate.
- Suppose that \$$x$ is your yearly income.
- Let $f(x)$ be the tax you pay.

	應課税入息實額	税率	税款
	\$		\$
首	50,000	2%	1,000
另	50,000	6%	3,000
	100,000		4,000
另	50,000	10%	5,000
	150,000		9,000
另	50,000	14%	7,000
	200,000		16,000
餘額		17%	

What is a Function? Why is it relevant?

- **Example:** In Hong Kong, the salary tax is calculated with a *progressive* rate.
- Suppose that $\$x$ is your yearly income.
- Let $f(x)$ be the tax you pay.

	應課稅入息實額	稅率	稅款
	$		$
首	50,000	2%	1,000
另	50,000	6%	3,000
	100,000		4,000
另	50,000	10%	5,000
	150,000		9,000
另	50,000	14%	7,000
	200,000		16,000
餘額		17%	

$$f(x) = \begin{cases} 0.02x, & 0 \le x \le 50000, \\ 1000 + 0.06(x - 50000), & 50000 \le x \le 100000 \\ 4000 + 0.1(x - 100000), & 100000 \le x \le 150000 \\ 9000 + 0.14(x - 150000), & 150000 \le x \le 200000 \\ 16000 + 0.17(x - 200000), & 200000 \le x. \end{cases}$$

What is a Function? Why is it relevant?

- A (1-dimensional) *function* can be visualized by **graph/plot**.
- **Example:** when $f(x) = 10x$

What is a Function? Why is it relevant?

- **Example:** $f(x) =$ tax you pay,

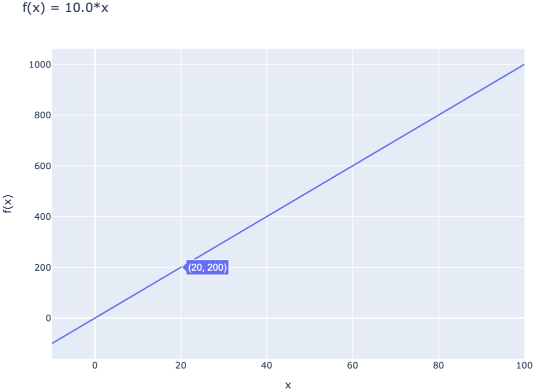
$$f(x) = \begin{cases} 0.02x, & x \leq 50000, \\ 1000 + 0.06(x - 50000), & 50000 \leq x \leq 100000 \\ 4000 + 0.1(x - 100000), & 100000 \leq x \leq 150000 \\ 9000 + 0.14(x - 150000), & 150000 \leq x \leq 200000 \\ 16000 + 0.17(x - 200000), & 200000 \leq x. \end{cases}$$

What is a Function? Why is it relevant?

- A (1-dimensional) *function* can be visualized by **graph/plot**.
- Plotting (complicated) function can be extremely tedious $\rightarrow$ refer to `interactive demo`.

What is a Function? Why is it relevant?

- Expressing relationship as a function (and graph it) enables us to **extrapolate** & make **decision** with ease.



- Also important for data visualization.

Nonlinear Function

- Function can be *nonlinear*, e.g.,

$$f(x) = 10x^2$$

$$f(x) = \sqrt{x}$$

(it can be checked that $f(\alpha x + \beta y) \neq \alpha f(x) + \beta f(y)$ for some x, y.)

- **Example:** area of a circle with radius x is

$$f(x) = \pi x^2$$

Function of Function (of Function ...)

- Function can be **composed** with another function or itself. Let $f_1 : \mathbb{R} \to \mathbb{R}$ and $f_2 : \mathbb{R} \to \mathbb{R}$, we have

$$f(x) = f_1(f_2(x))$$

  $\triangleright$ First evaluate $f_2(x)$, then take $f_2(x)$ as the input to $f_1(\cdot)$.

- **Example:** Suppose that you earn \$$300 \times 250 \times \sqrt{t}$ per year if you spend t hours at work every working day. How much tax do you have to pay?

Function of Function (of Function ...)

- Function can be **composed** with another function or itself. Let $f_1 : \mathbb{R} \to \mathbb{R}$ and $f_2 : \mathbb{R} \to \mathbb{R}$, we have

$$f(x) = f_1(f_2(x))$$

  ▷ First evaluate $f_2(x)$, then take $f_2(x)$ as the input to $f_1(\cdot)$.

- **Example:** Suppose that you earn \$$300 \times 250 \times \sqrt{t}$ per year if you spend t hours at work every working day. How much tax do you have to pay?

  ▷ take $f_2(t) = 300 \times 250 \times \sqrt{t}$ which relates t to annual income, and take $f_1(x)$ as the function from slide 6,

$$t \text{ hrs/day} \xrightarrow{f_2(\cdot)} \$ \text{ made per year} \xrightarrow{f_1(\cdot)} \text{tax to pay}$$

$$f(t) = f_1(f_2(t))$$

Random Functions (Advanced)

- Functions can also be **random**.
- **Example:** Adam can make \$10 per hour on a **sunny** day, and \$20 per hour on a **not sunny** day. How much will he make if work for x hours *tomorrow*?

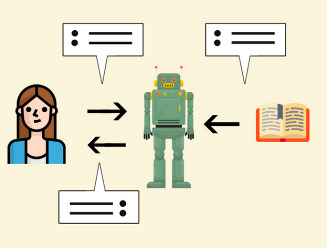
Random Functions (Advanced)

- Functions can also be **random**.
- **Example:** Adam can make \$10 per hour on a **sunny** day, and \$20 per hour on a **not sunny** day. How much will he make if work for x hours *tomorrow*?

$$f(x) = \begin{cases} 10x, & \text{if tomorrow is sunny,} \\ 20x, & \text{if tomorrow is not sunny.} \end{cases}$$

- Note the function $f(x)$ depends on the weather **tomorrow**, which is unknown and is random.
- Random (stochastic) functions are used in modeling decision.

Artificial Intelligence (Advanced)



Credits: `https://www.analyticsvidhya.com/blog/2022/11/comprehensive-guide-to-bert/`

- AI models such as `ChatGPT` does the following:

$$\text{(human prompt)} \xrightarrow{\texttt{ChatGPT}} \text{(human-like response)}$$

- It is essentially a **function** that maps *prompts* to *responses*[1].

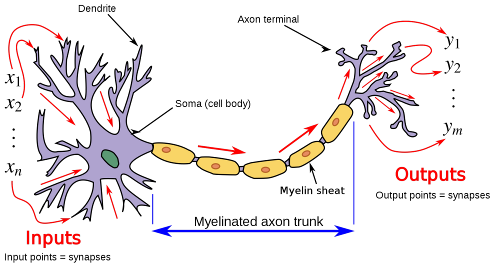
[1] They are *auto-regressive* model and *random* functions that are more complicated than anything we've seen so far, e.g., with *billions* of terms.

Artificial Intelligence (Advanced)

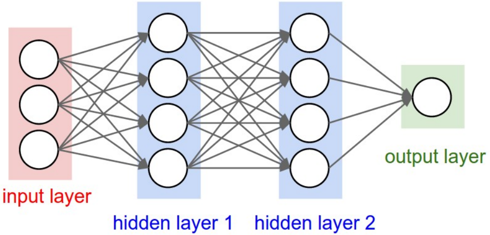
- What specifically is $f(\cdot)$ in `ChatGPT`?
- **Challenge 1**: prompts & responses $\neq$ 'numbers'

  $\triangleright$ encode into numbers, e.g., 'Yes' $\leftrightarrow$ 1, 'No' $\leftrightarrow$ 0, ...
- **Challenge 2**: The 'relationship' between *prompts* (as numbers) and *responses* (as numbers) is complicated

  $\triangleright$ needs complicated, i.e., 'expressive', functions $\rightarrow$ *artificial neural network* (ANN) that emulates brain with neurons.



Artificial Neural Networks

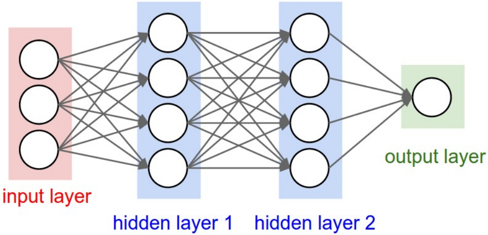


▷ With 3 inputs, x_1, x_2, x_3, the outputs at layer 1 are[2]

$$
\begin{aligned}
f_1^{(1)}(x_1, x_2, x_3) &= \max\{0, W_{11}^{(1)} x_1 + W_{12}^{(1)} x_2 + W_{13}^{(1)} x_3\} \\
f_2^{(1)}(x_1, x_2, x_3) &= \max\{0, W_{21}^{(1)} x_1 + W_{22}^{(1)} x_2 + W_{23}^{(1)} x_3\} \\
f_3^{(1)}(x_1, x_2, x_3) &= \max\{0, W_{31}^{(1)} x_1 + W_{32}^{(1)} x_2 + W_{33}^{(1)} x_3\} \\
f_4^{(1)}(x_1, x_2, x_3) &= \max\{0, W_{41}^{(1)} x_1 + W_{42}^{(1)} x_2 + W_{43}^{(1)} x_3\}
\end{aligned}
$$

[2] The actual ChatGPT is 10^9 times more complicated than this!

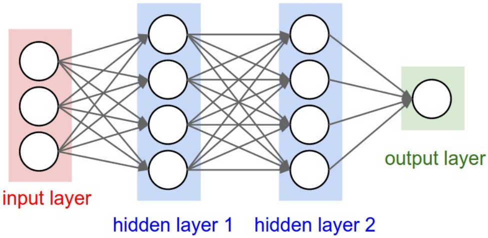
Artificial Neural Networks



- ▷ Outputs at layer 2 are

$$
\begin{aligned}
f_1^{(2)}(f_1^{(1)}, f_2^{(1)}, f_3^{(1)}, f_4^{(1)}) &= \max\{0, W_{11}^{(2)} f_1^{(1)} + W_{12}^{(2)} f_2^{(1)} + W_{13}^{(2)} f_3^{(1)} + W_{14}^{(2)} f_4^{(1)}\} \\
f_2^{(2)}(f_1^{(1)}, f_2^{(1)}, f_3^{(1)}, f_4^{(1)}) &= \max\{0, W_{21}^{(2)} f_1^{(1)} + W_{22}^{(2)} f_2^{(1)} + W_{23}^{(2)} f_3^{(1)} + W_{24}^{(2)} f_4^{(1)}\} \\
f_3^{(2)}(f_1^{(1)}, f_2^{(1)}, f_3^{(1)}, f_4^{(1)}) &= \max\{0, W_{31}^{(2)} f_1^{(1)} + W_{32}^{(2)} f_2^{(1)} + W_{33}^{(2)} f_3^{(1)} + W_{34}^{(2)} f_4^{(1)}\} \\
f_4^{(2)}(f_1^{(1)}, f_2^{(1)}, f_3^{(1)}, f_4^{(1)}) &= \max\{0, W_{41}^{(2)} f_1^{(1)} + W_{42}^{(2)} f_2^{(1)} + W_{43}^{(2)} f_3^{(1)} + W_{44}^{(2)} f_4^{(1)}\}
\end{aligned}
$$

Artificial Neural Networks



- ▷ Final output is:

$$f^{(3)}(f_1^{(2)}, f_2^{(2)}, f_3^{(2)}, f_4^{(2)}) = W_{31}^{(3)} f_1^{(2)} + W_{32}^{(3)} f_2^{(2)} + W_{33}^{(3)} f_3^{(2)} + W_{34}^{(3)} f_4^{(2)}$$

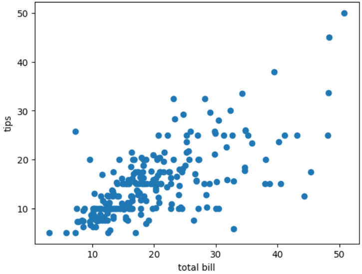
- ▷ The NN function can be seen as a **composition** of functions:

$$f_{\text{op}}(x_1, x_2, x_3) = f^{(3)}(f^{(2)}(f^{(1)}(x_1, x_2, x_3)))$$

- ▷ See demo at `https://playground.tensorflow.org`

Inverse Problem

- Given some relationships, how do we know the function?

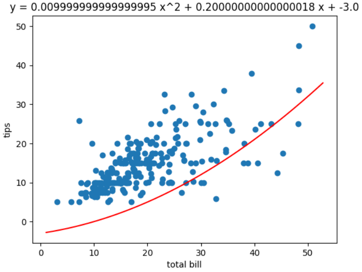


- **Assume** a **parametric** form for the function:

$$y = f(x; a, b, c) = ax^2 + bx + c$$

- *Brute force* to find the best a, b, c (see `interactive demo`) -

Inverse Problem & Optimization

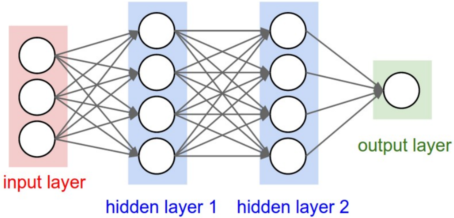


- Besides brute-forcing, a better approach to systemically find a, b, c that best **minimize** the error.
- This is an **optimization (regression) problem** (Monday)

$$\min_{a,b,c} \underbrace{|y_1 - f(x_1; a, b, c)| + \cdots + |y_m - f(x_m; a, b, c)|}_{\text{summed error!}}$$

Inverse Problem & Optimization

- Parameter fitting by optimization is also known as the **training** process for *machine learning*.



- We often use optimization to **train** the parameters (recall those $W_{11}^{(1)}, W_{12}^{(1)}, \ldots$) in an ANN!
- See demo `https://playground.tensorflow.org`

Summary

- Representing (i.e., abstractizing) relationships by function is a powerful concept —
  - allows us to study **trend** and make **prediction**
  - allows us to **model** real world behavior
  - important for **downstream** applications (AI, machine learning, etc.)
- After the break, we will build up more applications of functions: **inequality**, **graph problem**, **finance applications**, ...

How to use inequality to model decision?

Using inequality for decision modeling

$$X = \{x \mid f(x) \leq 0\}$$

- Interpreted as '*the set of all x such that $f(x) \leq 0$*'. Inequality defines a **set of numbers** 'shaped' by a function.
- **Example**:

$$f(x) = x - 10 \leq 0 \iff x \leq 10$$

- **Example**:

$$f(x) = x^2 - 10 \leq 0 \iff x^2 \leq 10$$

- **Example**:

$$f(x_1, x_2) = x_1 + x_2 - 10 \leq 0 \iff x_1 + x_2 \leq 10$$

Visualizing Inequality

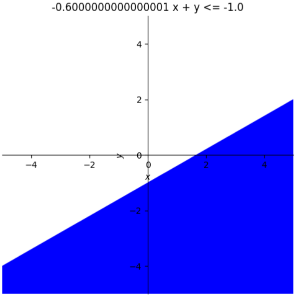
- **Example**: $x_1 + x_2 \leq 1$

- **Example**: $x_1^2 + x_2^2 \leq 1$

- *What about* $\{x_1, x_2 : x_1 + x_2 \leq 10, x_1^2 + x_2^2 \leq 1\}$*?*
  — interpreted as $x_1 + x_2 \leq 1$ AND $x_1^2 + x_2^2 \leq 1$

Linear Inequality & Half-space

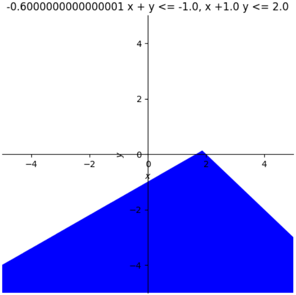
- When $f(x)$ is a **linear function**, its resulting inequality defines a *half-space* whose *half-plane* is defined by $f(x) = 0$.



- **Example**: refer to `interactive demo`

Intersection of Linear Inequalities

- Each linear inequality forms a *half-space*, the **intersection** of them forms a *polygon*!



- **Example**: how does the set

$$X = \{x_1, x_2 : x_1 + x_2 \leq 3, -x_1 + x_2 \leq 5, x_1 - x_2 \leq 3\}$$

look like? — refer to `interactive demo`

Optimizing with inequality

- Continuing with the 'inverse problem' example, we mayz consider

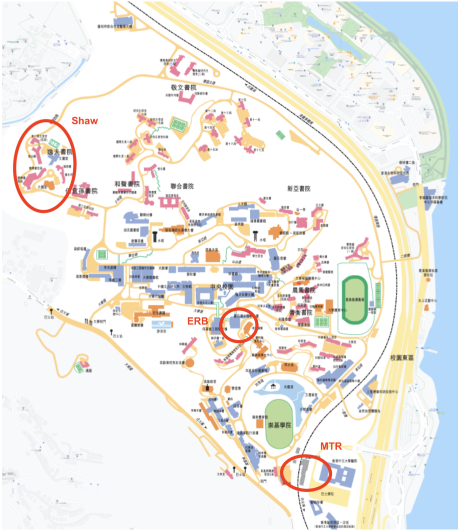
$$\begin{array}{ll} \min_{a,b,c} & |y_1 - f(x_1; a, b, c)| + \cdots + |y_m - f(x_m; a, b, c)| \\ \text{s.t.} & \underbrace{-10 \le a \le 10, -10 \le b \le 10, -10 \le c \le 10}_{\text{restrictions on } a, b, c} \end{array}$$

- Imagine that a, b, c are some decisions we are about to make, the perspective of **constrained optimization** allows us to make more **precise** decision.
- On Monday, we will learn about **linear program** which is a powerful class of optimization problems for decision making.

Graph Problem & Example of Algorithms (Optional)

Shortest Path Problem

- **Q**: what is the shortest path from *MTR* to *Shaw*?



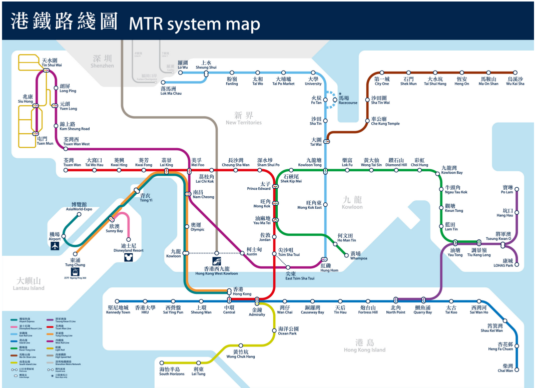
Shortest Path Problem

- How to solve the problem 'properly'? by *Mathematical Modeling*.
  - We have seen examples — functions, inequalities, etc. — on what are mathematical models
- In this case, we will need a **Graph Model**.
- The graph model enables us to **define** the problem properly.
- To solve the problem, we need **algorithm** — a set of procedures — that will lead us to solving **every** shortest path problem.

Examples of Graphs

- Metro networks (ignore the Lightrail line below)

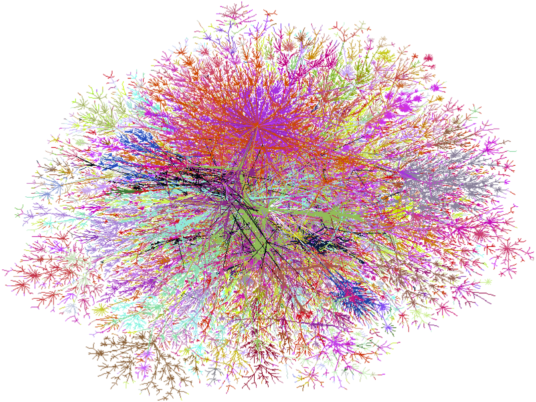
$$\text{vertex} \longleftrightarrow \text{station}$$
$$\text{edge} \longleftrightarrow \text{railtrack}$$



Examples of Graphs

- Internet

$$\text{vertex} \longleftrightarrow \text{webpage}$$
$$\text{edge} \longleftrightarrow \text{links}$$



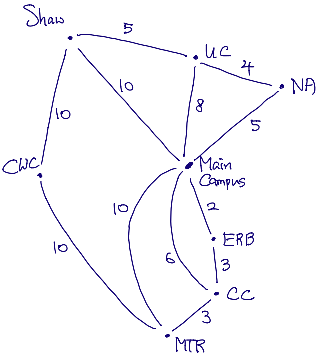
Examples of Graphs

- Social Network

$$\text{vertex} \longleftrightarrow \text{people}$$
$$\text{edge} \longleftrightarrow \text{friendship}$$

Shortest Path Problem

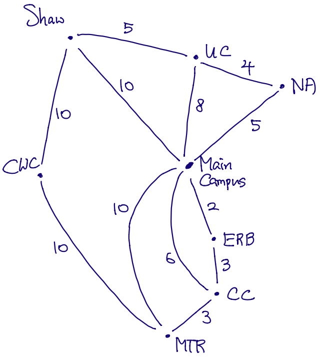
- Consider a **weighted graph**:



- Defined by $G = (V, E, w)$.
- V, E are called the **vertex** & **edge** sets.
- $w(\cdot)$ denotes the **weight** of an edge, e.g., the *travel time* on the edge.
- Example: $w(MTR, CC) = 3$, $w(CWC, Shaw) = 10$.

Shortest Path Problem (cont'd)

- **Q**: what is the shortest path from *MTR* to *Shaw*?

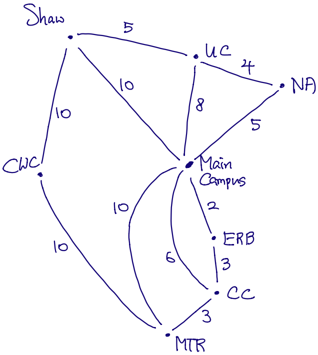


- Let $\mathrm{P} = (v_0, v_1, ..., v_k)$.
- Length of P is given by

$$\ell(\mathrm{P}) = \sum_{i=1}^{k} w(v_{i-1}, v_i)$$

Shortest Path Problem (cont'd)

- **Q**: what is the shortest path from *MTR* to *Shaw*?



- Possible paths are:

$$\mathrm{P}_1 : \mathit{MTR}, \mathit{CC}, \mathit{ERB}, \mathit{Main}, \mathit{Shaw}$$
$$\mathrm{P}_2 : \mathit{MTR}, \mathit{Main}, \mathit{Shaw}$$
$$\mathrm{P}_3 : \mathit{MTR}, \mathit{CWC}, \mathit{Shaw}$$
$$\mathrm{P}_4 : \mathit{MTR}, \mathit{Main}, \mathit{UC}, \mathit{Shaw}$$
$$\dots$$

- $\ell(\mathrm{P}_1) = 18$, $\ell(\mathrm{P}_2) = 20$, ...

how to find the shortest path on a large graph?

Properties of a Shortest Path

- **Observation 1**:

  A shortest path never passes twice the same vertex.

- **Observation 2**:

  If $\mathrm{P} = (v_1, v_2, ..., v_m)$ is a shortest path from v_1 to v_m, then a shortest path from v_1 to v_i, $i \leq m$ is the corresponding initial portion of P, i.e., $(v_1, ..., v_i)$.

Dijkstra's Algorithm

- **Initialize**: (A) maintain a table of cost $c(v)$, where starting vertex has cost $c(v_0) = 0$, others have cost $c(v_i) = \infty$, $i \neq 0$; (B) maintain a set of *visited* vertices $S = \emptyset$.
- **Iteration**:
  - Choose the unvisited vertex with minimum cost - denote it by v_{min} and update $S \leftarrow S \cup \{v_{min}\}$.
  - For *every neighbor* of v_{min} that are not in S,

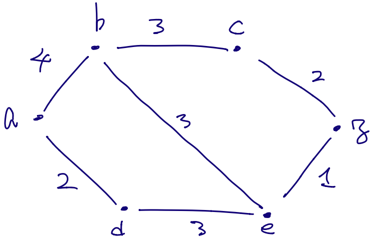
    $$c(v_i) \leftarrow \min\{c(v_i), c(v_{min}) + w(v_{min}, v_i)\}, \; \forall \; v_i \in \mathcal{N}_{v_{min}} \setminus S$$

  - End when every vertices are visited, i.e., $V = S$
- **Return**: a table of cost $c(v)$ with the shortest path distance.

Dijkstra's Algorithm - Example 1

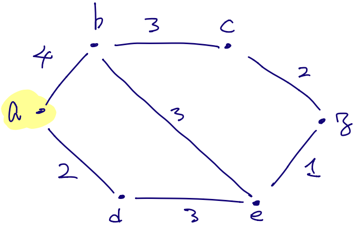
Q: what is the length of shortest path from a to z?

Initialization:



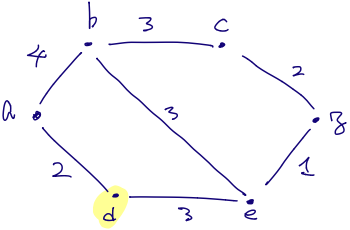
v	$C(v)$	Path
a	0	a.
b	∞	
c	∞	
d	∞	
e	∞	
z	∞	

$S = \phi.$



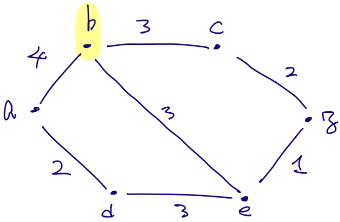
v	$C(v)$	Path
a	0	a
b	4	a,b
c	∞	-
d	2	a,d
e	∞	-
z	∞	-

$S = \{a\}$



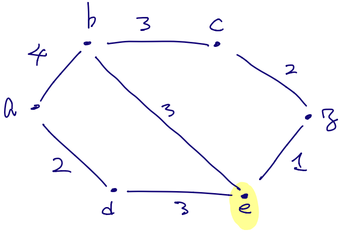
v	$C(v)$	Path
a	0	a
b	4	a,b
c	∞	-
d	2	a,d
e	5	a,d,e
z	∞	-

$S = \{a, d\}$



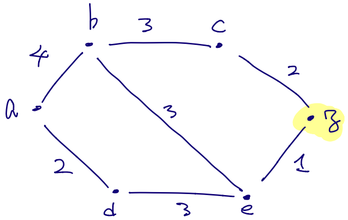
v	$C(v)$	Path
a	0	a
b	4	a, b
c	7	a, b, c
d	2	a, d
e	5	a, d, e
z	∞	–

$S = \{a, d, b\}$



v	$C(v)$	Path
a	0	a
b	4	a, b
c	7	a, b, c
d	2	a, d
e	5	a, d, e
z	6	a, d, e, z

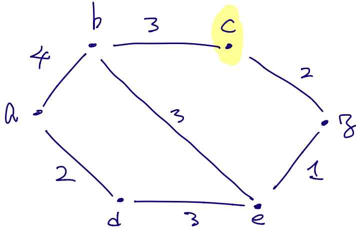
$S = \{a, d, b, e\}$



v	$C(v)$	Path
a	0	a
b	4	a, b
c	7	a, b, c
d	2	a, d
e	5	a, d, e
z	6	a, d, e, z

$S = \{a, d, b, e, z\}$

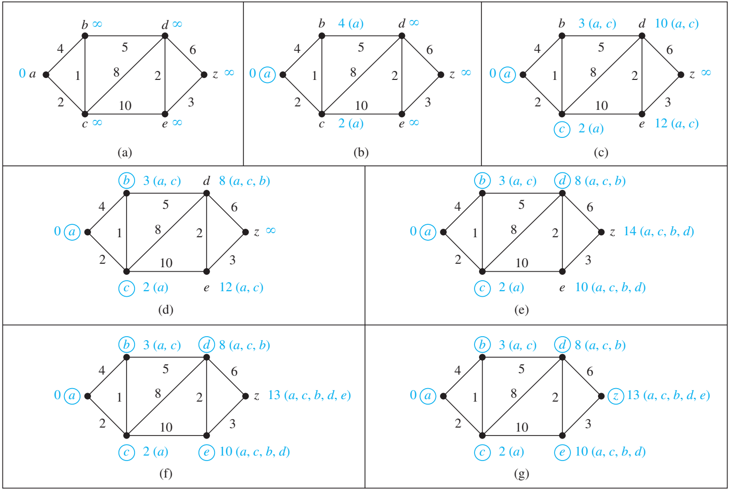
$\downarrow$



v	$C(v)$	Path
a	0	a
b	4	a, b
c	7	a, b, c
d	2	a, d
e	5	a, d, e
z	6	a, d, e, z

$S = \{a, d, b, e, z, c\}$

Dijkstra's Algorithm - Example 2



Dijkstra's Algorithm - Complexity

- **Claim**: The worst case runtime of the algorithm is $\mathcal{O}(n^2)$ (addition & comparisons), where n is the number of vertices.

1. The algorithm terminates in no more than n iterations.
2. At each iteration,
   - We can determine v^k with no more than $n-1$ comparison.
   - We can update $c(\cdot)$ by doing no more than $2(n-1)$ additions and comparisons.

- In the worst case, the algorithm completes in

$$n \times 3(n-1) = 3n^2 - 3n \text{ additions \& comparisons.}$$

- It may take much more steps if we don't use the Dijkstra's algorithm.

Summary

- A typical flow for problem solving:

  modelling $\longrightarrow$ **design & apply algorithm** $\longrightarrow$ **analyze output**

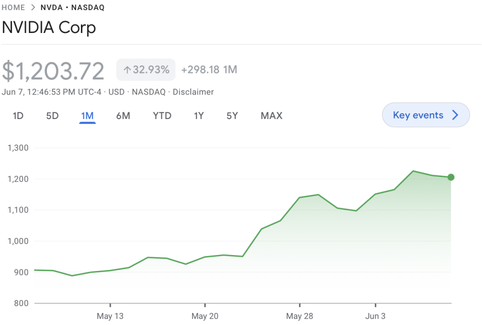
- These steps are inter-connected but requires **good understanding of the problem** and **good modelling**.

- Back to the Dijkstra's algorithm, note that it is one of the most popular algorithms for graph with applications to:
  - Google Maps, Rubik's cube, etc.

- Extensions: **graph with cycles**, **random shortest path** (to model congestion)

Mathematical Modeling in Finance, Systems Engineering & Data Science

Modeling Stock Prices



(Credits: Google Finance)

- A common model is the auto-regressive model — *stock price today* $\propto$ *stock price yesterday* $+$ *noise*

$$stock_t = a\, stock_{t-1} + b\, noise_t$$

- Parameters a, b are to be determined by **inverse problem**.
- Implication: knowing $a, b \Longrightarrow$ a model for stock price.

Modeling Opinions

- Our opinions are affected by peers/friends: we can model

$$\mathit{opinion}_i^{t+1} = \mathrm{avg}(\mathit{opinion}_j^t, \text{ j is i's friend})$$

- This leads to $\textbf{opinion}^{t+1} = f(\textbf{opinion}^t; \textit{network}) \Rightarrow$ can estimate the friendship network!

Takeaway for Today

- What is a function and how to visualize it.
- How to build inequalities (& constraints) based on functions and what are the geometric insights about it.

- (Optional) Advanced applications of functions (artificial intelligence, optimization, inverse problem).
- (Optional) Shortest path problem — how to go from math. modeling to applying an algorithm for problem solving.
- (Hopefully) Appreciate the beauty of mathematical modeling and optimization!

Enjoy and See you on Monday!