期中筆記

爬山演算法：

爬山演算法是一種局部擇優的方法，採用啟發式方法，是對深度優先搜尋的一種改進，它利用回饋資訊幫助生成解的決策。

爬山演算法一般存在以下問題：

1. 局部最大
2. 高地：也稱為平頂，搜尋一旦到達高地，就無法確定搜尋最佳方向，會產生隨機走動，使得搜尋效率降低。
3. 山脊：搜尋可能會在山脊的兩面來回震盪，前進步伐很小。

參考資料： https://zh.wikipedia.org/wiki/%E7%88%AC%E5%B1%B1%E7%AE%97%E6%B3%95

遺傳演算法：

基因演算法是計算數學中用於解決最佳化的搜尋演算法，是進化演算法的一種。進化演算法最初是借鑑了進化生物學中的一些現象而發展起來的，這些現象包括遺傳、突變、自然選擇以及雜交等等。

基因演算法通常實現方式為一種電腦模擬。對於一個最佳化問題，一定數量的候選解（稱為個體）可抽象表示為染色體，使種群向更好的解進化。傳統上，解用二進位表示（即0和1的串），但也可以用其他表示方法。進化從完全隨機個體的種群開始，之後一代一代發生。在每一代中評價整個種群的適應度，從當前種群中隨機地選擇多個個體（基於它們的適應度），通過自然選擇和突變產生新的生命種群，該種群在演算法的下一次疊代中成為當前種群。

參考資料：

https://zh.wikipedia.org/wiki/%E9%81%97%E4%BC%A0%E7%AE%97%E6%B3%95

八皇后問題：

八皇后問題是一個以西洋棋為背景的問題：如何能夠在8×8的西洋棋棋盤上放置八個皇后，使得任何一個皇后都無法直接吃掉其他的皇后？為了達到此目的，任兩個皇后都不能處於同一條橫行、縱行或斜線上。八皇后問題可以推廣為更一般的n皇后擺放問題：這時棋盤的大小變為*n*×*n*，而皇后個數也變成*n*。若且唯若*n* = 1或*n* ≥ 4時問題有解。

解法：

八個皇后在8x8棋盤上共有4,426,165,368（64C8）種擺放方法，但只有92個互不相同的解。如果將旋轉和對稱的解歸為一種的話，則一共有12個獨立解。

參考資料：

https://zh.wikipedia.org/wiki/%E5%85%AB%E7%9A%87%E5%90%8E%E9%97%AE%E9%A2%98

代數：

代數的研究對象不僅是數字，還有各種抽象化的結構。例如整數集作為一個帶有加法、乘法和序關係的集合就是一個代數結構。在其中我們只關心各種關係及其性質，而對於「數本身是甚麼」這樣的問題並不關心。常見的代數結構類型有群、環、域、模、線性空間等。

參考資料：

https://zh.wikipedia.org/wiki/%E4%BB%A3%E6%95%B0

幾何：

幾何學簡稱幾何。幾何學是數學的一個基礎分支，主要研究形狀、大小、圖形的相對位置等空間區域關係以及空間形式的度量。

幾何學可見的特性讓它比代數、數論等數學領域更容易讓人接觸，不過一些幾何語言已經和原來傳統的、歐幾里得幾何下的定義越差越遠，例如碎形幾何及解析幾何等。

現代概念上的幾何其抽象程度和一般化程度大幅提高，並與分析、抽象代數和拓撲學緊密結合。

幾何學應用於許多領域，包括藝術，建築，物理和其他數學領域。

參考資料：

https://zh.wikipedia.org/wiki/%E5%87%A0%E4%BD%95%E5%AD%A6

分析：

分析是將複雜的話題或事物逐漸拆分的過程，以此來達到對話題更好的理解。儘管「分析」作為一個正式的概念在近年來才逐步建立起來，這一技巧自亞里士多德就已經應用在了數學、邏輯學等多個領域。

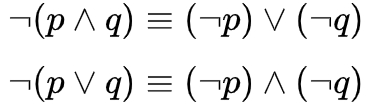
參考資料：

https://zh.wikipedia.org/wiki/%E5%88%86%E6%9E%90

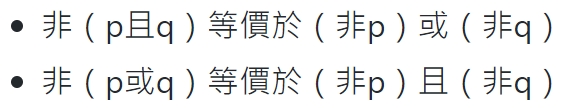
迪摩根定律：

在命題邏輯和邏輯代數中，迪摩根定律是關於命題邏輯規律的一對法則。

19世紀英國數學家奧古斯塔斯·德摩根首先發現了在命題邏輯中存在著下面這些關係：



即：



德摩根定律在數理邏輯的定理推演中，在電腦的邏輯設計中以及數學的集合運算中都起著重要的作用。他的發現影響了喬治·布爾從事的邏輯問題代數解法的研究，這鞏固了德摩根作為該規律的發現者的地位，亞里斯多德亦曾注意到類似的現象、且這也為古希臘與中世紀的邏輯學家熟知。

參考資料：

https://zh.wikipedia.org/wiki/%E5%BE%B7%E6%91%A9%E6%A0%B9%E5%AE%9A%E5%BE%8B

傅立葉轉換：

傅立葉轉換是一種線性積分轉換，用於信號在時域（或空域）和頻域之間的轉換，在物理學和工程學中有許多應用。因其基本思想首先由法國學者約瑟夫·傅立葉系統地提出，所以以其名字來命名以示紀念。實際上傅立葉轉換就像化學分析，確定物質的基本成分；信號來自自然界，也可對其進行分析，確定其基本成分。

參考資料：

https://zh.wikipedia.org/wiki/%E5%82%85%E9%87%8C%E5%8F%B6%E5%8F%98%E6%8D%A2

NumPy：

NumPy是Python語言的一個擴充程式庫。支援高階大量的維度陣列與矩陣運算，此外也針對陣列運算提供大量的數學函式函式庫。

參考資料：

https://zh.wikipedia.org/wiki/NumPy

SymPy：

SymPy是一個符號計算的Python庫。它的目標是成為一個全功能的計算機代數系統，同時保持代碼簡潔、易於理解和擴展。它完全由Python寫成，不依賴於外部庫。

參考資料：

https://zh.wikipedia.org/zh-tw/SymPy

反向傳遞演算法：

反向傳播是「誤差反向傳播」的簡稱，是一種與最優化方法（如梯度下降法）結合使用的，用來訓練人工神經網絡的常見方法。該方法對網絡中所有權重計算損失函數的梯度。這個梯度會反饋給最優化方法，用來更新權值以最小化損失函數。

參考資料：

https://zh.wikipedia.org/wiki/%E5%8F%8D%E5%90%91%E4%BC%A0%E6%92%AD%E7%AE%97%E6%B3%95

Pytorch：

Pytorch是一個開源的Python機器學習庫，基於Torch，底層由C++實現，應用於人工智慧領域，如自然語言處理。它最初由Facebook的人工智慧研究團隊開發，並且被用於Uber的概率編程軟體Pyro。

PyTorch主要有兩大特徵：

* 類似於NumPy的張量計算，可使用GPU加速；
* 基於帶自動微分系統的深度神經網路。

參考資料：

https://zh.wikipedia.org/wiki/PyTorch

循環神經網路：

循環神經網路是神經網路的一種。單純的RNN因為無法處理隨著遞歸，權重指數級爆炸或梯度消失問題，難以捕捉長期時間關聯；而結合不同的LSTM可以很好解決這個問題。

時間循環神經網路可以描述動態時間行為，因為和前饋神經網路（feedforward neural network）接受較特定結構的輸入不同，RNN將狀態在自身網路中循環傳遞，因此可以接受更廣泛的時間序列結構輸入。手寫識別是最早成功利用RNN的研究結果。

參考資料：

https://zh.wikipedia.org/wiki/%E5%BE%AA%E7%8E%AF%E7%A5%9E%E7%BB%8F%E7%BD%91%E7%BB%9C

LSTM:

LSTM是一種時間循環神經網路（RNN），論文首次發表於1997年。由於獨特的設計結構，LSTM適合於處理和預測時間序列中間隔和延遲非常長的重要事件。

LSTM的表現通常比時間循環神經網路及隱馬爾科夫模型（HMM）更好，比如用在不分段連續手寫辨識上。

參考資料:

https://zh.wikipedia.org/wiki/%E9%95%B7%E7%9F%AD%E6%9C%9F%E8%A8%98%E6%86%B6