計算思維與人工智慧應用導論

計算思維 & 功能模組

Dr. Chih-Hsun Wu

吳致勳 助理教授

國立政治大學人工智慧跨域研究中心

Date: 2023/9/19

課程簡介

- 本課程旨在介紹計算思維與人工智慧的基本概念、原理和應用，為之後修習資訊相關課程奠下基礎。21世紀數位時代，如何具有應用人工智慧技術能力，培養計算思維是第一步，因為計算思維是分析問題、制定解決方法的思考過程，其中解決的步驟得依循電腦運算邏輯，因此在有了抽象思考能力後，將進一步介紹電腦具體運作架構，兩者的結合將由人工智慧程式應用來加以練習，此外也將介紹人工智慧的發展歷程和未來趨勢，以及人工智慧對社會、經濟和文化的影響。

課程目標

- 透過數位世界中資訊科技的內涵、網路世界的發展與數位世界大數據的挑戰，培養同學資訊專業知能、簡單程式能力跟邏輯、跨領域知能、自我反思能力、公民素養等核心能力。透過期末小組專題培養溝通表達能力、領導力及團隊合作能力等核心能力。透過數位學習平台培養終身學習能力等核心能力。

課程進度

週次	課程主題	課程內容與指定閱讀
1	計算思維簡介	社會情境脈絡與未來發展 書目：1, 2, 3
2	計算思維	基本內涵與核心概念 書目：1, 2, 3
3	功能模組	問題拆解與型態辨認 書目：4, 5, 6
4	功能模組	抽象思考與演算邏輯 書目：4, 5, 6
5	國慶日	國定假日
6	類比至數位轉換 & 電腦運算架構	類比與數位訊號的基礎概念及類比轉換至數位訊號的原理 & 電腦組成元件與其運算架構 書目：7 chapter 1 & 4 & 5
7	大數據應用	大數據中資料科學的基礎分析概念與商業相關應用
8	學習成果測試	期中評量/作業活動
9	運算思維測驗	國際運算思維挑戰賽
10	人工智慧發展	人工智慧發展歷程與未來趨勢
11	人工智慧技術與應用	人工智慧各式技術與應用案例 書目：8
12	人工智慧應用場景	人工智慧跨域應用
13	人工智慧學習模型實作	Nocode AI 練習 – Rapidminer 書目：9
14	人工智慧倫理	生成AI (如：ChatGPT、Deepfake、Midjourney)、假新聞及未來人工智慧應用上的倫理問題
15	人工智慧專題	海報展示
16	計算思維與人工智慧	期末報告
17	彈性補充教學	人工智慧相關競賽經驗交流
18	彈性補充教學	校園人工智慧應用發想

教材與Office Hours

- 教材
- 將放至Moodle平台

- Office Hours
- 請先電郵聯繫 20031214@nccu.edu.tw
- 地點:行政大樓八樓AI中心

評分標準

- TA課 30%
- 作業 25%
- 課堂出席、參與 10%
- 期中作業/評量 10%
- 期末專題 25%　整合 Micro:Bit / RapidMiner 設計　參考他人作品，務必註明出處
  期末海報展演 (12/21 週四11:30~13:30 4門課聯合)+ 期末報告
  含組內互評/組外互評/助教與老師評量

使用生成式AI工具注意事項

(1) 作業得適當使用生成式AI工具，使用的過程和結果應在作業中透明地說明和引用，並需於附件載明所使用工具名稱，並詳述使用過程。

(2) AI生成內容應被視為工具產生的建議，請經過調整再作為作業內容。生成式AI工具得作為報告輔助，但學生仍需要進行批判性思考。

評分標準

- 作業 25%
- Moodle 平台出題/繳交
- 遲交 / 補交
  - 該次作業分數打八折 => 滿分只有 80
  - 包括線上存為草稿，未送出
  - 最後補交期限為期末報告前一個星題天，之後不再開放補交
  - **缺交會影響總成績 = 25/作業數**
    - 例如: 6次作業，缺交一次就 -4.167

加分

1. 參與人工智慧、大數據分析相關競賽

2. 課堂報告
   - 自訂題目(需先向老師說明主題大綱等並經老師認可)
   - 3-5人一組
   - 報告一週前繳交投影片給老師及助教，經確認OK後可於隔週報告
   - 報告時間8分鐘

計算思維與人工智慧 – TA 課說明

助教資訊

程至榮

111753151@g.nccu.edu.tw

林孝道

112755009@g.nccu.edu.tw

管漢程

112753134@g.nccu.edu.tw

課程內容

單元	標題	授課內容	課堂作業	上課地點
1	程式桌遊	Coding ocean:海霸	點名	
2	micro:bit 1	Basics, Logic	程式碼改寫	
3	micro:bit 2	Loops, Functions	程式碼改寫	
4	RapidMiner 1	機器學習概念、數據預處理	Kaggle 競賽	
5	RapidMiner 2	數據分割、ML 模型(迴歸問題)	Kaggle 競賽	
6	RapidMiner 3	ML 模型(分類問題)	Kaggle 競賽	
7	期末報告討論	分組討論期末報告	分組討論	

作業規則

- TA 課前會發布上課簡報 + 作業相關檔案，每次作業 Deadline 為**兩週 (上課當週視為第一週)**
- 未在時間內繳交作業則視為**補繳，作業分數打 8 折**。
- 如需作業補交，請在 Deadline 後**兩週**內繳交，並且email 通知助教; 逾期不予補繳。
- 作業必須**全對**才會有分數 (包含繳交正確格式)

給分規則

TA 課程佔總成績的 30%

- 每次作業佔 5%，共 6 次。
- 程式桌遊 (1次) → 有點名就得 5 分
- micro:bit (2次) → 繳交作業專案連結，依據完成度給分 (基本 4 分 + 加分 1 分)
- RapidMiner (3次) → 參與指定 Kaggle 競賽，依據班上排名給分 (參與競賽 4 分 + 班排前 50% 1 分)

上課原則

- 只要來上課，助教會帶著大家起碼拿到基本題的分數 (4 分)
- 要拿到作業分數不困難，但是上課內容對初學者應該會有點難，建議跟課
- 軟體操作的問題請寄 email 約時間實體討論

加簽規則

- 請注意課號有沒有選錯
- 需要加簽的同學，請**務必**加選這門課、並確認此課有出現在你的【遞補清單】，之後才可以從選課系統列印符合規則的加簽單，無法透過其他管道加簽 (如自己列印空白加簽單，這是無效的)。請同學熟讀選課規則
- 列印加簽單後，請到行政大樓八樓AI中心找行政助理陳柏宏蓋章，並送資訊學院辦公室

新設課程介紹

人工智慧跨域微學程
(AI跨域微學程)

還在煩惱如何跨界人工智慧領域？
那建議你一定不要錯過AI跨域微學程
往下看帶你了解！

課程規劃單位：資訊學院 X 人工智慧應用學士學位學程(AI學位學程) X 人工智慧跨域研究中心(AI中心)

背景介紹

01 - 何謂微學程

微學程是為鼓勵學生系統性的修習各領域知識，以強化同學自身能力而規劃的課程型態，以8-12學分課程組成，依課程組合特性分為入門微學程及進階微學程，學生修滿規定科目及學分數後可取得微學程設置單位核發的修業證明。

非輔系

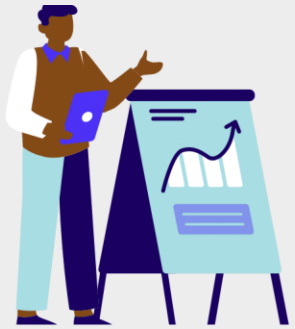
非雙主修

02 - AI跨域微學程又是...?

AI跨域微學程是因應AI世代來臨，為培養同學對人工智慧(AI)之素養，並建立認識AI、辨別AI以及運用AI之學習管道而規劃設立之課程，在分類上屬於入門微學程，非常適合想初探人工智慧領域學生選修。

AI跨域微學程組成

基礎資訊通識/3學分
計算思維與人工智慧



基礎課程旨在介紹計算思維與人工智慧的基本概念、原理和應用，為之後修習資訊相關課程奠定基礎。將先介紹計算思維於電腦中運作之方式，進而深化至人工智慧如何學習知識，此外也將介紹人工智慧的發展歷程和未來趨勢，以及人工智慧對社會、經濟和文化的影響。

零基礎可入門

進階選修/3學分
人工智慧方法與工具

進階課程主要介紹人工智慧應用中常用之技術工具，搭配資料分析、社群網路分析、機器學習、深度學習、生成式AI等不同主題引入不同工具之介紹，藉由操作練習培養運用人工智慧技術之能力，並透過成果報告方式促進不同領域背景學生互相交流學習成果。

電腦教室上課手把手教學

應用選修/3學分
人工智慧實務專題

應用課程為實務專題演練課程，將由本校不同領域背景學生組成跨域小組，共同創新激盪於AI世代潮流中得以運用AI解決之問題，並提出解決方案，在創意發想實踐的同時，亦思考AI議題之多元性與反思AI技術之影響，希冀學生得於修習後具備自主運用AI、懂得審慎辨別AI影響之跨域人才。

需先修畢人工智慧方法與工具

修業規範

修滿基礎、進階、應用科目各 3 學分，共 9 學分

以下任一程式能力（於申請修業證明時提出）

(1)大學程式設計先修檢測(Advanced Placement Computer Science, APCS) 之程式設計觀念題 3 級分以上及程式設計實作題 2 級分以上。

(2)大學程式能力檢定 (Collegiate Programming Examination, CPE) 考試通過 1 題以上。

(3)修畢「程式設計概論」或其他經認定之程式設計相關課程者。

可申請AI跨域微學程修業證明

我能修課嗎?

身份/項目	能否修課？	採計為畢業學分？	學分費？
學士班	✓	✓	• 大一至大四不需繳學分費 • 延畢者，大五(含)以上需按當年度學雜費收費標準繳交學分費及相關費用
碩博班	✓	✗	按當年度學雜費收費標準繳交學分費及相關費用

- 微學程學分認定、學分費詳見本校微學程設置辦法

112-1 AI跨域微學程開課表

	一	二		三	四	五
1						
2		人工智慧方法與工具 進階選修 預收**60**人	計算思維與人工智慧應用導論 基礎選修 預收**100**人	人工智慧方法與工具 進階選修 預收**60**人	人工智慧方法與工具 進階選修 預收**60**人	計算思維與人工智慧應用導論 基礎選修 預收**100**人
3						
4						
C						
D	人工智慧方法與工具 進階選修 預收**60**人					
5						
6						
7				計算思維與人工智慧 基礎通識 預收**135**人	計算思維與人工智慧 基礎通識 預收**100**人	
8						
E						
F				計算思維與人工智慧 基礎通識 為三**78**之實習課程 兩週一次		
G						

- 微學程細則將於本學期修法，因作業時程較長，故112-1學期基礎科目將開設兩門：選修課「計算思維與人工智慧應用導論」，以及通識課「計算思維與人工智慧」，同學可任擇一修習，皆可採計為AI跨域微學程之修業學分。

AI跨域微學程 vs AI學位學程

項目/課程	人工智慧跨域微學程(AI跨域微學程)	人工智慧應用學士學位學程(AI學位學程)
課程定位	微學程	學位學程(雙主修)
修畢學分	基礎/進階/應用各3學分 共計9學分	最低畢業學分： 必修6學分+群修36學分=42學分
修業證明	由資訊學院授予修業證明	由學校授予資訊學學士學位
招收對象	全校學士班學生 (碩博生經授課教師同意可修習，但不採計為畢業學分)	全校各學系大二(含)以上學生
招收名額	依各科目開設班級與人數為準 1.基礎-計算思維與人工智慧，每學期3班，每班100人 2.進階-人工智慧方法與工具，每學期3班，每班60人 3.應用-人工智慧實務專題，每學期3班，預計每班60人	112年為24名，每年名額依當年度教務處公告為準
學分折抵	學生修習112-1開設之選修課：計算思維與人工智慧應用導論， 得採列為AI跨域微學程之修業學分	學分抵免說明詳見AI學位學程網站

跨界AI領域

歡迎選修 AI跨域微學程 與 AI學程

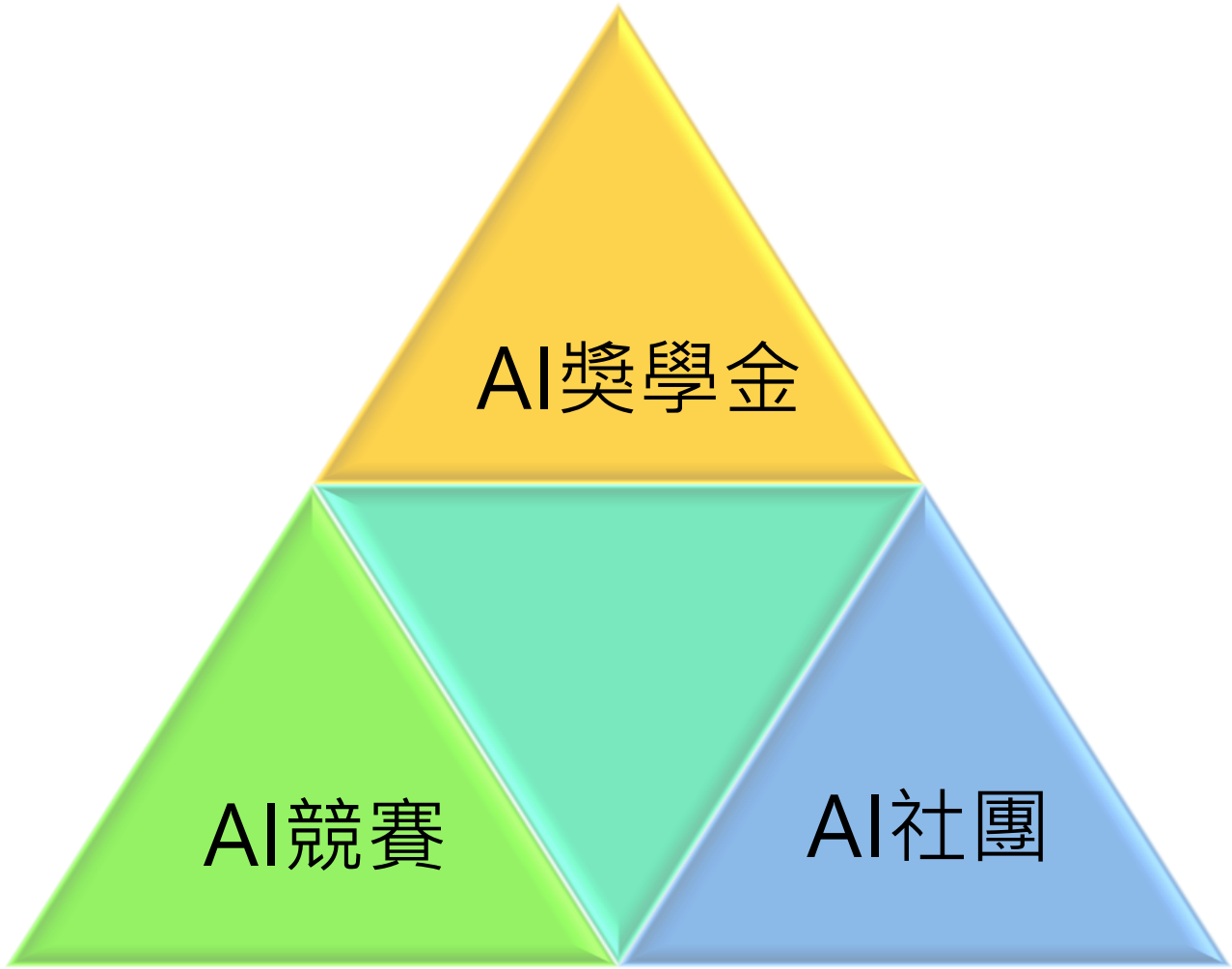
修課建議

AI跨域微學程　AI學程

校內AI跨域課程地圖
詳見AI中心網站

AI跨域微學程課程諮詢：ai@nccu.edu.tw

掌握政大AI三方向



計算機演進

計算機科學大事紀

西元前3000年

早期的計算裝置之一是算盤(abacus)。西元前3000年算盤已出現在亞洲，有人說是巴比倫人發明的；也有人說是中國人發明的，無論如何，亞洲人發明了算盤應無庸置疑。

西元1642年

法國數學家Blaise Pascal發明了機械式的加法器 Pascaline。

剛開始計算機的設計是以齒輪技術為基礎

機械式計算機

- mechanical computer: 由槓桿、齒輪等機械部件而非電子部件構成。最常見的例子是加法器和機械計數器，它們使用齒輪的轉動來增加顯示的輸出。更複雜的例子可以進行乘法和除法。1960年代曾出售一個計算平方根的模型。

Credit by 109301022

剛開始計算機的設計是以齒輪技術為基礎

- https://youtu.be/3h71HAJWnVU?t=46s

西元1801年

Joseph-Marie Jacquard 發明了Jacquard loom

- 第一部使用儲存器及程式設計概念的機器。

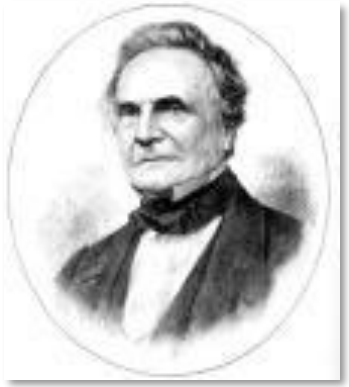
以打孔卡片(punched card)來控制織布機的編織流程。

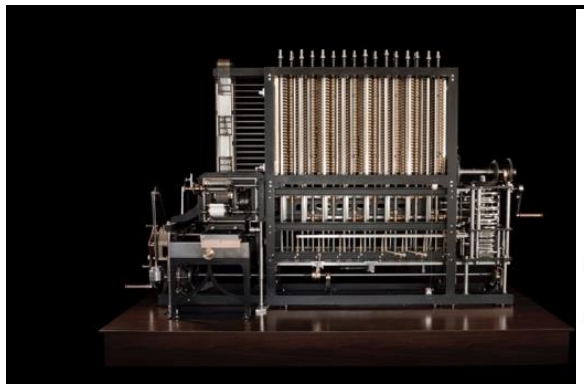
https://youtu.be/K6NgMNvK52A?t=9s
https://youtu.be/K6NgMNvK52A?t=2m6s

西元1822年

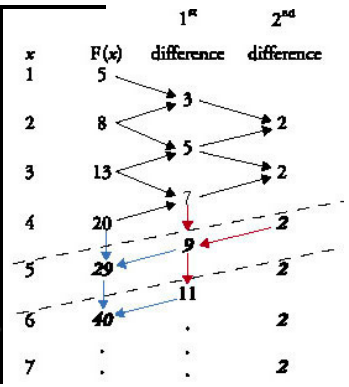
Charles Babbage開始設計Difference Engine，指令可修改而執行許多簡單的數學運算，也可解多項式問題。

There were enough 'registers' for seven differences, allowing it to compute 31-digit values for polynomials with terms up to x^7.





http://www.computerhistory.org/babbage/



http://www.computerhistory.org/babbage/

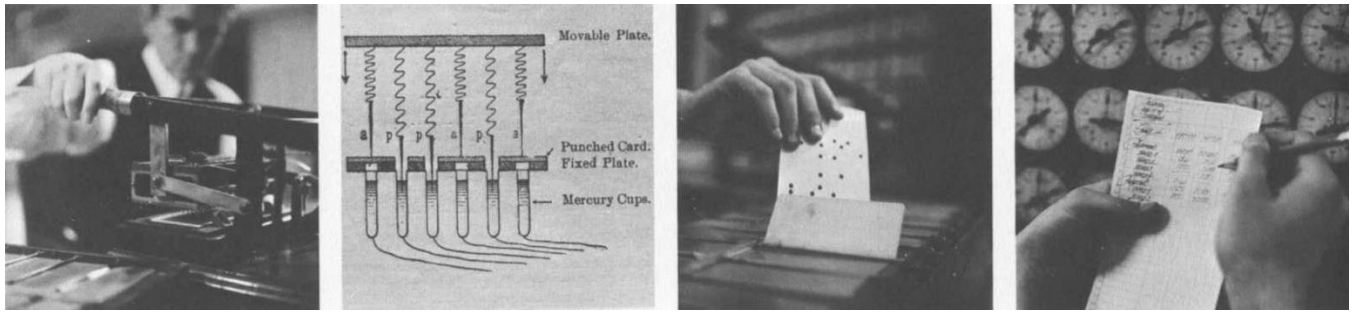
https://youtu.be/jiRgdaknJCg?t=8s

西元1889年

1880 美國人口普查耗時 7 年
Herman Hollerith 設計以打孔卡片來儲存資料並排序的電動機器。協助1890人口普查，只耗費兩年半即完成調查工作，原來這項工作預計需耗時13年

Herman Hollerith 在西元1896年時成立了Tabulating Machine Company，於西元1924年2月14日，正式改名為IBM (International Business Machines Corporation)。

https://en.wikipedia.org/wiki/File:HollerithMachine.CHM.jpg



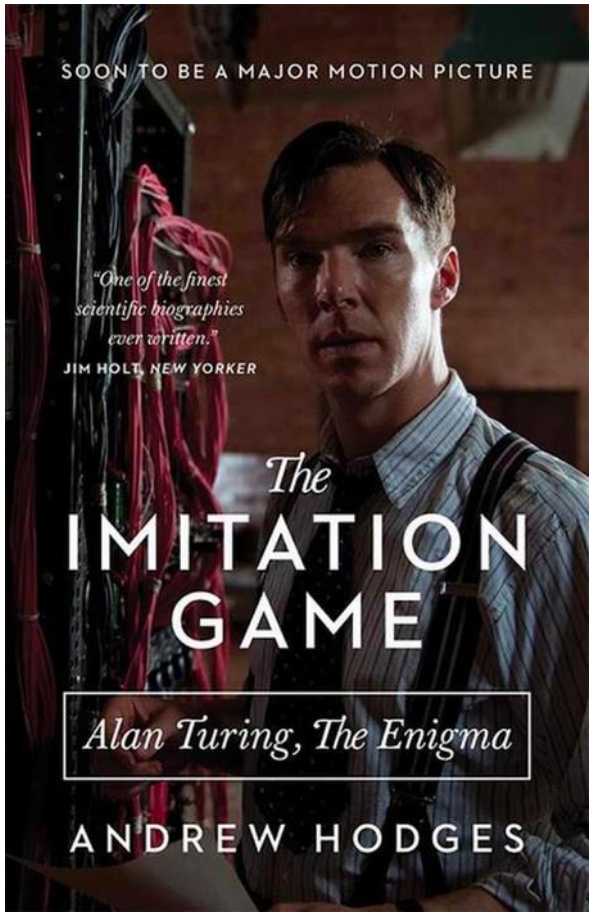
https://youtu.be/YnnGbcM-H8c?t=28s

Alan Turing

Alan Turing

1912-1954
電腦科學創始人、數學家、哲學家、密碼破解者、夢想家

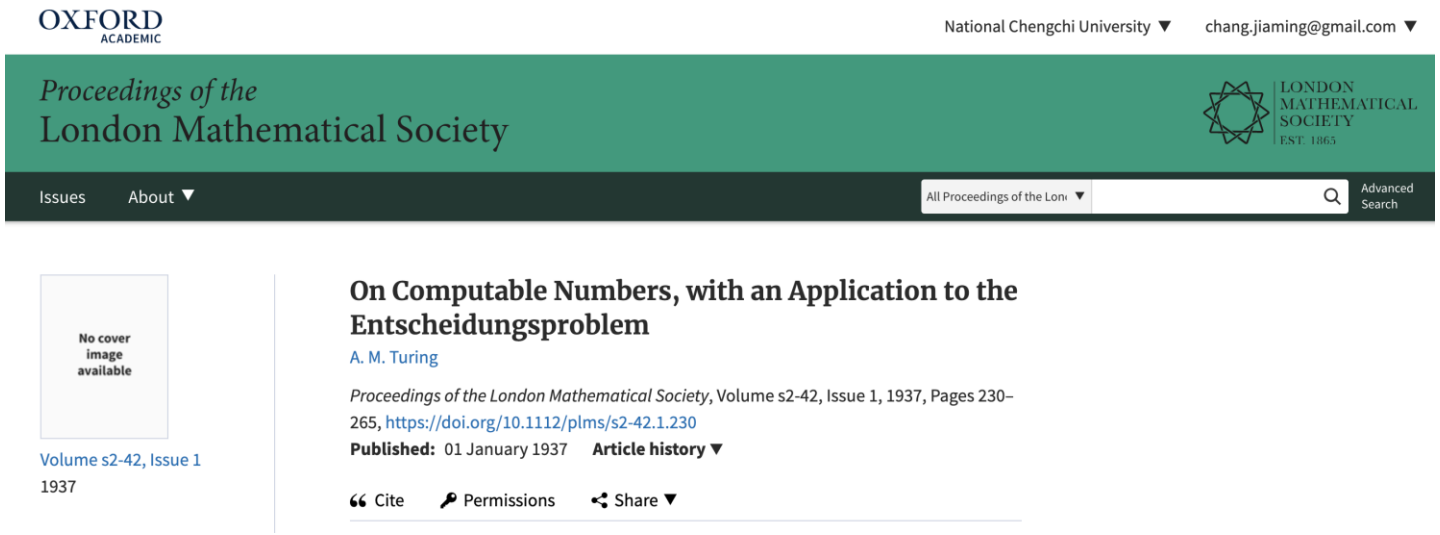
1912.06.23 ~ 1954.06.07



http://yule.sohu.com/20140910/n404190750.shtml

圖靈機（Turing Machine）

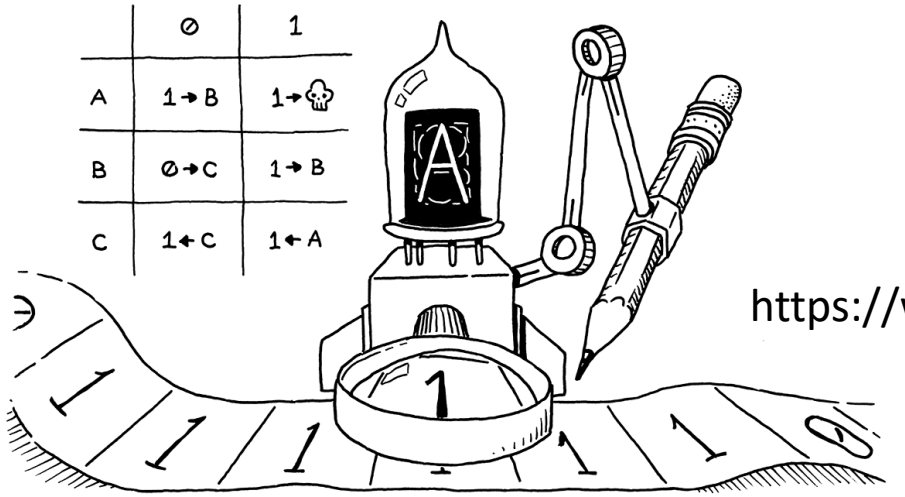
- 1936 年，24 歲的圖靈發表了一篇論文《論可計算數及其在判定問題上的應用》（On Computable Numbers, with an Application to the Entscheidungsproblem）。
- 在這篇極富開創性的論文中，圖靈提出了**「圖靈機」**（Turing Machine）概念。
  - https://academic.oup.com/plms/article-abstract/s2-42/1/230/1491926?redirectedFrom=fulltext



Credit by 寫點科普，請給指教 https://pansci.asia/archives/119338

圖靈機

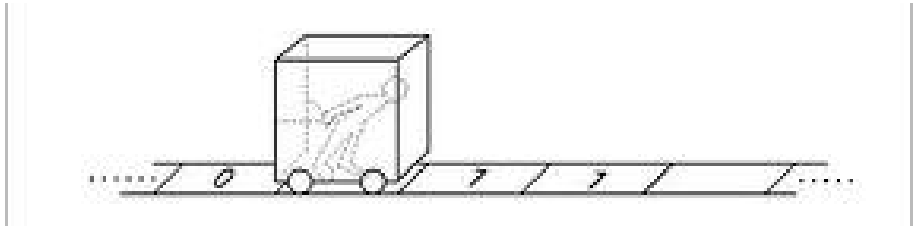
- 圖靈機就是一張表格：現在的狀態 →下一步怎麼走 → 移動 → 現在的狀態。
- 圖靈機 @ Wikipedia
  - 圖靈的基本思想是用機器來類比人們用紙筆進行數學運算的過程，他把這樣的過程看作下列兩種簡單的動作：
    - 在紙上寫上或擦除某個符號；
    - 把注意力從紙的一個位置移動到另一個位置；
    - 而在每個階段，人要決定下一步的動作，依賴於（a）此人當前所關注的紙上某個位置的符號和（b）此人當前思維的狀態。

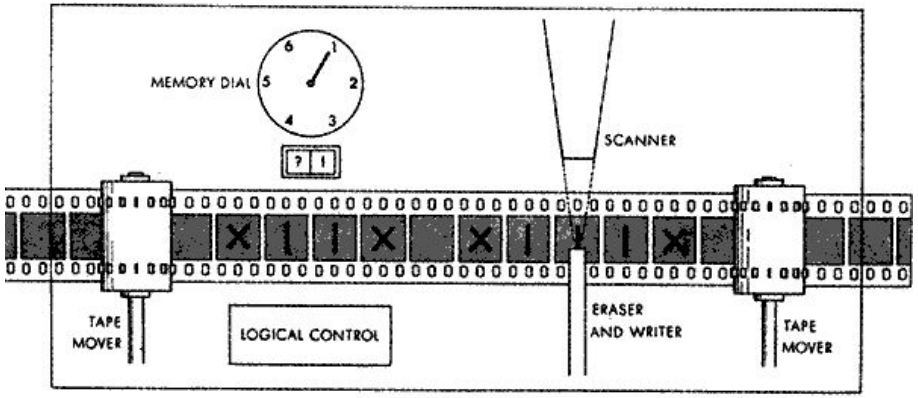


https://www.itread01.com/hkcpfli.html

Turing Machine

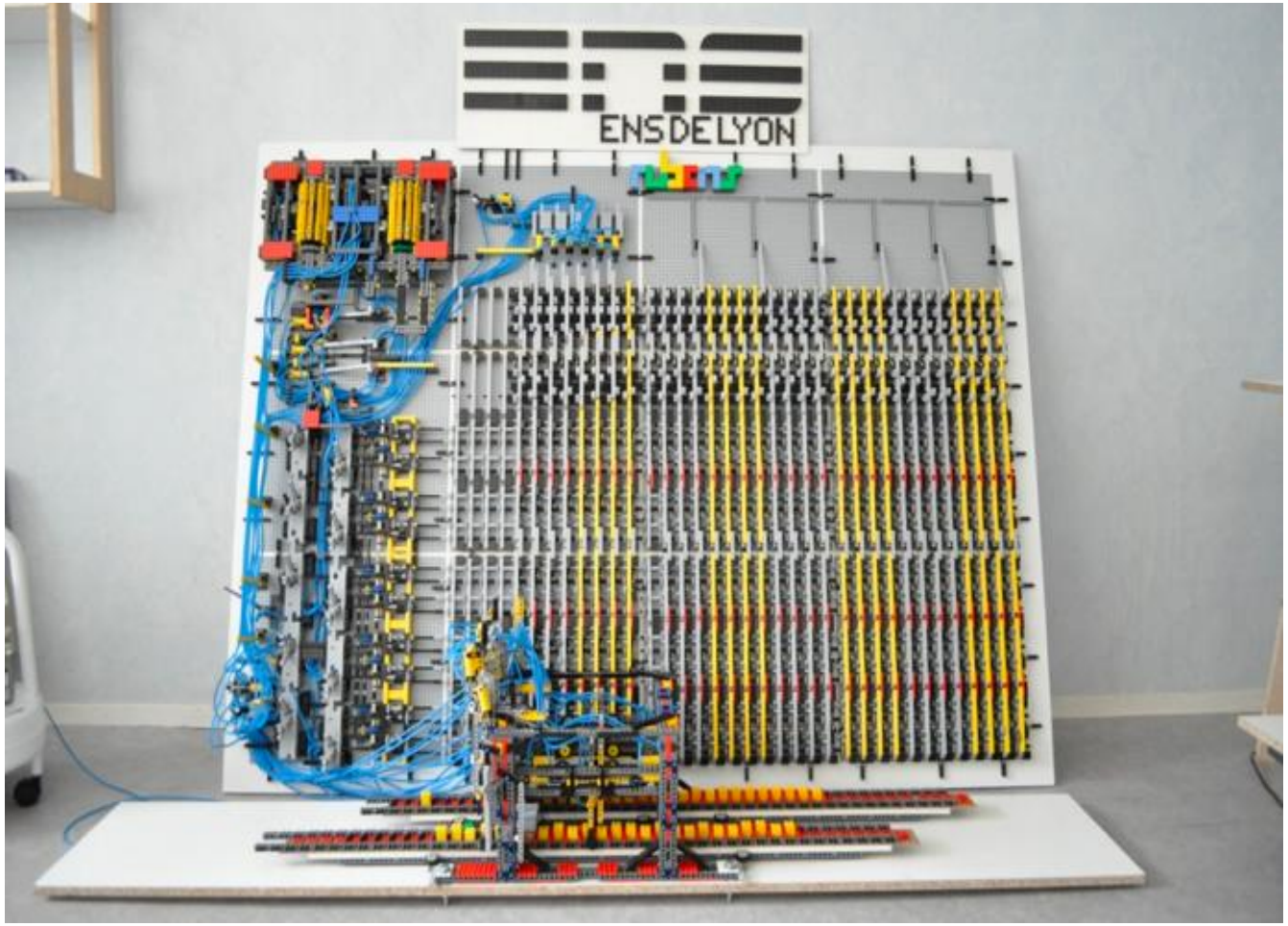
- https://www.youtube.com/watch?v=dNRDvLACg5Q
- https://www.youtube.com/watch?v=gJQTFhkhwPA





A Turing machine realisation in LEGO

https://www.youtube.com/watch?v=FTSAiF9AHN4



VOL. LIX. No. 236.] [October, 1950

MIND

A QUARTERLY REVIEW

OF

PSYCHOLOGY AND PHILOSOPHY

I.—COMPUTING MACHINERY AND INTELLIGENCE

BY A. M. TURING

1. *The Imitation Game.*

I PROPOSE to consider the question, 'Can machines think?' This should begin with definitions of the meaning of the terms 'machine' and 'think'. The definitions might be framed so as to reflect so far as possible the normal use of the words, but this attitude is dangerous. If the meaning of the words 'machine' and 'think' are to be found by examining how they are commonly used it is difficult to escape the conclusion that the meaning and the answer to the question, 'Can machines think?' is to be sought in a statistical survey such as a Gallup poll. But this is absurd. Instead of attempting such a definition I shall replace the question by another, which is closely related to it and is expressed in relatively unambiguous words.

The new form of the problem can be described in terms of a game which we call the 'imitation game'. It is played with three people, a man (A), a woman (B), and an interrogator (C) who may be of either sex. The interrogator stays in a room apart from the other two. The object of the game for the interrogator is to determine which of the other two is the man and which is the woman. He knows them by labels X and Y, and at the end of the game he says either 'X is A and Y is B' or 'X is B and Y is A'. The interrogator is allowed to put questions to A and B thus:

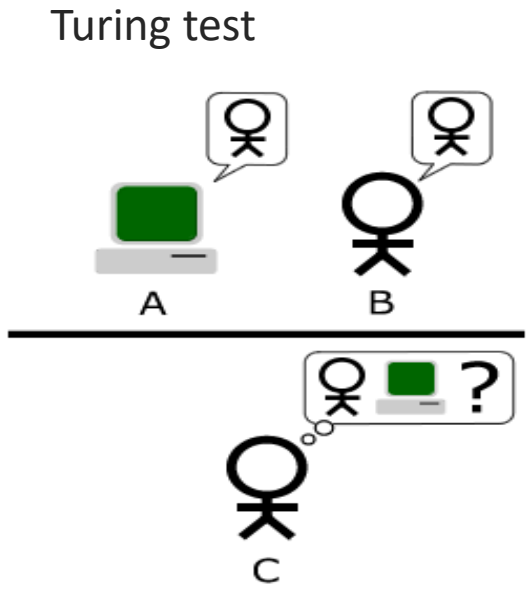
C: Will X please tell me the length of his or her hair?

Now suppose X is actually A, then A must answer. It is A's

28 433

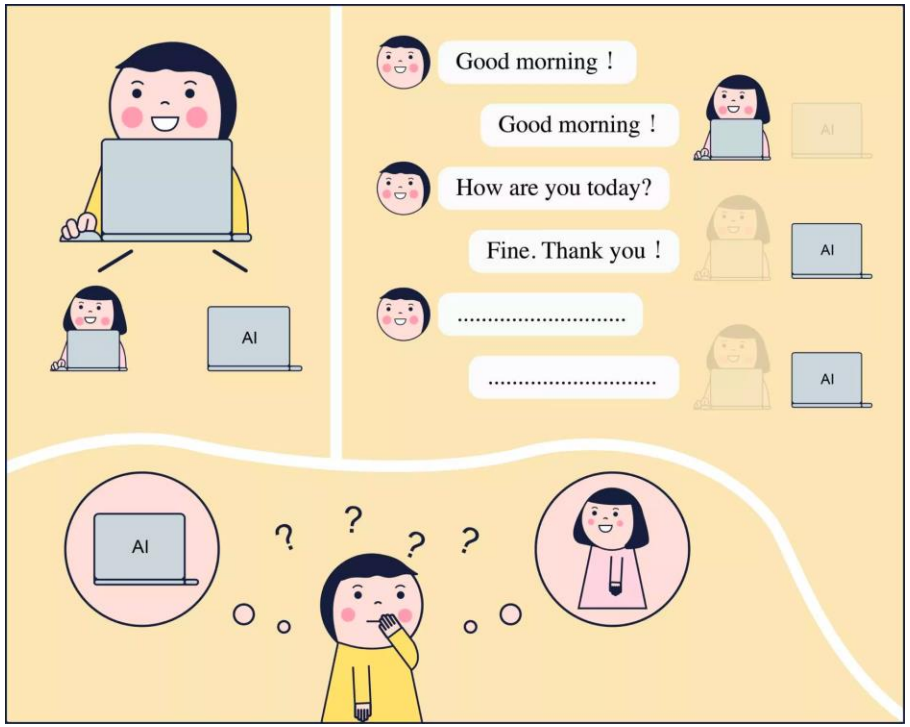
https://academic.oup.com/mind/article/LIX/236/433/986238?login=true

《計算機器和智慧》的論文
提問「機器會思考嗎？」（*Can Machines Think?*）
圖靈測試(Turing Test)：如果一台機器與人類對話而不被辨別出其機器身分，那麼稱這台機器具有智慧。
利用電腦模仿人類交談的遊戲，來判斷機器是否像人一樣有思考能力。說明「思考的機器」是可能的。



Figures adapted from Saygin, 2000

圖靈機到人工智慧，誰讓電腦強大？是數學！



- https://research.sinica.edu.tw/turing-machine-math-makwlih/
- Credit by 2017/12/22 研之有物
- 本篇來自 研之有物 ，INSIDE 經授權轉載。

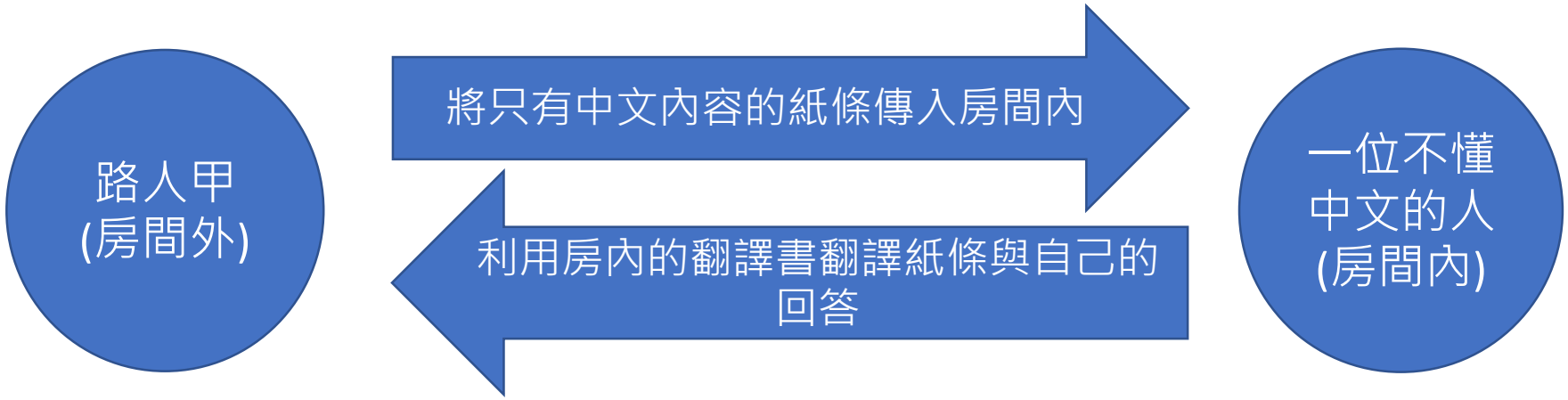
人造意識

https://www.imdb.com/title/tt0470752/

Ex Machina | 'The Turing Test' Clip | Film4
https://www.youtube.com/watch?v=LI3Z5vIDAgE

圖靈測試合理嗎？

- The Chinese room
  - https://www.youtube.com/watch?v=TryOC83PH1g
- 由約翰·希爾勒提出的一個思想實驗,藉以反駁強人工智慧的觀點。
- 實驗概要:



電腦科學的諾貝爾獎?

- 物理學獎
- 化學獎
- 生理學或醫學獎
- 文學獎
- 和平獎
- 瑞典銀行經濟學獎

- 數學獎:
  - 4年一次，40歲以下

諾貝爾獎

菲爾茲獎

圖靈獎 Turing Award

- The ACM A.M. Turing Award, often referred to as the “Nobel Prize of Computing,” carries a $1 million prize, with financial support provided by Google, Inc. It is named for Alan M. Turing, the British mathematician who articulated the mathematical foundation and limits of computing.
- Credit by https://amturing.acm.org/

By the Association for Computing Machinery (ACM)

Alan Mathison Turing Award

(2021)
Dongarra, Jack

(2020)
Aho, Alfred Vaino
Ullman, Jeffrey David

(2019)
Catmull, Edwin E.
Hanrahan, Patrick M.

(2018)
Bengio, Yoshua
Hinton, Geoffrey E
LeCun, Yann

(2017)
Hennessy, John L
Patterson, David

(2016)
Berners-Lee, Tim

(2015)
Diffie, Whitfield
Hellman, Martin

(2014)
Stonebraker, Michael

(2013)
Lamport, Leslie

(2012)
Goldwasser, Shafi
Micali, Silvio

(2011)
Pearl, Judea

(2010)
Valiant, Leslie Gabriel

(2009)
Thacker, Charles P. (Chuck) *

(2008)
Liskov, Barbara

(2007)
Clarke, Edmund Melson *
Emerson, E. Allen
Sifakis, Joseph

(2006)
Allen, Frances ("Fran") Elizabeth *

(2005)
Naur, Peter *

(2004)
Cerf, Vinton ("Vint") Gray
Kahn, Robert ("Bob") Elliot

(2003)
Kay, Alan

(2002)
Adleman, Leonard (Len) Max
Rivest, Ronald (Ron) Linn
Shamir, Adi

(2001)
Dahl, Ole-Johan *
Nygaard, Kristen *

(2000)
Yao, Andrew Chi-Chih

(1999)
Brooks, Frederick ("Fred")

(1998)
Gray, James ("Jim") Nicholas *

(1997)
Engelbart, Douglas *

(1996)
Pnueli, Amir *

(1995)
Blum, Manuel

(1994)
Feigenbaum, Edward A ("Ed")
Reddy, Dabbala Rajagopal ("Raj")

(1993)
Hartmanis, Juris *
Stearns, Richard ("Dick") Edwin

(1992)
Lampson, Butler W

(1991)
Milner, Arthur John Robin Gorell ("Robin") *

(1990)
Corbato, Fernando J ("Corby") *

(1989)
Kahan, William ("Velvel") Morton

(1988)
Sutherland, Ivan

(1987)
Cocke, John *

(1986)
Hopcroft, John E
Tarjan, Robert (Bob) Endre

(1985)
Karp, Richard ("Dick") Manning

(1984)
Wirth, Niklaus E

(1983)
Ritchie, Dennis M.*
Thompson, Kenneth Lane

(1982)
Cook, Stephen Arthur

(1981)
Codd, Edgar F. ("Ted") *

(1980)
Hoare, C. Antony ("Tony") R.

(1979)
Iverson, Kenneth E. ("Ken") *

(1978)
Floyd, Robert (Bob) W *

(1977)
Backus, John *

(1976)
Rabin, Michael O.
Scott, Dana Stewart

(1975)
Newell, Allen *
Simon, Herbert ("Herb") Alexander *

(1974)
Knuth, Donald ("Don") Ervin

(1973)
Bachman, Charles William *

(1972)
Dijkstra, Edsger Wybe *

(1971)
McCarthy, John *

(1970)
Wilkinson, James Hardy ("Jim") *

(1969)
Minsky, Marvin *

(1968)
Hamming, Richard W*

(1967)
Wilkes, Maurice V.*

(1966)
Perlis, Alan J *

AlphaGo

by DeepMind, 2015~2017

人工智慧圍棋程式 AlphaGo，壓倒性擊敗棋王而轟動全世界，令人不禁疑問：機器可以思考嗎？機器可以超越人類心靈嗎？而著名的「圖靈 測試」(Turing Test)，就是對此問題的一種判定。

2015/10 AlphaGo vs 樊麾 5:0

- 2016/01/27, Nature Paper

樊麾歐洲圍棋冠軍 二段

Mastering the game of Go with deep neural networks and tree search

David Silver[1]*, Aja Huang[1]*, Chris J. Maddison[1], Arthur Guez[1], Laurent Sifre[1], George van den Driessche[1], Julian Schrittwieser[1], Ioannis Antonoglou[1], Veda Panneershelvam[1], Marc Lanctot[1], Sander Dieleman[1], Dominik Grewe[1], John Nham[2], Nal Kalchbrenner[1], Ilya Sutskever[2], Timothy Lillicrap[1], Madeleine Leach[1], Koray Kavukcuoglu[1], Thore Graepel[1] & Demis Hassabis[1]

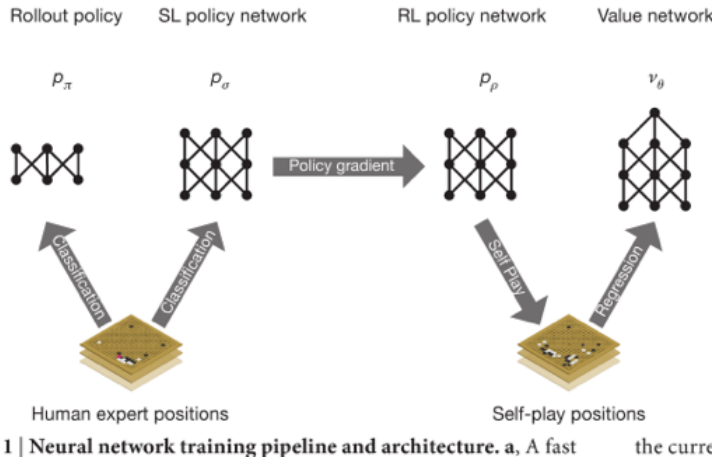


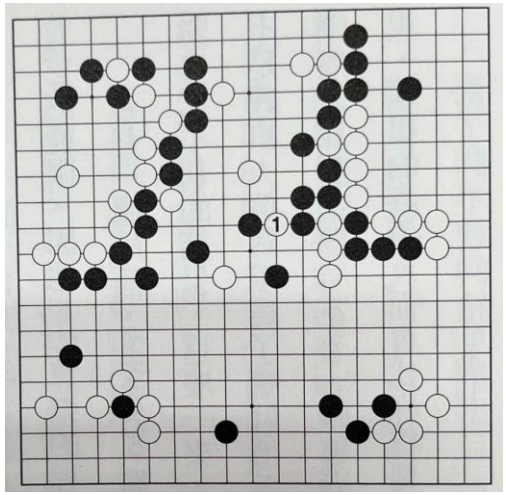
Figure 1 | Neural network training pipeline and architecture. a, A fast the curre

Silver,D. *et al.* (2016) Mastering the game of Go with deep neural networks and tree search. *Nature*, **529**, 484–9.

2016/03/09~15 @南韓首爾
AlphaGo vs 李世乭　4:1

局次	日期	黑方	白方	結果	手數
1	2016年3月9日	李世乭	AlphaGo	白中盤勝	186
2	2016年3月10日	AlphaGo	李世乭	黑中盤勝	211
3	2016年3月12日	李世乭	AlphaGo	白中盤勝	176
4	2016年3月13日	AlphaGo	李世乭	白中盤勝	180
5	2016年3月15日	李世乭 註	AlphaGo	白中盤勝	280
最終結果： AlphaGo 4－1 李世乭					

Credit by https://zh.wikipedia.org/zh-tw/AlphaGo李世乭五番棋



李世乭78手 – 挖, credit by王銘琬, 棋士與AI

李世乭韓國九段

http://www.9star.com.tw/shownews.asp?id=1775

2016/12 ~ 2017/01

- 2016/12/29, 韓國對弈網站 Tygem – Magist
  - ~10 勝/day
  - 30連勝
- 2017/01/01, 中國圍棋網站 野狐 – Master
  - 01/04, 60 連勝, 黃士傑: 我是 Master 的操手
    - 2017/11/10 AlphaGo - 深度學習與強化學習的勝利 by 黃士傑, 人工智慧年會
      - https://www.bnext.com.tw/article/46984/alphago-deepmind-aja-huang
      - 2016師大傑出校友黃士傑

2017/05/23~27 @ 圍棋未來高峰會,中國烏鎮
AlphaGo vs 柯潔 3:0

- Google's DeepMind lab is retiring its Go-playing machine as researchers eye a much bigger future.
  - https://www.wired.com/2017/05/win-china-alphagos-designers-explore-new-ai/

場次	日期	黑	白	結果	手數	
1	2017年5月23日	柯潔	AlphaGo	白半目勝	289	sgf
2	2017年5月25日	AlphaGo	柯潔	白棋認輸	155	sgf
3	2017年5月27日	AlphaGo	柯潔	白棋認輸	209	sgf
結果：AlphaGo3：0獲勝						

Credit by http://news.ltn.com.tw/news/world/breakingnews/2076634 (路透社)

2017/10/19, AlphaGo Zero

- Silver, D. et al. Mastering the game of Go without human knowledge. *Nature* 550,354–359 (2017).
  - https://www.nature.com/articles/nature24270

Mastering the game of Go without human knowledge

David Silver[1]*, Julian Schrittwieser[1]*, Karen Simonyan[1]*, Ioannis Antonoglou[1], Aja Huang[1], Arthur Guez[1], Thomas Hubert[1], Lucas Baker[1], Matthew Lai[1], Adrian Bolton[1], Yutian Chen[1], Timothy Lillicrap[1], Fan Hui[1], Laurent Sifre[1], George van den Driessche[1], Thore Graepel[1] & Demis Hassabis[1]

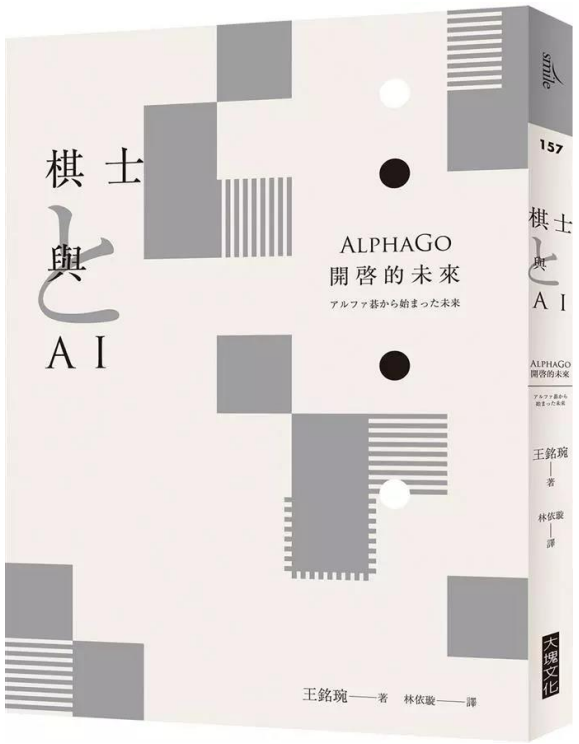
A long-standing goal of artificial intelligence is an algorithm that learns, *tabula rasa*, superhuman proficiency in challenging domains. Recently, AlphaGo became the first program to defeat a world champion in the game of Go. The tree search in AlphaGo evaluated positions and selected moves using deep neural networks. These neural networks were trained by supervised learning from human expert moves, and by reinforcement learning from self-play. Here we introduce an algorithm based solely on reinforcement learning, without human data, guidance or domain knowledge beyond game rules. AlphaGo becomes its own teacher: a neural network is trained to predict AlphaGo's own move selections and also the winner of AlphaGo's games. This neural network improves the strength of the tree search, resulting in higher quality move selection and stronger self-play in the next iteration. Starting *tabula rasa*, our new program AlphaGo Zero achieved superhuman performance, winning 100–0 against the previously published, champion-defeating AlphaGo.

AlphaGo Zero

- https://deepmind.com/blog/alphago-zero-learning-scratch/
- https://youtu.be/tXlM99xPQC8

棋士與AI: AlphaGo開啓的未來

- 作者：王銘琬
- 譯者：林依璇
- 出版社：大塊文化出版股份有限公司
- 出版日期：2018/09/28
- AI與人類的交會點
  - 事物的背後需要故事性嗎?
- 身為人的證明
  - 人類的矛盾與多種面貌
- 王銘琬 / AlphaGo大勝棋王後，人類的下一步？ 2016/05/12
  - https://www.twreporter.org/a/opinion-alphago



Deep Learning + Genomics

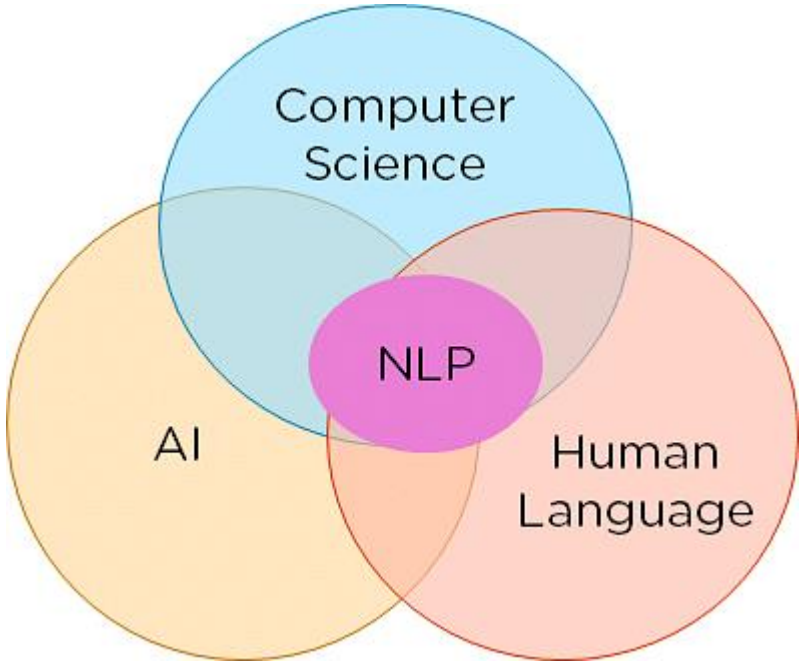
- An AI-Driven Genomics Company Is Turning to Drugs
  - Deep Genomics aims to develop drugs by using deep learning to find patterns in genomic and medical data.
  - Credit by Will Knight May 3, 2017 MIT Technology Review
  - The rush to apply AI techniques to medicine and drug development is partly driven by the emergence of powerful new algorithms, but also by cost-effective new ways of sequencing whole genomes, the entire readout of a person's DNA. "There's an opening of a new era of data-rich, information-based medicine,"
- 基因+人工智能，DeepGenomics将会把精准医疗带往那里？
  - 加拿大的《举世邮报》暗示“这家多伦多创业公司意图撼动基因测序市场”；而美国《华盛顿邮报》则评价说“Deep Genomics，一家将深度进修的能量带到基因组学的创业公司”；Gizmag称“Deep Genomics意欲借助深度进修改良基因医疗”；《连线》之前的报道称“呆板智能破译遗传节制”；《科学美国人》说得很玄乎，“我们DNA的某些角落潜伏疾病线索–深度进修之光照亮基因突变鲜为人知的角落”
  - Credit by 中国人工智能网

ChatGPT

AI 中心
IAIC

自然語言處理 (NLP, Natural Language Processing)

NLP is a field of artificial intelligence that focuses on enabling computers to understand, interpret, and generate human language.理解、解釋和生成人類語言
NLP combines linguistics, computer science, and machine learning to create algorithms that can process, analyze, and produce natural language text or speech.



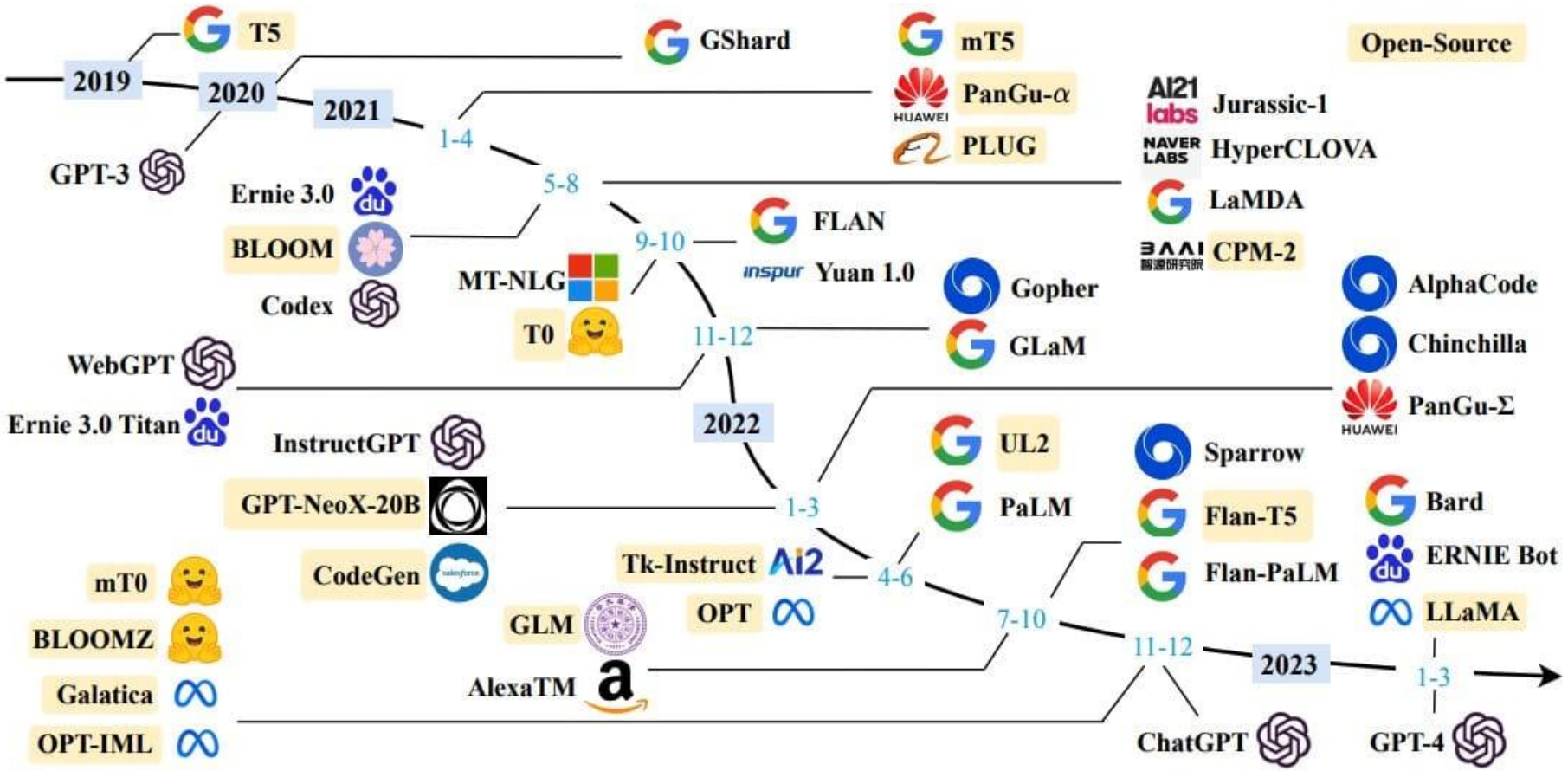
https://www.simplilearn.com/tutorials/artificial-intelligence-tutorial/what-is-natural-language-processing-nlp
https://blog.hlb.im/generative-ai-openai-%E8%A9%9E%E5%BD%99%E5%B0%8D%E7%85%A7%E8%A1%A8-591d2a1f1612

大型語言模型 (Large Language Models)

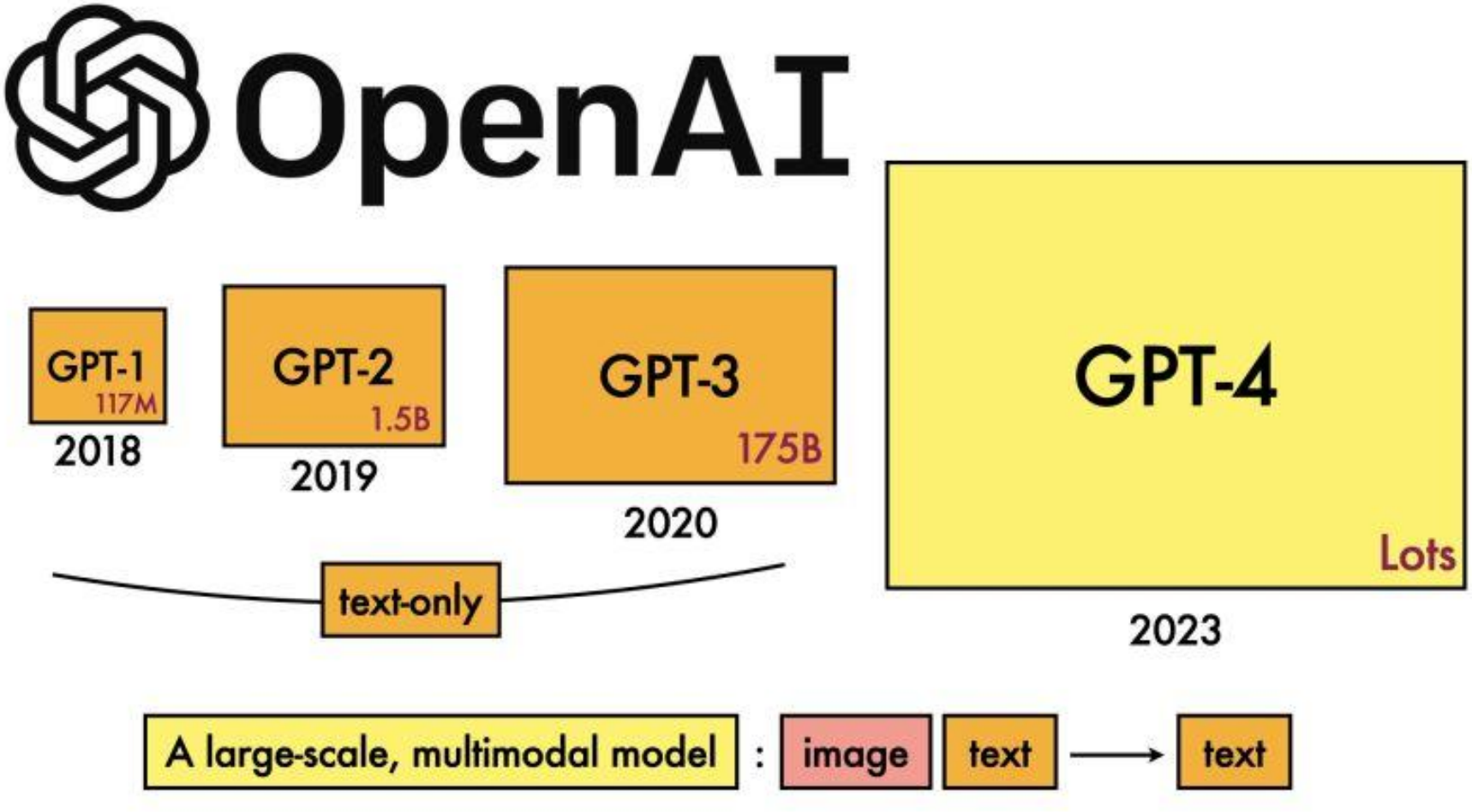
Large language models are artificial intelligence models trained on vast amounts of textual data, enabling them to understand and generate human-like text.理解和生成類似人類的文本
These models can learn intricate patterns, context, and knowledge from the training data, resulting in an impressive ability to generate coherent, contextually relevant text.
學習複雜的模式、上下文和知識，從而具有生成連貫、與上下文相關的文本

Large language models, such as OpenAI’ s GPT series, have demonstrated remarkable performance in various natural language processing tasks, including text completion, summarization, and translation.

https://blog.hlb.im/generative-ai-openai-%E8%A9%9E%E5%BD%99%E5%B0%8D%E7%85%A7%E8%A1%A8-591d2a1f1612



https://www.facebook.com/groups/121499715181411/permalink/1163174821013890/?mibextid=Nif5oz



https://www.facebook.com/100063954998986/posts/pfbid0Hdfk6uqJtCHvEEXpsPtQ6KcZWcWVjYtPr5ePoUYMVSEGEtjq9jLBrFmenb1ShETDl/?mibextid=Nif5oz

Credit-> Kalyan KS

Generative Pre-trained Transformer (GPT)

- 2018 OpenAI 推出了 GPT 模型
  - GPT 模型是一種生成式預訓練轉換器模型，它可以通過大量的文本和程式碼數據進行訓練，生成類似人類的文本。
- GPT 模型的發展可分三個階段：
  - GPT-1： 2018/01，使用 5GB 資料進行訓練、1.17億參數
    - 可以生成類似人類的文本，但其性能仍有待提高。
  - GPT-2：2019/11，使用 1.56TB 資料進行訓練、15億參數
    - GPT-2 的性能比 GPT-1 提高了許多，它可以生成更流暢、更有創意的文本
  - GPT-3：2020/12 ，使用 175TB資料進行訓練、1750億參數
    - GPT-3 的性能是 GPT-2 的數千倍，它可以生成逼真、流暢的文字，並且能夠回答開放式、具有挑戰性的問題。

Brown, Tom, et al. "Language models are few-shot learners." Advances in neural information processing systems 33 (2020): 1877-1901.

- Sizes, architectures, and learning hyper-parameters of the models which we trained

Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{head}	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	6.0×10^{-4}
GPT-3 Medium	350M	24	1024	16	64	0.5M	3.0×10^{-4}
GPT-3 Large	760M	24	1536	16	96	0.5M	2.5×10^{-4}
GPT-3 XL	1.3B	24	2048	24	128	1M	2.0×10^{-4}
GPT-3 2.7B	2.7B	32	2560	32	80	1M	1.6×10^{-4}
GPT-3 6.7B	6.7B	32	4096	32	128	2M	1.2×10^{-4}
GPT-3 13B	13.0B	40	5140	40	128	2M	1.0×10^{-4}
GPT-3 175B or “GPT-3”	175.0B	96	12288	96	128	3.2M	0.6×10^{-4}

- Datasets used to train GPT-3.

Dataset	Quantity (tokens)	Weight in training mix	Epochs elapsed when training for 300B tokens
Common Crawl (filtered)	410 billion	60%	0.44
WebText2	19 billion	22%	2.9
Books1	12 billion	8%	1.9
Books2	55 billion	8%	0.43
Wikipedia	3 billion	3%	3.4

Chat Generative Pre-trained Transformer (ChatGPT)

- 2022/11/30: OpenAI 開發的大型語言模型聊天機器人 ChatGPT
  - 使用 GPT-3.5 模型進行訓練。ChatGPT 可以生成類似人類的文本，並且能夠回答開放式、具有挑戰性的問題。
- 2023/07/20: ChatGPT4，使用 GPT-4 模型進行訓練，1.8萬億參數
  - GPT-4 的性能是 GPT-3 的數百倍，ChatGPT 的性能也得到了大幅提升。

NEWS FEATURE | 25 July 2023

ChatGPT broke the Turing test — the race is on for new ways to assess AI

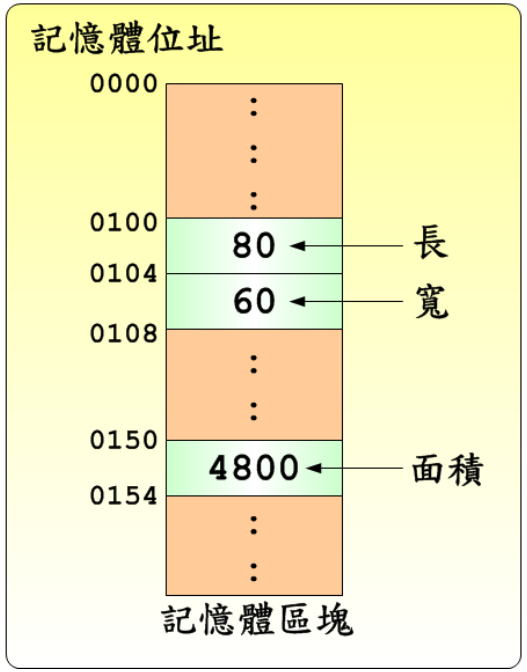
資料型態

資料型態（Data Type）- 變數(variable)

- 程式的運作實際上是依靠著變數的變化而完成的
- 變數代表在程式執行過程中可能會被改變的某個數值。
- 在程式的運作過程中，事實上也是靠眾多變數的變化來完成工作的
- 例如：計算長方形面積的程式，至少須包含3變數：長、寬、面積。如果長與寬都不能改變數值，這個程式將只能解決某一個固定的小問題，例如：只能固定計算長為3、寬為2的長方形面積。
- 因此，若要程式具有較大的彈性解決更多的問題，就必須將長、寬設為可接受使用者輸入數值的變數，由於長與寬可以變動，因此面積也必須是一個可以變動的變數。

資料型態（Data Type）

- 程式的運作主要是靠CPU與記憶體的合作來完成
- 程式中的變數將存放在記憶體（實際運作時可能會被搬移到CPU的暫存器中）。
  - 以上面的範例來說，在記憶體中，就必須儲存長、寬、面積等3個變數如下圖。

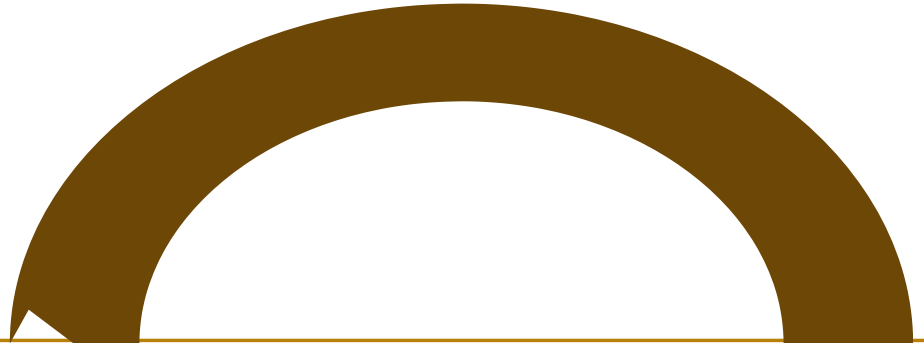


資料型態（Data Type）

- 變數在記憶體中佔用了某一小塊記憶體，程式可以由記憶體位址取得這些變數內容
  - 對於人來說，記憶體位址是非常難以記憶與了解的
  - 所有的高階語言都提供了以名稱來代替變數在記憶體中的位址
  - 換句話說，我們只要給該變數一個名稱，就可以直接透過名稱來取得變數值，而不用煩惱該變數究竟在記憶體中是哪一個位址。

資料型態（Data Type）

- 對於初學程式設計者來說，您只要記住變數名稱實際上將對應到某一個記憶體位址，並且該變數就是用來儲存某項資料即可。
- 3件關於變數的重要事項：
    1. 變數的命名方式。（各語言規範不同）
    2. 宣告變數及設定變數值的方式。（各語言規範不同）
    3. 變數的資料型態種類。



運算子與運算式

- 程式的目的是解決問題，而手段則是依靠各個變數值的改變，想要變更變數值就必須透過運算式加以完成。
- **運算式(Expression)**是由**運算元(Operand)**與**運算子(Operator)**共同組成，常見的數學公式就是最基本的運算式
    - area=r*r*3.14
        - 運算元：r、3.14
        - 運算子: =、*

運算子與運算式

- 每一種程式語言運算子及符號不盡相同
  - + 、- 、* 、/ : 都是屬於最基本且大多相同的運算子。
  - 餘數運算子： Visual Basic = Mod，C/C++ = %
  - 有些語言會提供特殊的運算子，i.e., C/C++及Java: ++ 遞增運算子。
- 運算子的種類大致上可以區分為下列四大類：
  - 設定運算子（Assignment Operator）：可用來設定變數值。
  - 算術運算子（Arithmetic Operator）：可用來進行數值運算，例如+、-
  - 邏輯運算子(Logical Operator)：可用來進行邏輯運算，例如AND、OR。
  - 比較運算子(Comparison Operator)：可用來比較，但不具有設定效果。

Function/Procedure

函式/程序

程序(Procedure)與方法

- 在結構化程式設計中，我們通常會將具有某種特定功能（解決某一子問題）或常常重複出現的敘述區塊獨立出來，成為一個**程式單元**，此時我們會將之賦予一個名稱，以便於其他程式呼叫及使用，此程式單元稱之為**程序(Procedure)**。

程序(Procedure) 優點

1. 程序是**模組化**的一大特色

將一個大的應用程式切割為數個副程式（程序），就可以由許多的程式設計師分工撰寫某些副程式，如此一來，可以加快程式的開發速度，不過在切割功能以及撰寫之前，必須討論出一定的規格，以免發生不協調的狀況。

2. 程序具有**可重複使用性(reusability)**，只要寫好一次，就可以在不同的地方呼叫該程序，而不需要重新撰寫相同的程式碼。
3. 程序最好具有特定功能，並且程序的程式碼應該越簡單越好，如此才能夠提高程式的**可讀性(readability)**以及有利於**除錯**與日後的**維護**。

- 雖然程序具有許多優點，但由於呼叫程序時必須增加一些手續（例如將必要資訊儲存入堆疊中），因此會比直接將程序敘述寫入程式碼中慢一點。

Functions

- Each function contains the following
  - Name
  - Description
  - Parameters
  - Returns

Functions

- triangle.cpp v.s. triangleFun.cpp

```c
#include <stdio.h>

char line[100];/* line of input data */
int  height;   /* the height of the triangle */
int  width;    /*the width of the triangle */
int  area;     /* area of the triangle (computed) */

int main() {
    printf("Enter width height? ");

    fgets(line, sizeof(line), stdin);
    sscanf(line, "%d %d", &width, &height);
    area = (width * height) / 2;
    printf("The area is %d\n", area);

    return (0);
}
```

```c
#include <stdio.h>

char line[100];/* line of input data */
float  height;   /* the height of the triangle */
float  width;    /*the width of the triangle */
float  area;     /* area of the triangle (computed) */

float triangle(float width, float height)
{
    float area;
    area = (width * height) / 2;
    return area;
}

int main() {
    printf("Enter width height? ");
    fgets(line, sizeof(line), stdin);
    sscanf(line, "%f %f", &width, &height);

    area = triangle(width, height);
    printf("The area is %f\n", area);

    return (0);
}
```

Recursive Function

遞迴函式

遞迴函式/程序(recursive function/procedure)

- A function calls itself directly or indirectly
- 一個程序經由直接或間接呼叫程序本身，稱之為程序的『遞迴呼叫』。
  - 直接呼叫 : pro1()呼叫pro1()
  - 間接呼叫 : pro1()呼叫pro2()且pro2()呼叫pro1()

遞迴函式/程序

- 遞歸函數必須遵循兩個基本規則:
  - 它必須有一個終點ending point
  - 它必須使問題變得更簡單
- 可以輕鬆解決一些資訊領域常見的問題（例如：樹狀圖的相關演算法），而且相當簡潔使人易懂，但執行效率則略遜一疇。

常見遞迴程序

- n!計算
- Fibonacci數列（簡稱費氏數列）
- 河內塔

Sum up

- sum(1 8 3 2)

= 1 + sum(8 3 2)

= 1 + 8 + sum(3 2)

= 1 + 8 + 3 + sum(2)

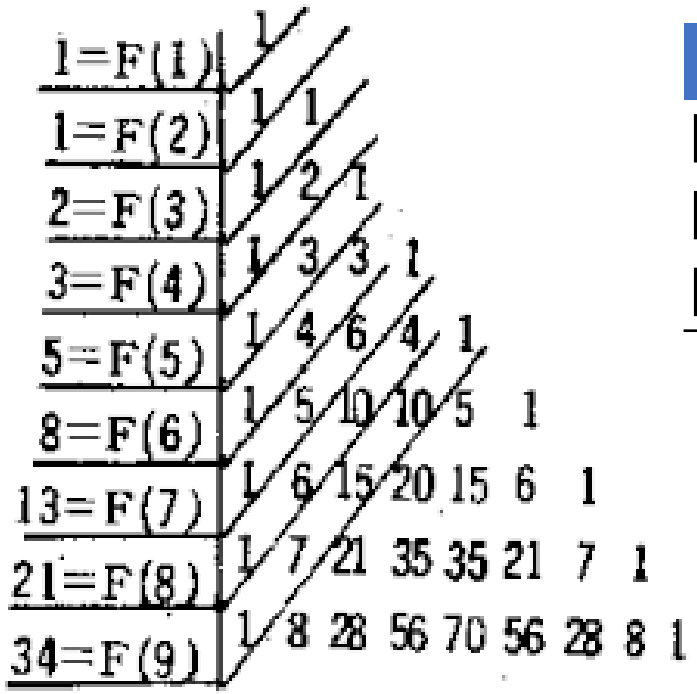
= 1 + 8 + 3 + 2

```
int sum(int first, int last, int array[])
{
    if (first == last)
        return (array[first]);
    /* else */
    return (array[first] + sum(first+1, last, array));
}
```

Factorial => n!

5! = fact(5)
= 5 x fact(4)
= 5 x 4 x fact(3)
= 5 x 4 x 3 x fact(2)
= 5 x 4 x 3 x 2 x fact(1)
= 5 x 4 x 3 x 2 x 1 x fact(0)
= 5 x 4 x 3 x 2 x 1 x 1

Fibonacci (費氏數列)



Credit by http://140.128.107.194/wpmu/thuhsc006/wp-content/blogs.dir/162/files/1/bio.gif

費氏數列的遞迴定義式

$$F(0) = 0 \qquad n = 0$$
$$F(1) = 1 \qquad n = 1$$
$$F(n) = F(n-1)+F(n-2) \quad n \geqq 2$$

Credit 博碩文化 陳錦輝 陳湘揚 探索未來 第八版

Euclidean algorithm (歐幾里得算法,輾轉相除法)

- The greatest common divisor (gcd) of two or more integers = when at least one of them is not zero, is the largest positive integer that is a divisor of both numbers.
- *a*>=*b*, *gcd*(*a*, *b*) =

gcd (b, a % b)

```
int gcd (int a, int b) {
  if (b == 0)
    return a;
  else
    return gcd(b, a % b);
}
```

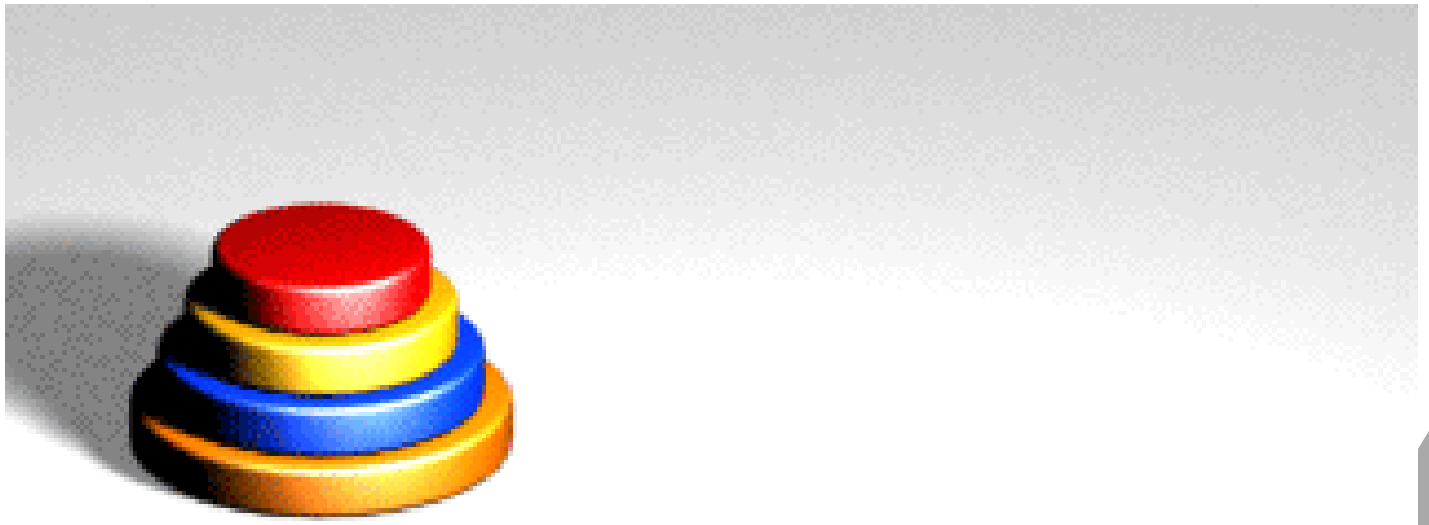
Towers of Hanoi (河內塔)

- The puzzle was invented by the French mathematician *Edouard Lucas* in 1883.
- three towers, and a number of disks of different sizes which can slide onto any tower. The puzzle starts with the disks in a neat stack in ascending order of size on one tower, the smallest at the top. The objective of the puzzle is to move the entire stack to another tower, obeying the following simple rules:
  - Only one disk can be moved at a time.
  - Each move consists of taking the upper disk from one of the stacks and placing it on top of another stack i.e. a disk can only be moved if it is the uppermost disk on a stack.
  - No disk may be placed on top of a smaller disk.

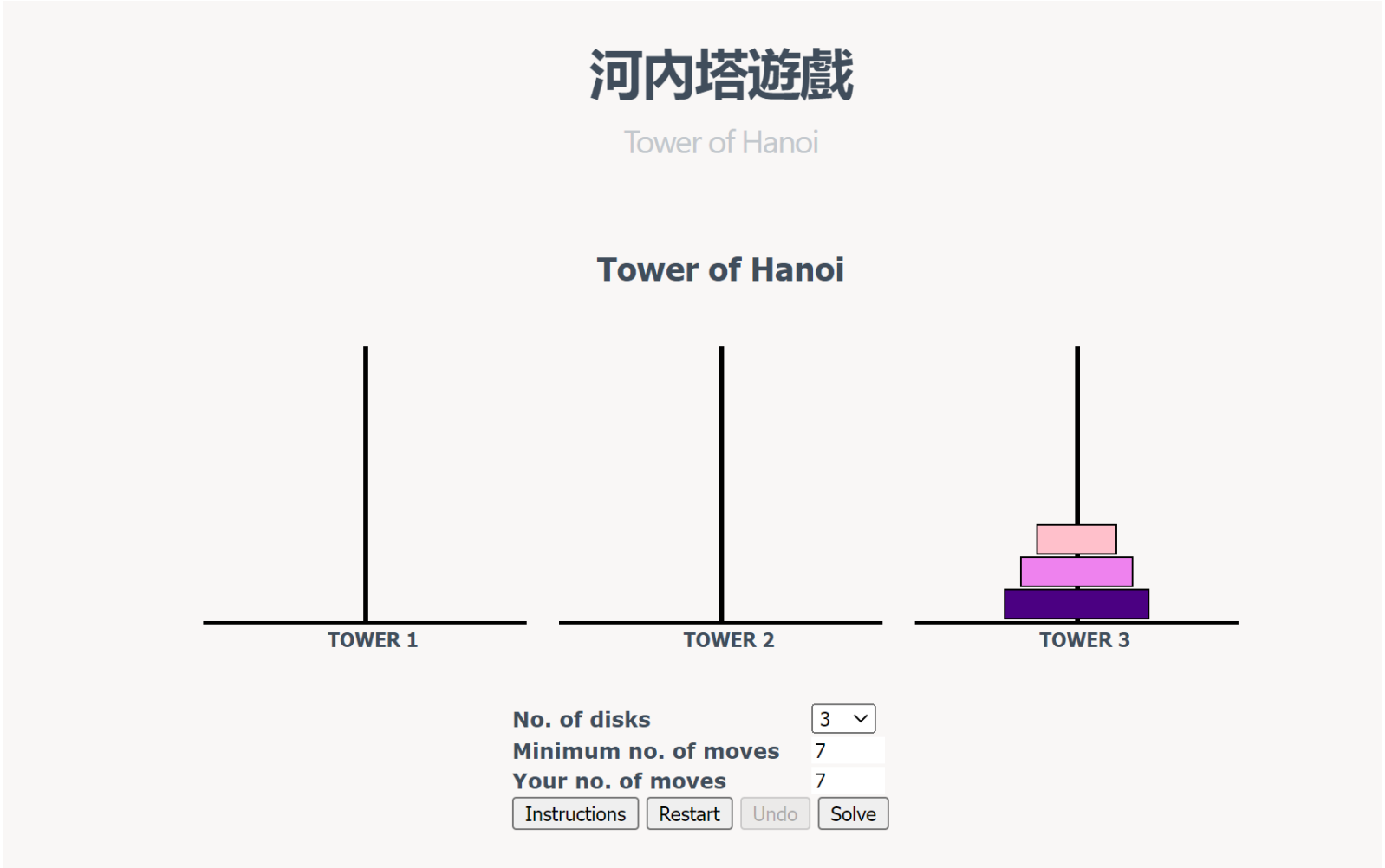
Credit by https://en.wikipedia.org/wiki/Tower_of_Hanoi

河內塔

- Animation with 8 disks
- Animation with 3 disks



Credit by https://en.wikipedia.org/wiki/Tower_of_Hanoi



https://tygtw.ddns.net/game/hanoi/

隨堂練習:自訂disks，將完成結果截圖

河內塔函式

- Number of switches, *f*(*n*), for *n* disks

$$f(n) = \begin{cases} 0, n = 0 \\ 2f(n-1) + 1, n > 0 \end{cases}$$

$$f(n) = \begin{cases} 1, n = 1 \\ 2f(n-1) + 1, n > 1 \end{cases}$$

How to calculate # of switches?

- $f(n) = \begin{cases} 0, n = 0 \\ 2f(n-1)+1, n > 1 \end{cases}$ $f(n) = \begin{cases} 1, n = 1 \\ 2f(n-1)+1, n > 1 \end{cases}$
- i.e, with 8 disks

$f(8) = 2f(7)+1$

$= 2(2f(6)+1)+1 = 2^2f(6)+2+1$

$= 2^2(2f(5)+1)+2+1 = 2^3f(5)+2^2+2+1$

$= 2^3(2f(4)+1)+2^2+2+1 = 2^4f(4)+2^3+2^2+2+1$

$= 2^4(2f(3)+1) +2^3+2^2+2+1 = 2^5f(3)+2^4+2^3+2^2+2+1$

$= 2^5(2f(2)+1)+ 2^4+2^3+2^2+2+1 = 2^6f(2)+2^5+2^4+2^3+2^2+2+1$

$= 2^6(2f(1)+1)+2^5+2^4+2^3+2^2+2+1 = 2^7f(1)+2^6+2^5+2^4+2^3+2^2+2+1$

$= 2^7(2f(0)+1)+2^6+2^5+2^4+2^3+2^2+2+1 = 2^8f(0)+2^7+2^6+2^5+2^4+2^3+2^2+2+1$

$=2^7+2^6+2^5+2^4+2^3+2^2+2+1$

$= 2^8-1$

- $f(n)=2^n-1 = O(2^n)$

$f(n)=2^n-1$ 有多大?

- 如果N=15，最少需移動32767次
  - 如果一個人從 3 歲到 99 歲，每天移動一塊圓盤，他最多僅能移動 15 塊

- 隨堂練習:如果N=64，需要 $2^{64}-1$ 步才能完；若每秒可完成一個盤子的移動，需要 ??年才能完成。

- #整個宇宙現在也不過 137 億年。

Credit by https://en.wikipedia.org/wiki/Tower_of_Hanoi

hanoi.cpp

```
void hanoi(int num_desks, char from, char tmp, char to) {
    if(num_desks == 1) {
        printf("Move sheet from %c to %c\n", from, to);
        num_switch++;
    }
    else {
        hanoi(num_desks-1, from, to, tmp);
        hanoi(1, from, tmp, to);
        hanoi(num_desks-1, tmp, from, to);
    }
}
```

$$f(n) = \begin{cases} 1, n = 1 \\ 2f(n-1) + 1, n > 0 \end{cases}$$

```
A = [3, 2, 1]
B = []
C = []
def move(n, source, target, auxiliary):
    if n > 0:
        # move n - 1 disks from source to auxiliary, so they are out of the way
        move(n - 1, source, auxiliary, target)
        # move the nth disk from source to target
        target.append(source.pop())
        # Display our progress
        print(A, B, C, '###############', sep = '\n')
        # move the n - 1 disks that we left on auxiliary onto target
        move(n - 1, auxiliary, target, source)
# initiate call from source A to target C with auxiliary B
move(3, A, C, B)
```


Credit by https://en.wikipedia.org/wiki/Tower_of_Hanoi

the *Reve's puzzle*

- If there are **four** towers instead of **three** towers, $g(n)$ denotes the minimum number of moves needed for n disks for four towers.
- What is $g(n)$ in *recurrence relation* form?
  - $g(n)$ should contain $f(n)$

Credit by https://en.wikipedia.org/wiki/Tower_of_Hanoi#With_four_pegs_and_beyond

The Frame–Stewart algorithm

Giving an optimal for four (and conjecturally for even more) pegs, is described below:

- Let *n* be the number of disks.
- Let *r* be the number of pegs.

- Define ***T*(*n*, *r*)** to be the minimum number of moves required to transfer ***n* disks** using ***r* pegs**
  - *Stewart, B. M.; Frame, J. S. (March 1941). "Solution to advanced problem 3819". American Mathematical Monthly.* ***48*** *(3): 216–9. doi:10.2307/2304268. JSTOR 2304268.*

Credit by https://en.wikipedia.org/wiki/Tower_of_Hanoi#With_four_pegs_and_beyond

The Frame–Stewart algorithm for $T(n, r)$

1. For some k, $1 \leq k < n$, transfer the top k disks to a single peg other than the start or destination pegs
   => $T(k, r)$ moves.
2. Without disturbing the peg that now contains the top k disks, transfer the remaining $n-k$ disks to the destination peg, using only the remaining $r-1$ pegs => $T(n-k,r-1)$ moves.
3. Finally, transfer the top k disks to the destination peg => $T(k, r)$moves.

$$T(n,r) = 2T(k,r) + T(n-k,r-1)$$

Credit by https://en.wikipedia.org/wiki/Tower_of_Hanoi#With_four_pegs_and_beyond

the *Reve's puzzle*

- If there are four towers instead of three towers, $g(n)$ denotes the minimum number of moves needed for n disks for four towers.
- What is $g(n)$ in *recurrence relation* form?
  - $g(n)$ should contain $f(n)$
  - $T(n,r) = 2T(k,r) + T(n-k, r-1)$
  - $\mathrm{g}(n) = 2g(k) + f(n-k), \text{where } 1 \leq k < n$

Credit by https://en.wikipedia.org/wiki/Tower_of_Hanoi#With_four_pegs_and_beyond

Reve's puzzle

- Although the three-peg version has a simple recursive solution as outlined above which has long been known, the *optimal* solution for the Tower of Hanoi problem with four pegs (called **Reve's puzzle**) was not verified until 2014, and in the case of more than four pegs the optimal solution is still an open problem, even if a proof that the Frame–Stewart algorithm is optimal was proposed by Demontis in 2016.

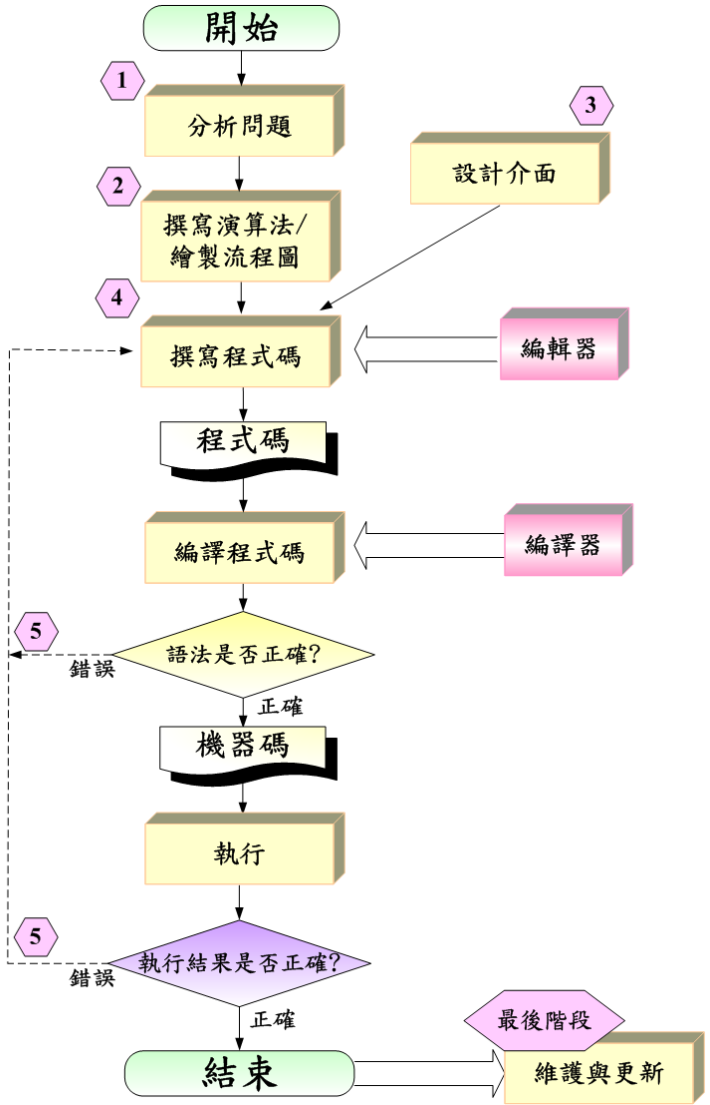
Credit by https://en.wikipedia.org/wiki/Tower_of_Hanoi#With_four_pegs_and_beyond

程式設計流程

程式的設計流程

- 開發程式是為了解決人類的問題，因此，每一個程式的細節都會依照問題而有或多或少的不同。
- 但大多數程式開發的流程卻是相同的
- 以一個視窗應用程式而言其開發流程分為五個階段如下。
  - 第一階段：分析問題階段（Problem Analysis）
  - 第二階段：設計演算法及流程圖
  - 第三階段：設計使用者介面(User Interface Design)
  - 第四階段：撰寫程式碼(Coding)
  - 第五階段：驗證程式正確性(Verification)
  - 最後階段：維護與更新

程式的設計流程

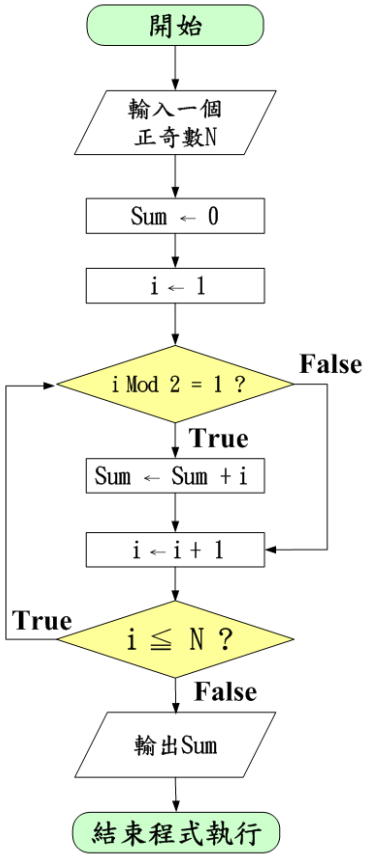


Credit 博碩文化 陳錦輝 陳湘揚 探索未來 第八版

虛擬碼(pseudo code)

- 虛擬語言(Pseudo-Language)是演算法的描述語言
  輸入（Input）：輸入一個正奇數N
  輸出（Output）：輸出Sum，Sum= 1 + 3 + 5 + … + N

  解決問題的步驟：
  Step 1：輸入一個正奇數N
  Step 2：令Sum = 0
  Step 3：令 i = 1
  Step 4：若 i Mod 2 = 1則執行Step5，否則跳至Step 6。
  Step 5：令Sum = Sum + i
  Step 6：令 i = i + 1
  Step 7：若 i ≦ N，則跳回到Step 4執行，否則執行Step 8。
  Step 8：輸出Sum
  Step 9：結束程式執行



整合開發環境 Integrated Development Environment (IDE)

- 傳統使用編譯器開發程式的流程有些繁複，後來為了方便程式設計師設計程式，因此發展了許多IDE，原本整合開發環境不應該稱之為程式語言
  - 例如Delphi指的是Object Pascal程式語言的視窗整合開發環境，也就是說，Delphi並非一種程式語言。
- 但隨著程式語言由廠商推出（而非純學術用途或標準），因此有些整合開發環境與程式語言同名，例如Visual Basic .NET。

Alan Mathison Turing Award

(2021)
Dongarra, Jack

(2020)
Aho, Alfred Vaino
Ullman, Jeffrey David

(2019)
Catmull, Edwin E.
Hanrahan, Patrick M.

(2018)
Bengio, Yoshua
Hinton, Geoffrey E
LeCun, Yann

(2017)
Hennessy, John L
Patterson, David

(2016)
Berners-Lee, Tim

(2015)
Diffie, Whitfield
Hellman, Martin

(2014)
Stonebraker, Michael

(2013)
Lamport, Leslie

(2012)
Goldwasser, Shafi
Micali, Silvio

(2011)
Pearl, Judea

(2010)
Valiant, Leslie Gabriel

(2009)
Thacker, Charles P. (Chuck) *

(2008)
Liskov, Barbara

(2007)
Clarke, Edmund Melson *
Emerson, E. Allen
Sifakis, Joseph

(2006)
Allen, Frances ("Fran") Elizabeth *

(2005)
Naur, Peter *

(2004)
Cerf, Vinton (“Vint”) Gray
Kahn, Robert (“Bob”) Elliot

(2003)
Kay, Alan

(2002)
Adleman, Leonard (Len) Max
Rivest, Ronald (Ron) Linn
Shamir, Adi

(2001)
Dahl, Ole-Johan *
Nygaard, Kristen *

(2000)
Yao, Andrew Chi-Chih

(1999)
Brooks, Frederick ("Fred")

(1998)
Gray, James ("Jim") Nicholas *

(1997)
Engelbart, Douglas *

(1996)
Pnueli, Amir *

(1995)
Blum, Manuel

(1994)
Feigenbaum, Edward A ("Ed")
Reddy, Dabbala Rajagopal ("Raj")

(1993)
Hartmanis, Juris *
Stearns, Richard ("Dick") Edwin

(1992)
Lampson, Butler W

(1991)
Milner, Arthur John Robin Gorell ("Robin") *

(1990)
Corbato, Fernando J ("Corby") *

(1989)
Kahan, William (“Velvel”) Morton

(1988)
Sutherland, Ivan

(1987)
Cocke, John *

(1986)
Hopcroft, John E
Tarjan, Robert (Bob) Endre

(1985)
Karp, Richard ("Dick") Manning

(1984)
Wirth, Niklaus E

(1983)
Ritchie, Dennis M.*
Thompson, Kenneth Lane

(1982)
Cook, Stephen Arthur

(1981)
Codd, Edgar F. ("Ted") *

(1980)
Hoare, C. Antony ("Tony") R.

(1979)
Iverson, Kenneth E. ("Ken") *

(1978)
Floyd, Robert (Bob) W *

(1977)
Backus, John *

(1976)
Rabin, Michael O.
Scott, Dana Stewart

(1975)
Newell, Allen *
Simon, Herbert ("Herb") Alexander *

(1974)
Knuth, Donald ("Don") Ervin

(1973)
Bachman, Charles William *

(1972)
Dijkstra, Edsger Wybe *

(1971)
McCarthy, John *

(1970)
Wilkinson, James Hardy ("Jim") *

(1969)
Minsky, Marvin *

(1968)
Hamming, Richard W*

(1967)
Wilkes, Maurice V.*

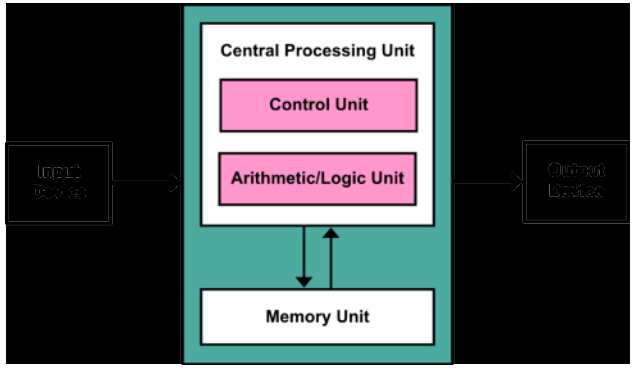
(1966)
Perlis, Alan J *

電腦之父 - John von Neumann

1903.12.28~ 1957.2.8

https://en.wikipedia.org/wiki/John_von_Neumann#/media/File:JohnvonNeumann-LosAlamos.gif

Von Neumann architecture



二進位

1 Bit = 0/1
1 Byte = 8 Bits
1 Kilobyte (KB) = 1024 Bytes
1 Megabyte (MB) = 1024 KB
1 Gigabyte (GB) = 1024 MB
1 Terabyte (TB) = 1024 GB
1 Petabyte (PB) = 1024 TB

出處:
http://city.udn.com/66782/4868792#ixzz413hkUP2n

演算法之父

- 比爾·蓋茲在1995年說 “如果你認為你是一名真正優秀的程式員，就去讀第一卷，確定可以解決其中所有的問題，如果你能讀懂整套書的話，請給我發一份你的簡歷。”
- Volume 1 (1968)~ 4 (2015) Volume 5, Syntactic Algorithms, in preparation. estimated to be ready in 2025.

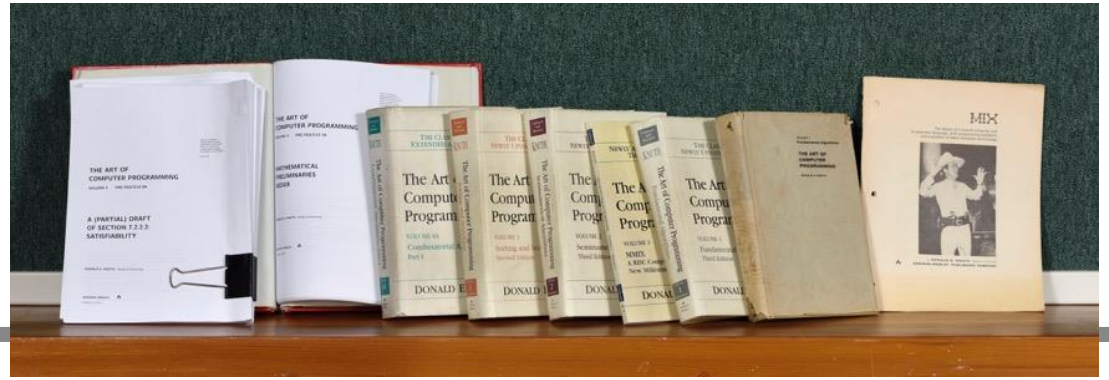
DONALD ("DON") ERVIN KNUTH

United States – 1974

CITATION

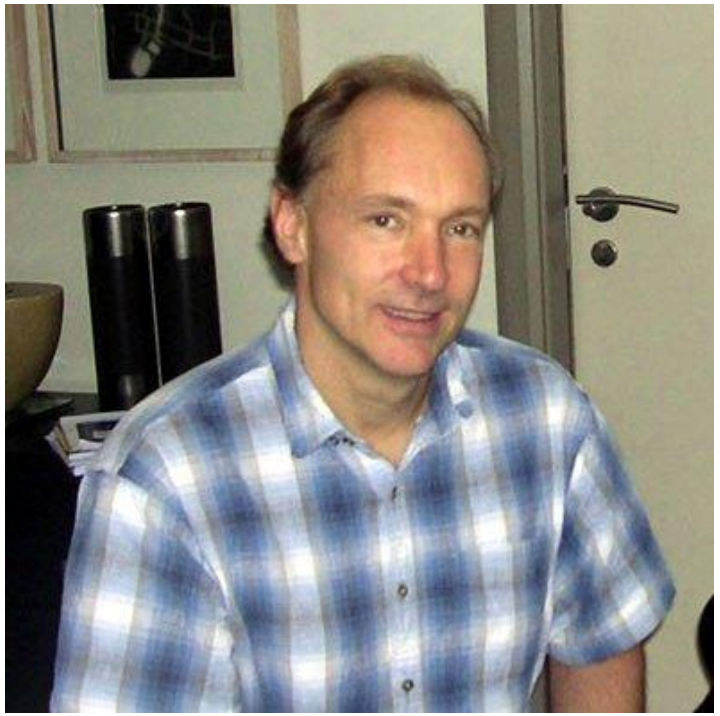
For his major contributions to the analysis of algorithms and the design of programming languages, and in particular for his contributions to the "art of computer programming" through his well-known books in a continuous series by this title.

BIRTH: January 10, 1938, in Milwaukee, Wisconsin.



http://www-cs-faculty.stanford.edu/~uno/taocp.html

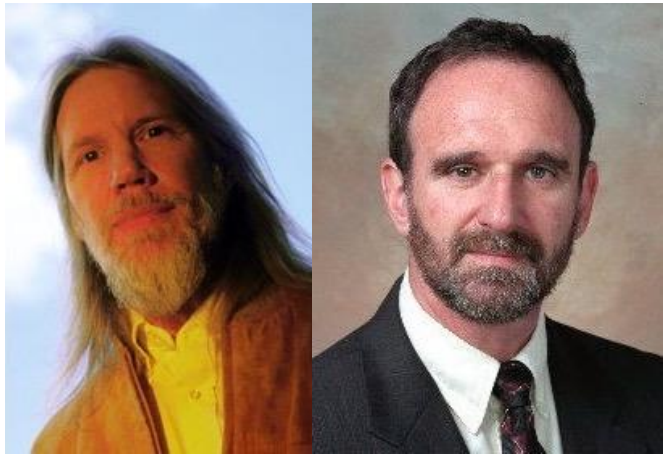
全球資訊網發明者 - Tim Berners-Lee



- 1955.6.8~
- 我只要把超文字系統和傳輸控制協定、網域名稱系統結合在一起，就能得出全球資訊網了

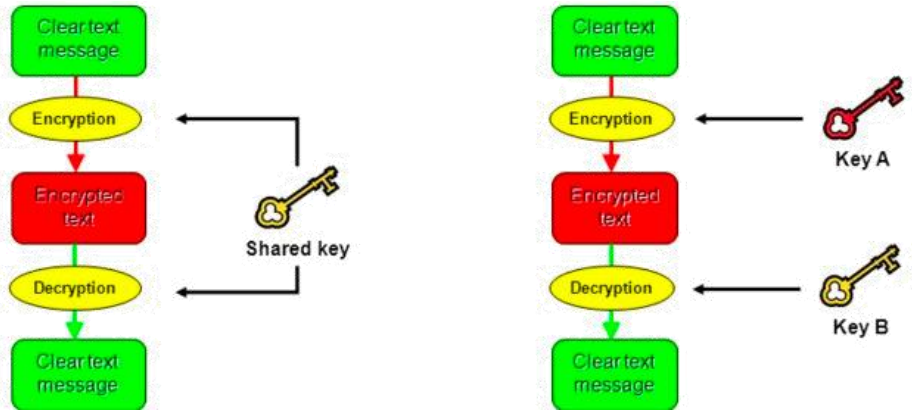
  - By wikipedia
  - https://www.w3.org/People/Berners-Lee/Kids.html

公開金鑰加密



Whitfield Diffie　　Martin Hellman

http://blogs.getcertifiedgetahead.com/symmetric-and-asymmetric-encryption/



http://www.myshared.ru/slide/397766/

RSA加密演算法

RONALD (RON) LINN RIVEST

United States – 2002

CITATION

Together with Leonard M. Adleman and Adi Shamir, for their ingenious contribution to making public-key cryptography useful in practice.

BIRTH:
1947, Schenectady, New York, USA

ADI SHAMIR

Israel – 2002

CITATION

Together with Leonard M. Adleman and Ronald Rivest, for their ingenious contribution to making public-key cryptography useful in practice.

BIRTH:
July 6, 1952, Tel Aviv, Israel

LEONARD (LEN) MAX ADLEMAN

United States – 2002

CITATION

Together with Ronald Rivest and Adi Shamir, for their ingenious contribution to making public-key cryptography useful in practice.

BIRTH:
December 31, 1945 in San Francisco, California

人工智慧

MARVIN MINSKY

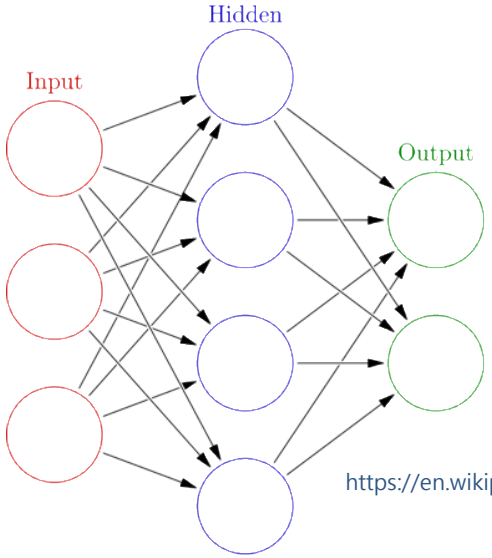
United States – 1969

CITATION

For his central role in creating, shaping, promoting, and advancing the field of Artificial Intelligence.

SHORT ANNOTATED BIBLIOGRAPHY | ACM DL AUTHOR PROFILE | ACM TURING AWARD LECTURE | RESEARCH SUBJECTS | ADDITIONAL MATERIALS

Artificial Neural Network



https://en.wikipedia.org/wiki/Artificial_neural_network

2000 ACM A.M. Turing Award

BIRTH:

1946, Shanghai, China

EDUCATION:

B.S. (Physics, National University of Taiwan, 1967); A.M. (Physics, Harvard University, 1969); Ph.D. (Physics, Harvard University, 1972); Ph.D. (Computer Science, University of Illinois Urbana-Champaign, 1975).

EXPERIENCE:

ANDREW CHI-CHIH YAO

China – 2000

CITATION

In recognition of his fundamental contributions to the theory of computation, including the complexity-based theory of pseudorandom number generation, cryptography, and communication complexity.

SHORT ANNOTATED BIBLIOGRAPHY

RESEARCH SUBJECTS

Andrew Chi-Chih Yao was born in Shanghai, China, on December 24, 1946. After moving with his family to Hong Kong for two years he immigrated to Taiwan. In 1967 he received a B.S. in Physics from the National University of Taiwan. He then started graduate studies in Physics at Harvard University, where he received an A.M. in 1969 and a Ph.D. in 1972 under the supervision of Sheldon Glashow, winner of the 1979 Nobel Prize in Physics. He subsequently entered the Ph.D. program in Computer Science at the University of Illinois Urbana-Champaign, and received his degree just two years later, in 1975. Yao completed his dissertation, *A Study of Concrete Computational Complexity,* under the supervision of Chung Laung Liu.

Credit by https://amturing.acm.org/award_winners/yao_1611524.cfm

2018 ACM A.M. Turing Award

- Yoshua Bengio, Geoffrey Hinton, and Yann LeCun
- https://www.youtube.com/watch?v=Fn589zeMij4
- https://buzzorange.com/techorange/2019/03/28/2019-turing-award/

Credit by https://www.washingtonpost.com/technology/2019/03/27/artificial-intelligence-pioneers-win-turing-award/?noredirect=on

2019 ACM A.M. Turing Award

- PIONEERS OF MODERN COMPUTER GRAPHICS RECOGNIZED WITH ACM A.M. TURING AWARD
- Hanrahan and Catmull's Innovations Paved the Way for Today's 3-D Animated Films
- Turing Award for fundamental contributions to 3-D computer graphics, and the revolutionary impact of these techniques on computer-generated imagery (CGI) in filmmaking and other applications. Catmull is a computer scientist and former president of Pixar and Disney Animation Studios. Hanrahan, a founding employee at Pixar, is a professor in the Computer Graphics Laboratory at Stanford University.

Edwin E. Catmull

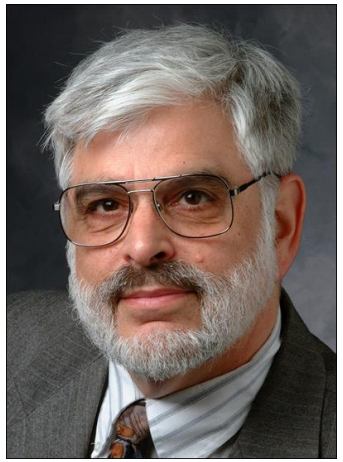
Patrick M. Hanrahan

Credit by https://amturing.acm.org/

2020 ACM A.M. Turing Award

- ACM Turing Award Honors Innovators Who Shaped the Foundations of Programming Language Compilers and Algorithms
- Columbia's Aho and Stanford's Ullman Developed Tools and Fundamental Textbooks Used by Millions of Software Programmers around the World
- Influential Textbooks
  - The Design and Analysis of Computer Algorithms (1974)
  - Principles of Compiler Design (1977)

Alfred Vaino Aho



Jeffrey David Ullman

Credit by https://amturing.acm.org/

2021 Turing Award – high performance computing

- ACM Turing Award Honors Jack Dongarra for Pioneering Concepts and Methods Which Resulted in World-Changing Computations
- Dongarra's Algorithms and Software Fueled the Growth of High-Performance Computing and Had Significant Impacts in Many Areas of Computational Science from AI to Computer Graphics
- Dongarra has led the world of high-performance computing through his contributions to efficient numerical algorithms for linear algebra operations, parallel computing programming mechanisms, and performance evaluation tools. For nearly forty years, Moore's Law produced exponential growth in hardware performance. During that same time, while most software failed to keep pace with these hardware advances, high performance numerical software did—in large part due to Dongarra's algorithms, optimization techniques, and production-quality software implementations.

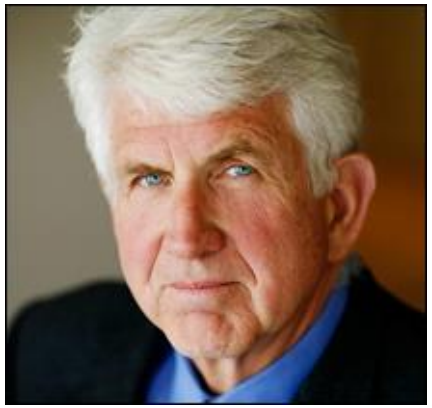
Jack J. Dongarra

Credit by https://amturing.acm.org/

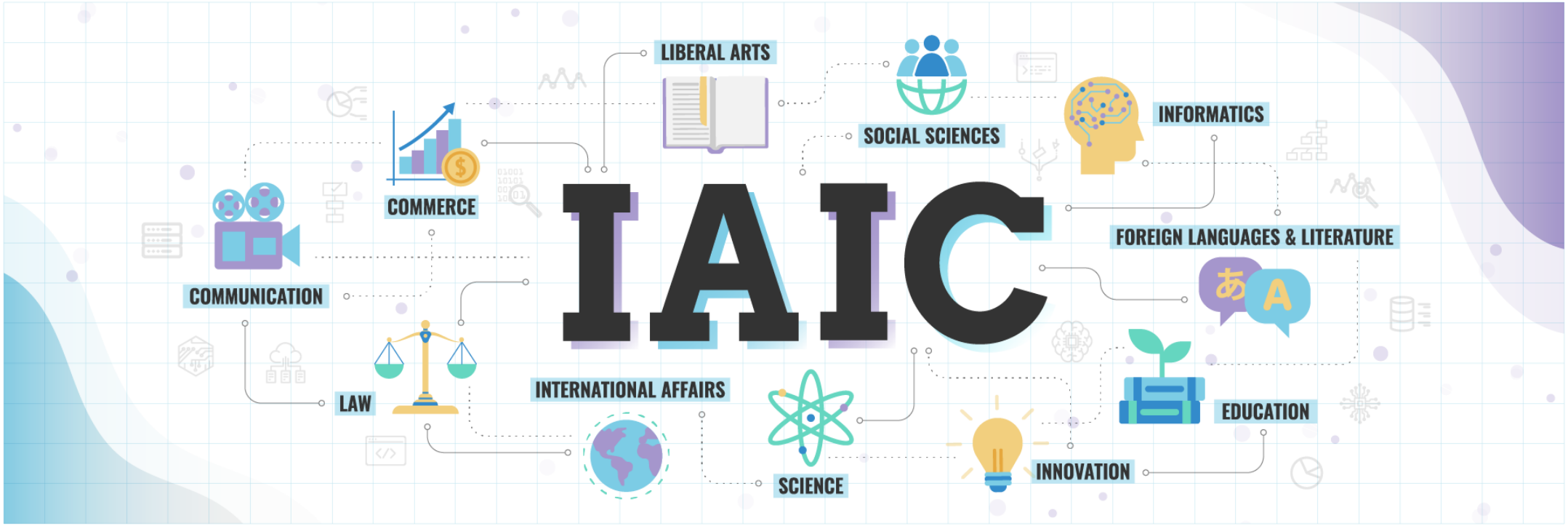
2022 Turing Award - Invention of The Ethernet

- ACM named Bob Metcalfe as recipient of the 2022 Turing Award for the invention, standardization, and commercialization of Ethernet.
- Technology Developed 50 Years Ago Remains the Dominant Way of Connecting Computers and Billions of Other Devices to Each Other and the Internet

Robert Melancton Metcalfe



Credit by https://amturing.acm.org/



研究合作 跨域教學 多元服務

NATIONAL CHENGCHI UNIVERSITY

人工智慧
跨域研究中心
Interdisciplinary Artificial Intelligence Center

Dr. Chih-Hsun Wu
吳致勳 助理教授
20031214@nccu.edu.tyw
j20031214@gmail.com