B06505004 莊博翰 HW3

```
/*************************************************************************
 *  Describe how your firstIndexOf() method in BinarySearchDeluxe.java
 *  finds the first index of a key that is equal to the search key.
 *************************************************************************/
```

基本上跟一般的 binary search 一樣，不過在找到相同的 key 時，會視為找到比較大的並更新一個名為 lastid(最後找到的 id)的變數，如此一來會繼續往左(小)找，直到 first index == last index ，回傳 lastid，這個時後 first index 會剛好停留在第一個(id 最小)的目標值

```java
// that is equal to the search key, or -1 if no such key.
public static <Key> int firstIndexOf(Key[] a, Key key, Comparator<Key> comparator) {
    int idf = 0;
    int ide = a.length - 1;
    int mid;
    int cmp;
    int lastid = -1;
    while (idf <= ide) {
        mid = idf + (ide - idf) / 2;
        cmp = comparator.compare(key, a[mid]);
        if (cmp < 0) ide = mid - 1;
        else if (cmp > 0) idf = mid + 1;
        else {
            ide = mid - 1;
            lastid = mid;
        }
    }
    return lastid;
}
```

```
/*************************************************************************
 *  Identify which sorting algorithm (if any) that your program uses in the
 *  Autocomplete constructor and instance methods. Choose from the following
 *  options:
 *
 *     none, selection sort, insertion sort, mergesort, quicksort, heapsort
 *
 *  If you are using an optimized implementation, such as Arrays.sort(),
 *  select the principal algorithm.
 *************************************************************************/
```

因為題目要求 n log n worst case 所以選用 merge sort

Autocomplete() : merge sort

allMatches() : merge sort

numberOfMatches() : none (不須知道順序)

```
/*************************************************************************
```

```
 *  How many compares (in the worst case) does each of the operations in the
 *  Autocomplete data type make, as a function of both the number of terms n
 *  and the number of matching terms m? Use Big Theta notation to simplify
 *  your answers.
 *
 *  Recall that with Big Theta notation, you should discard both the
 *  leading coefficients and lower-order terms, e.g., Theta(m^2 + m log n).
**************************************************************************
```

Autocomplete(): Theta(n log n) #merge sort
//要把所有 term 先排序一次

allMatches(): Theta(m log m +log n) # binary search + merge sort
// 先找到目標集合，在把集合中的 term 排序

numberOfMatches(): Theta(log n)# binary search
// 找到目標集合，自然會知道集合大小(lasted-firstid +1)