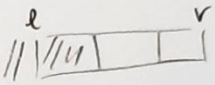
Algorithm A2 , 410821305 薛祖恩

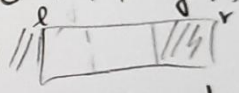
2-b.

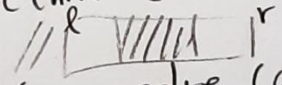
(c++ psuedo)

```
int solve(int *arr, int left, int right, int key);
    if (right ≥ left) {
        int mid1 = left + (right - left) / 3,
            mid2 = right - (right - left) / 3;
        if (arr[mid1] or arr[mid2] is key)
            return mid1 or mid2;
        if (key < arr[mid1])
            return solve(arr, left, mid-1, key);
        else if (key > arr[mid2])
            return solve(arr, mid2+1, right, key);
        else
            return solve(arr, mid1+1, mid2-1, key);
    }
    return -1; // Not found

}
```




Time $B(n) = O(1)$ , $W(n) = O(\log_3 n)$ #

Space b => $O(1)$ #

2-7.

```
int solve (int *arr, int left, int right) {
    if (left == right)
        return arr[left];
    int mid = (left + right)/2;
    return max(solve(arr, left, mid), solve(arr, mid+1, right));
}
```

(n elements in array)

Time $T(n) \Rightarrow O(n)$ #

Space 4 $\Rightarrow O(1)$ #

2-13.

```
void mergesort(int *arr, int low, int high)
    if(high - low) return;
    int mid1 = low + (high-low)/3;
    int mid2 = low + 2 x (high-low)/3 + 1;
    mergesort(arr, low, mid1);
    mergesort(arr, mid1, mid2);
    mergesort(arr, mid2, high);
    merge(arr, low, mid1, mid2, high);

int merge(int *arr, int low, int mid1, int mid2, int high)
    i = low, j = mid1, k = mid2, l = low, ans[size];
    while( i<mid1 && j<mid2 && k< high)
        compare the smallest value and put it in "ans" away.

    while( i< mid1 && j< mid2)
        compare the smallest value and put it in "ans" away.
    while(j<mid2 && k< high)
        compare the smallest value and put it in "ans" away.

    while (i<mid1 && k<high)
        compare the smallest value and put it in "ans" away
    while (i<mid1)
        put the rest in the "ans" away
    while (j<mid2)
        put the rest in the "ans" away
    while (k< high)
        put the rest in the "ans" away

    return *ans;
```

Time complexity.
$O(n \log_3 n)$
Space complexity
$O(n)$