

演算法程式作業一

資工 2B 林晉宇 110403518

一、大數相乘

本次作業實作大數相乘的兩種方法，分別為傳統作法及 Karatsuba 優化版本，並用 python 產生的測資拿來測試運行時間，並做比較，找出超車點為第幾位。

二、傳統作法

傳統的部分，我使用模擬直式乘法，每一位相乘完後相加。

- **資料結構:** 使用兩個字串代表乘數與被乘數；另開一個陣列儲存乘積的

每一位數字。

- **時間複雜度:** $O(n^2)$

- **Pseudo code:**

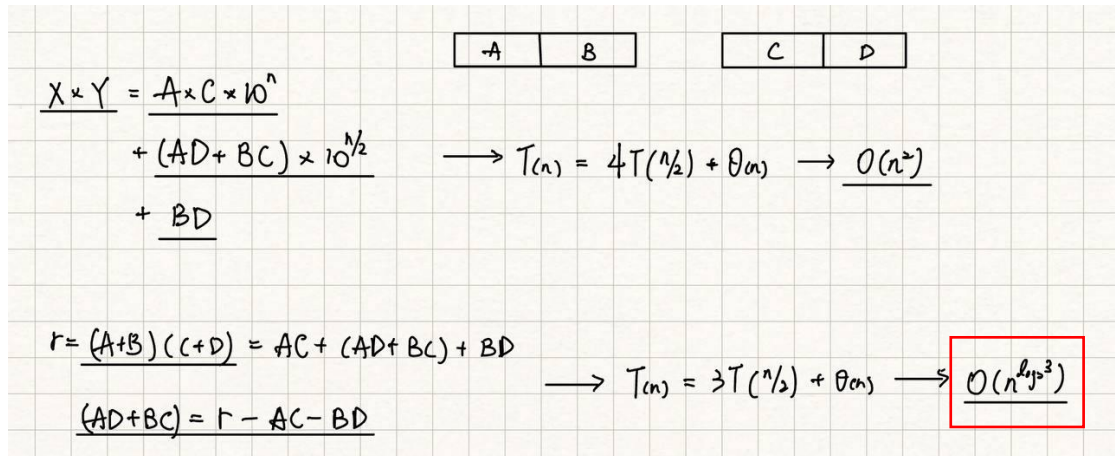
```
string n,m <- input;
int num_dig;
int product[10001]={0}

for(i = n.Length()-1 ~ 0){
    for(j = m.Length()-1 ~ 0){
        int ia=n.Length()-1-i,ib=m.Length()-1-j;
        product[ia+ib] += n[i]*m[j];
        num_dig = max(num_dig, ia+ib);
        if(product[ia+ib]>=10){
            product[ia+ib+1]+=product[ia+ib]/10;
            product[ia+ib]%=10;
            num_dig=max(num_dig, ia+ib+1);
        }
    }
}
```

三、Karatsuba 乘法

Karatsuba 使用上課教的方法，可以將時間複雜度降低。

- 時間複雜度:


$$\begin{aligned} X \times Y &= \underbrace{A \times C \times 10^n}_{\text{}} \\ &+ \underbrace{(AD + BC) \times 10^{n/2}}_{\text{}} \\ &+ \underbrace{BD}_{\text{}} \end{aligned} \quad \longrightarrow T(n) = 4T(n/2) + O(n) \longrightarrow O(n^2)$$
$$\begin{aligned} r &= \underbrace{(A+B)(C+D)}_{\text{}} = AC + (AD + BC) + BD \\ \underbrace{(AD + BC)}_{\text{}} &= r - AC - BD \end{aligned} \quad \longrightarrow T(n) = 3T(n/2) + O(n) \longrightarrow \boxed{O(n^{2.83})}$$

- 資料結構: 一樣使用兩個字串去儲存乘數以及被乘數。
- 函式: 包含大數的 相乘、相減、相加、平移、補 0 這些 function。
- Pseudo code:

```
string karatsuba(string x, string y)
{
    if(x.Length() == 1 && y.Length() == 1)
        return x[0] * y[0];

    if(x.Length() > y.Length())    y = appendZero();
    else    x = appendZero();
    if(x.Length() % 2)
    {
        x = "0" + x;
        y = "0" + y;
    }
    string a, b, c, d, ac, bd, abcd; // abcd = (a+b) * (c+d)
    int n = x.Length();
    a = x.substr(0, n/2);
    b = x.substr(n/2);
```

```

c = y.substr(0, n/2);
d = y.substr(n/2);

ac = karatsuba(a,c);
bd = karatsuba(b,d);
abcd = karatsuba(addXY(a,b),addXY(c,d));
return addXY(addXY(mul10(ac, n),mul10(subXY(subXY(abcd,ac),
bd), n/2)), bd);
}

```

四、比較

使用 python 產生測資，並將每次執行兩種方法所花的時間記錄下來，透過圖表呈現出來。

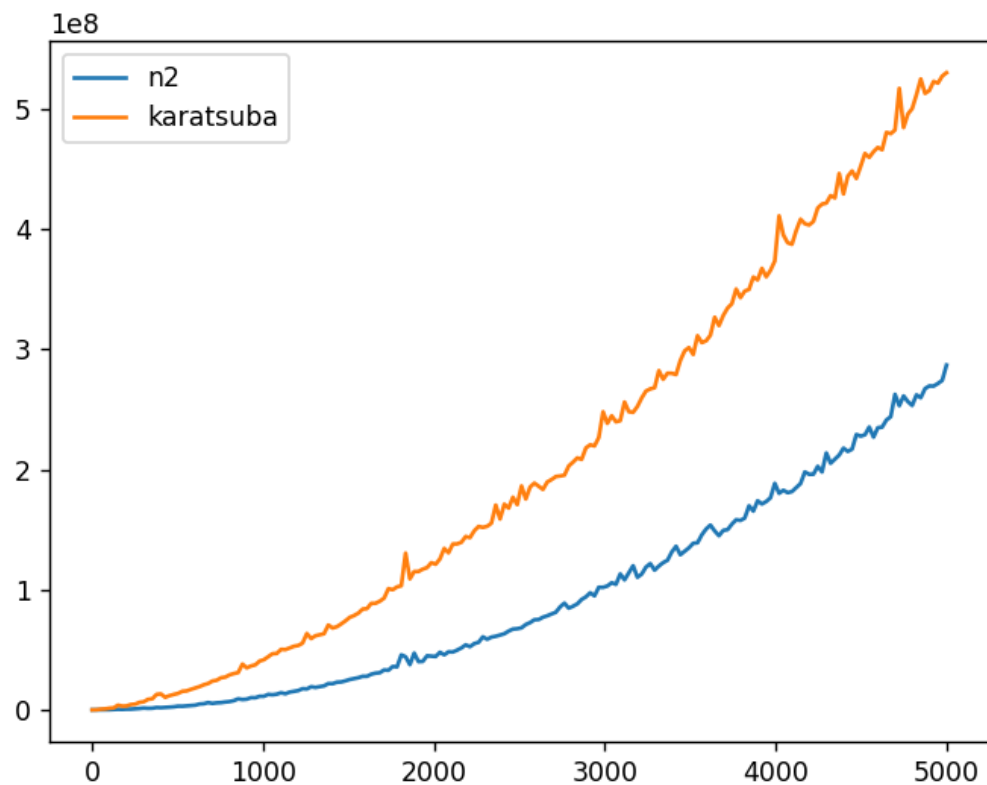
- **設計測資:** 範圍從 1~5000 位，每 20 位產生一比測資(用 python random 產生隨機數)，並丟進兩個 C++程式去執行，計算執行時間，在透過圖表呈現(matplotlib)。
- **Code: (隨機產生數字)**

```

16  ∨ for i in range(0,r+1):
17      # print(str(i)+" 位數 ")
18      n = ""
19      ⚡ m = ""
20      n += str(random.randrange(1,10))
21      m += str(random.randrange(1,10))
22  ∨ for j in range(0,i-1):
23      n += str(random.randrange(10))
24      m += str(random.randrange(10))
25      f=open("number.txt","w")
26      f.write(n)
27      f.write("\n")
28      f.write(m)
29  ∨ if i==0:
30      continue
31      os.system('n2')
32      os.system("karatsuba")

```

- **圖表:** 以下為實作出來的圖表，可以看到 **karatsuba** 跑出來的時間皆大於傳統方法，並不符合理論結果；然而我改進了許多次 **karatsuba** 的實作方式，也從網路上測試不同的程式，跑出來的結果皆慢於傳統方法。



五、結論

此次作業的實驗結果不是很理想，就理論來看的話，**karatsuba** 應該在位數越大的情況下，執行時間相比傳統做法要快得多，但實作出來的結果卻是 **karatsuba** 比傳統方法慢好幾倍。

我的猜想是有可能我實作 **karatsuba** 的加、減、平移或是補 0 的 function 太慢(然而我不確定如何能改進更快)，雖然這些 function 應該都是 $O(n)$ 級別的，但我的程式可能跑出來的常數十分大，導致其增長速度

比傳統快很多。

Threshold 的部分，如果以傳統方法為 $O(n^2)$ ，karatsuba 為 $O(3n^{\lg 3})$

的情況去推斷的話，我認為理想值應該會落在 13 到 20 之間，然而由於

我的實驗結果為傳統方法皆快於 karatsuba，所以並沒有一個明確的臨

界點，十分抱歉。