Introduction to algorithm(Part 1)

by

Triton Ho 

今天大綱

- Nearest Neighbour
- K-th item in array

Nearest Neighbour

Nearest point 背景

- 在 Python 版，有人發問：
  - 我有 10000 個點，我想找出每一個點的最近鄰居
  - 然後，一堆人回答：
    R-tree, Quadtree.....

我的回應

- 如果才 10000 個點
  單純 2 層 for-loop ， O(n^2) solution 吧
  - 現實跑起來應該少於 1 秒
- 如果 n > 1M 則肯定不能暴力解

我想說……

- Bruteforce algorithm （暴力解法）也是算法的一種啊！
  - 特別在 n 很小的時間
- Engineering 是
  - 在有限資源（開發時間／ server cost ／效能）下找出最可行解法

我繼續想說……

- N 很小時， O(n^2) ／ O(n^3) 也好
  1ms 和 0.1ms 對用戶感覺是沒差別的
- 越深的 algorithm ，很多時其 implementation 就越複雜，這代表：
  - 更高的開發時間
  - 更多可以犯錯的細節
  - 更多的 source code ，增加維護困難度

羅馬不是一天建成的

- 好的開發者，應該：
  - 先正確地理解問題（請看後頁）
  - 了解 N （像是 QPS / user count / data size ）
  - 給 PM 說明不同解法的開發時間和其預測界限
    找出平衡的方案
  - 開發後，清楚寫下你用了那種解法
  - 管理系統中的 tech debt ，在 N 變大時換上更好解法

正確地理解問題

- 原問題：
  - 我有 10000 個點，我想找出每一個點的最近鄰居
    暴力解法： O(n^2)
    Quadtree 解法： O (n log n)
  - 反正你只用解一次，暴力解法跑完後把結果存起來就好
- 相似問題：
  - 對一個任意點 x ，我想知道在 10000 個點中，誰跟 x 最接近
    暴力解法： O(m*n), m ＝你 query 的次數
    Quadtree 解法： O(m * log n)
  - 因為 m 可以很大，所以暴力解法不一定能接受

Quadtree

- 傳統 quadtree 是每次把 2 維空間切成 4 份
  直到 leave node 內的物件少於上限
  https://en.wikipedia.org/wiki/Quadtree
- 但是，你也可以每一次把空間切得更細碎

Geohash

- https://en.wikipedia.org/wiki/Geohash
- 簡單來說：這是把地球坐標轉成 string 的 encoding
- 4byte geohash 誤差是 20km
  5byte geohash 誤差是 2.4km
  6byte geohash 誤差是 610m
  ……
  （請自己看 wiki 詳情）

Geohash （續）

- 你可以想像：
  每一個 geohash 都是一個長方形
- 在一個長方形內的點，其最近鄰居一定是……
  - 在同一長方形內，或在鄰近八個長方形內
- 對於一個 geohash ，要找出鄰近 8 個長方形的所需時間是：
  - O(1) ，真的是 O(1)

具體解法（簡化版）

- 把每一個點，建立從 1byte 到 i byte 的 geohash 的 column
  然後把這些資料放進 elastic search
- 對於任意點 x
  - 先建立 i byte 的 geohash （還有其相鄰 8 格的 geohash）
  - 把這 9 個 geohash 丟到 ES 搜尋
  - 如果有結果返回，則一個一個對比
    最近的點就是答案
  - 否則用 i-1 byte 的 geohash 從來～

解法優點

- 這是 quadtree 變種，所以其時間是 O(m log n)
  m = query 次數
- 因為其 fanout factor 很高，所以 tree height 很低
- Geohash 有現有 library ，不用自己來寫
- 能善用 elastic search / 一般 RDBMS 的 indexing ，不用自己建立和管理 quadtree

如果你用 postgresql

- 使用 postGIS 套件，以上的都不用自己來寫，有人替你寫好了
  - https://postgis.net/workshops/postgis-intro/knn.html
- 謎之聲：……

額外例子
K-th item in array

K-th item in array

- 有在練 leetcode 的朋友一定會答：quick search algorithm
- 實戰上，如果 n < 1M ，我會……
  - 用 standand library 跑 sorting
  - 然後 return array[k]

實戰不是刷 leetcode

- 在真的寫 quick search 之前……
  - 你有聽過 medium of medium pivot 嗎？
- 如果沒有，你的 quick search 很容易跌入 O(n^2) 耶

再說一次：請正確地理解問題

- 如果 k 很大， quick search 是最好解法
  - 如果 k 很小， quick search 不一定最好解法
    - 像是 k < 256
  - 最極端例子：你不會 k=1 時用 quick search 的

- 選擇演算法時，你有越多的額外資訊，你越有可能找到比通用算法更快的方案

如果 k 真的很小

- 算法：
  - 把 array 第 1 到 k 個 item 放到 maxHeap
    maxHeap 的 Top 就是：到目前為止的第 k 個 item
  - 之後，對 array 的第 K+1 到 n 個 item ：
    - 如果 currentItem 比 maxHeap.Top() 更大：

      maxHeap.Pop()
      maxHeap.Add(currentItem)
  - 然後， return maxHeap.Top()

O(n log k) vs O(n)

- 表面上，用 heap 對 array 掃一次，會比 quick sort 更慢
  - 如果 k 有足夠少，用 heap 掃瞄一次會更快
  - 延伸： Fibonacci heap 理論很快，實務上輸 binary heap 很大
- 延伸思考：
  - 為何 quicksort(和其變種) 幾乎打趴其他的 sorting ？

延伸思考１：

- 現在有 100M 個 integer ，我要拿第 100th 個 integer
- 我在用 AMD Ryzen 5950x
  （幻想中，沒錢買也搶不到……）
- 我可怎善用 16 core 32 thread 的優勢？

延伸思考２：

- 跟剛才一樣，但是我想拿第 1000000th 個 integer ？
- 答案是額外付費內容ＸＤ

完