演算法程式作業一

大數乘法

110502518陳文獻

實作大數的資料結構

65535(2^16)進位的無號整數陣列

在c++使用unsigned int [max_size]

max_size估計 = log_65535(10^5000) ~= 2077

設計測資說明

- 使用C++ rand()函式
- 一位一位生成兩個隨機長度5000的數字字串寫入txt檔(並確保開頭不為0)

傳統作法

- 對整數A*B，可實作成

(An,… ,A0)*(Bm,… ,B0)

= A0*B0 + A0*B1*base+…+A0*Bn*base^m+…An*Bm*base^(n+m)

共需要n*m次乘法

Karatsuba

- u = w * d^(n/2) + x
- v = y * d^(n/2) + z
- uv = wy d^n + (wz + xy) d^(n/2) + xz
- 1. Compute r = (w+ x)(y + z) = wy + (wz + xy) + xz
- 2. Compute (wz + xy) = r - wy – xz

次乘法次數可下降至 O(N^log_2(3))

實作方法

- 使用<windows.h>中的QueryPerformanceCounter()計時
- 進行字串轉大數->大數乘法->大數轉字串
- 200次取平均

遇到的困難

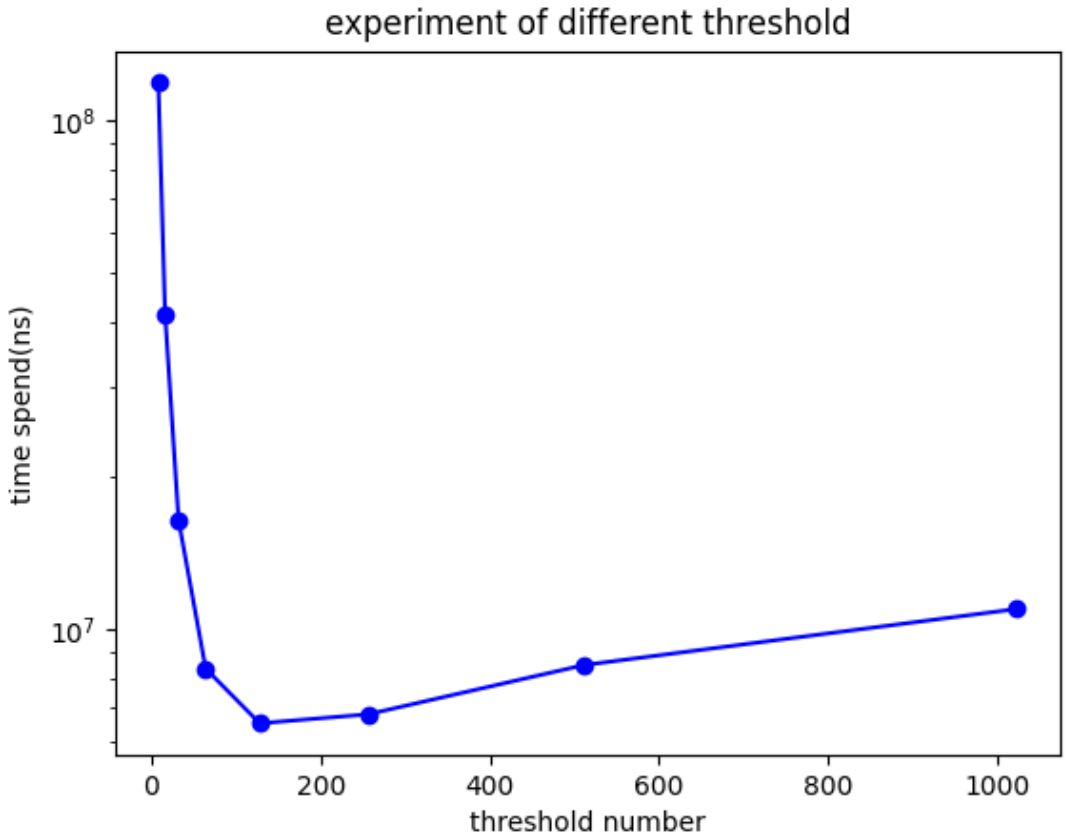
在我的原先的實作方法中，對於長度5000的整數，各部件所花費時間如下:

- 字串轉大數: 34892700 (ns)
- 大數乘法(傳統): 23930700 (ns)
- 大數乘法(Karatsuba1000): 48162100 (ns) # threshold = 1000
- 大數轉字串: 63092700 (ns)

經過修正後(優化套用傳統算法時的方法):

大數乘法(Karatsuba t=1000): 10696600 (ns)

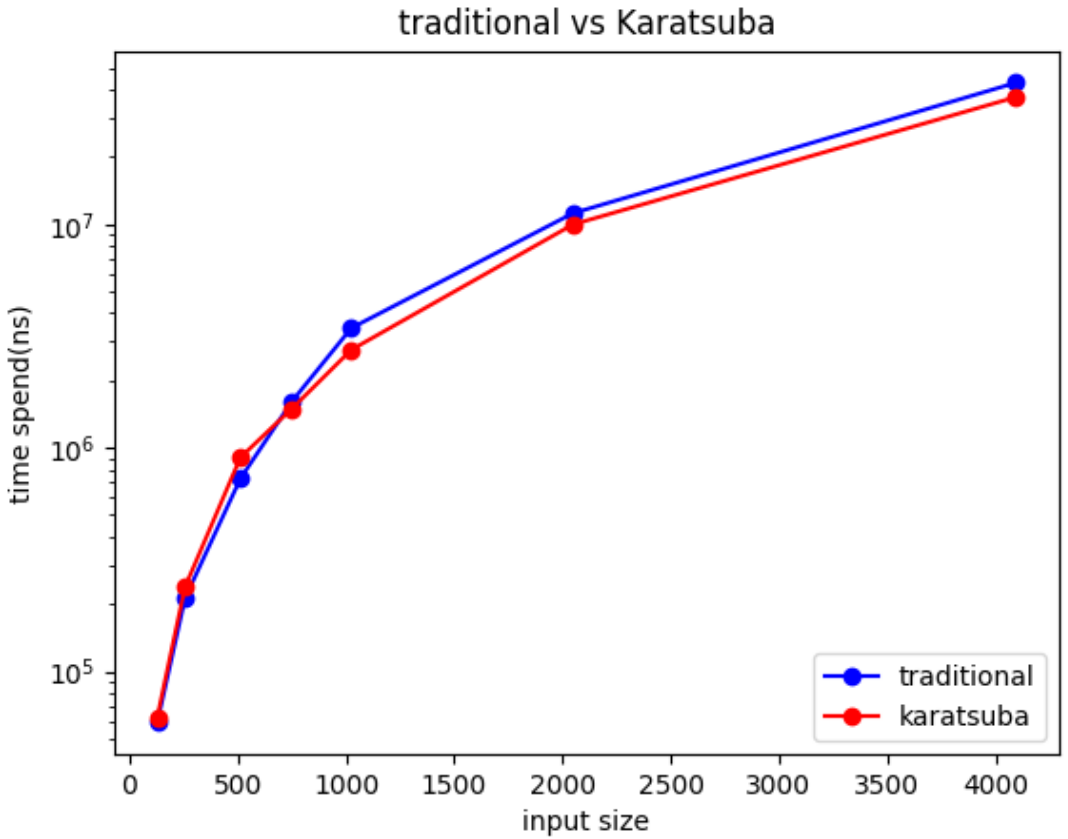
threshold的選擇



根據實驗結果，在輸入長度5000時，應將threshold設為128。

(此時只計算乘法過程而不包含轉換資料結構的時間)

實驗結果



根據實驗結果，在threshold=128時，karatsuba演算法會在輸入大小2^9(512)~2^10(1024)間超車。在input=750時 karatsuba 依然快於傳統算法。