

Programming Assignment #1

Find Kth smallest element in an input array

112062636 游竣量

1. Your report must contain the flowchart or the pseudo code of your program. You have to describe how your approach works.

(a.) BFPRT 算法 (Median of medians)

```
FUNCTION partition(arr: ARRAY OF INTEGER, l: INTEGER, r: INTEGER,
pivot: INTEGER) -> INTEGER:
```

```
    // 在 arr 中從索引 l 到 r 找到 pivot 的位置
    pivotIndex <- find pivot in arr from index l to r
    // 交換 pivotIndex 和 r 的元素
    swap elements at pivotIndex and r in arr
```

```
    i <- l - 1
```

```
    FOR j FROM l TO r - 1:
```

```
        // 如果當前元素小於等於 pivot，則進行交換
```

```
        IF arr[j] <= pivot THEN:
```

```
            increment i
```

```
            swap elements at i and j in arr
```

```
    END FOR
```

```
    // 交換 i + 1 和 r 的元素
```

```
    swap elements at i+1 and r in arr
```

```
    RETURN i + 1
```

```
END FUNCTION
```

```
FUNCTION findMedian(arr: ARRAY OF INTEGER, l: INTEGER, r: INTEGER)
-> INTEGER:
```

```
    // 對 arr 從索引 l 到 r 進行排序
```

```
    sort arr from index l to r
```

```
    // 返回中位數
```

```
    RETURN arr at index (l + (r - l) / 2)
```

```
END FUNCTION
```

```

FUNCTION BFPRT(arr: ARRAY OF INTEGER, l: INTEGER, r: INTEGER, k:
INTEGER, G: INTEGER) -> INTEGER:
    // 如果只有一個元素，則直接返回
    IF l == r THEN:
        RETURN arr[l]

    size <- r - l + 1
    medians <- empty array
    rounds <- size divided by G

    FOR i FROM 0 TO rounds - 1:
        start <- l + i * G
        // 找到分組的中位數
        median <- findMedian of arr from start to start + G - 1
        // 添加到 medians 列表中
        add median to medians
    END FOR

    // 如果有餘數的分組，也計算中位數
    IF size % G != 0 THEN:
        median <- findMedian of arr from l + rounds * G to r
        add median to medians

    sizeM <- length of medians
    // 遞迴尋找 medians 的 median
    median <- BFPRT(medians, 0, sizeM - 1, sizeM / 2, G)

    // 進行分區操作
    p <- partition arr with indices l, r and pivot median
    q <- p - l + 1
    IF q == k THEN:
        // 如果找到目標位置，則返回對應元素
        RETURN arr[p]
    ELSE IF q > k THEN:
        // 如果目標位置在左側，則繼續在左側尋找
        RETURN BFPRT(arr, l, p - 1, k, G)
    ELSE:

```

```

        // 如果目標位置在右側，則繼續在右側尋找
        RETURN BFPRT(arr, p + 1, r, k - q, G)
    END IF
END FUNCTION

```

(b.) Randomized Select 隨機挑選 pivot

```

FUNCTION randomizedPartition(arr: ARRAY OF INTEGER, l: INTEGER, r:
INTEGER) -> INTEGER:
    // 隨機選擇一個索引
    randomIndex <- random number between l and r (inclusive)
    // 使用 partition 函數並將隨機選擇的元素作為 pivot
    RETURN partition(arr, l, r, arr[randomIndex])
END FUNCTION

```

```

FUNCTION randomizedSelect(arr: ARRAY OF INTEGER, l: INTEGER, r:
INTEGER, k: INTEGER) -> INTEGER:
    // 如果只有一個元素，則直接返回
    IF l == r THEN:
        RETURN arr[l]

    // 進行隨機分區操作
    p <- randomizedPartition(arr, l, r)
    // 計算小於等於 median 的元素數目
    q <- p - l + 1
    IF q == k THEN:
        // 如果找到目標位置，則返回對應元素
        RETURN arr[p]
    ELSE IF q > k THEN:
        // 如果目標位置在左側，則繼續在左側尋找
        RETURN randomizedSelect(arr, l, p - 1, k)
    ELSE:
        // 如果目標位置在右側，則繼續在右側尋找
        RETURN randomizedSelect(arr, p + 1, r, k - q)
    END IF
END FUNCTION

```

2. You should compare the running time of your algorithm with the input elements are divided into groups 3, 5, 7, 9, and Randomized-Select. Average the execution time of 50~100 experiments for each group size and Randomized-Select. Besides, you must analyze the time complexity in different group sizes and show your results.

時間複雜度分析：

(a.) BFPRT 算法 (Median of medians)

● 3

當 group size 為 3 時，用 $O(n)$ 時間找出 medians，接著遞迴找出 Median of medians 花 $T(\text{ceil}(n/3))$ 時間，每次以 Median of medians 做 partition 時至少都能去掉 $n/3$ 個元素，對剩餘 $2n/3$ 個元素做遞迴花 $T(2n/3)$ 時間。

當 n 很大時：

$$T(n) = T(n/3) + T(2n/3) + O(n)$$

用遞迴樹解得 $T(n) = O(n \log n)$

● 5

當 group size 為 5 時，用 $O(n)$ 時間找出 medians，接著遞迴找出 Median of medians 花 $T(\text{ceil}(n/5))$ 時間，每次以 Median of medians 做 partition 時至少都能去掉 $3n/10$ 個元素，對剩餘 $7n/10$ 個元素做遞迴花 $T(7n/10)$ 時間。

當 n 很大時：

$$T(n) = T(n/5) + T(7n/10) + O(n)$$

用遞迴樹解得 $T(n) = O(n)$

● 7

當 group size 為 7 時，用 $O(n)$ 時間找出 medians，接著遞迴找出 Median of medians 花 $T(\text{ceil}(n/7))$ 時間，每次以 Median of medians 做 partition 時至少都能去掉 $4n/14$ 個元素，對剩餘 $10n/14$ 個元素做遞迴花 $T(10n/14)$ 時間。

當 n 很大時：

$$T(n) = T(n/7) + T(10n/14) + O(n)$$

用遞迴樹解得 $T(n) = O(n)$

● 9

當 group size 為 9 時，用 $O(n)$ 時間找出 medians，接著遞迴找出 Median of medians 花 $T(\text{ceil}(n/9))$ 時間，每次以 Median of medians 做 partition 時至少都能去掉 $5n/18$ 個元素，對剩餘 $13n/18$ 個元素做遞迴花 $T(13n/18)$ 時間。

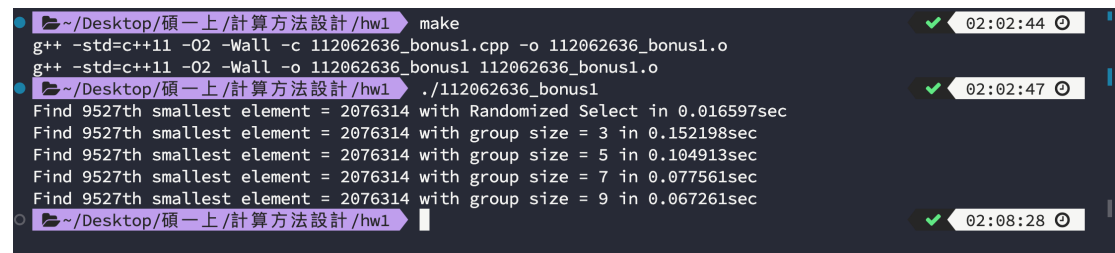
當 n 很大時：

$$T(n) = T(n/9) + T(13n/18) + O(n)$$

用遞迴樹解得 $T(n) = O(n)$

(b.) Randomized Select 隨機挑選 pivot

結果：



```
~/Desktop/碩一上/計算方法設計/hw1 make 02:02:44
g++ -std=c++11 -O2 -Wall -c 112062636_bonus1.cpp -o 112062636_bonus1.o
g++ -std=c++11 -O2 -Wall -o 112062636_bonus1 112062636_bonus1.o
~/Desktop/碩一上/計算方法設計/hw1 ./112062636_bonus1 02:02:47
Find 9527th smallest element = 2076314 with Randomized Select in 0.016597sec
Find 9527th smallest element = 2076314 with group size = 3 in 0.152198sec
Find 9527th smallest element = 2076314 with group size = 5 in 0.104913sec
Find 9527th smallest element = 2076314 with group size = 7 in 0.077561sec
Find 9527th smallest element = 2076314 with group size = 9 in 0.067261sec
~/Desktop/碩一上/計算方法設計/hw1 02:08:28
```

討論：可以觀察到 **Randomized Select** 在 **real time** 執行時間是最小的，而在 **Median of medians** 算法中，當選擇的 **group size** 越大，執行時間越小。即使 **Median of medians** 算法在數學上的執行時間是 $O(n)$ ，但是因為它的 **constant c** 是很大的，因此效果甚至不如 **Randomized Select**，**Median of medians** 算法通常只在學術上應用。