第九章作业

2022211363 谢牧航

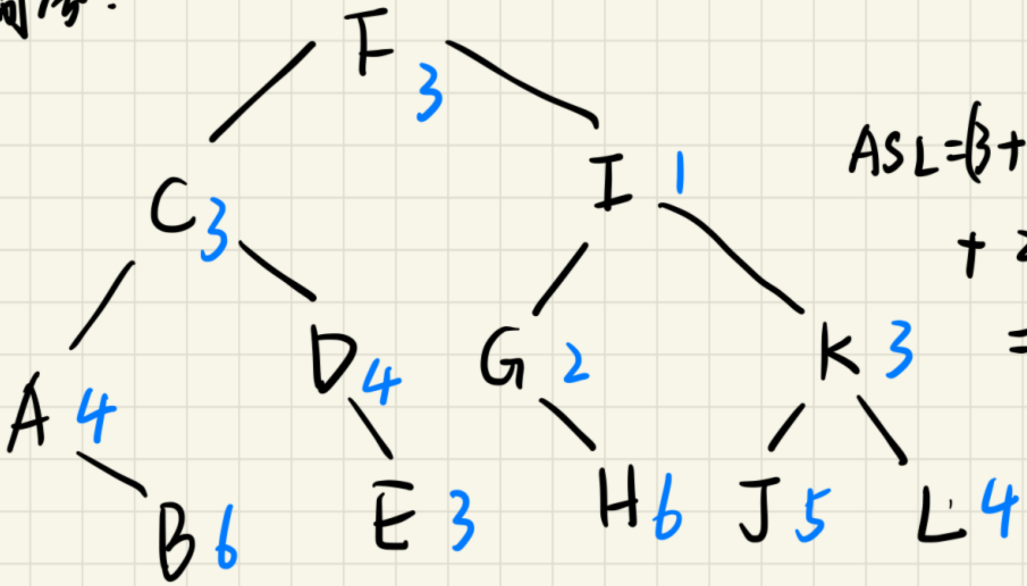
第九章作业 2022211363 谢牧航

一、选择.

1. A 2. D 3. C. 4. D 5. A 6. A 7. B

二、简答.

1.(1)



$ASL = (3 + 4\times 2 + 13\times 3 + 24\times 4)/12$

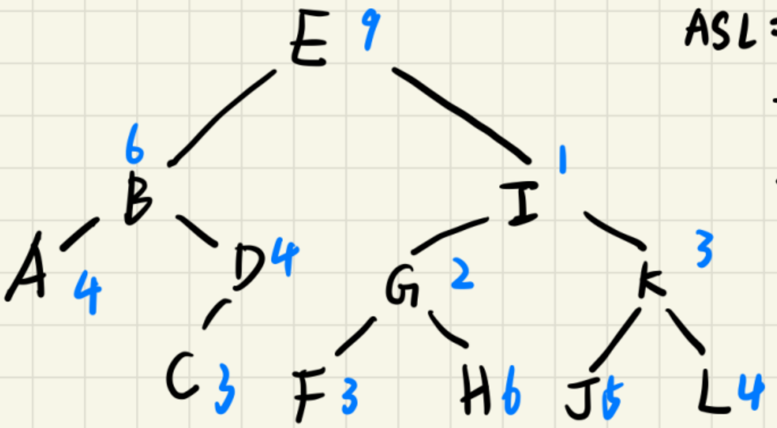
$= 146/12$

$= \frac{73}{6}$

(2). 前缀和

A	B	C	D	E	F	G	H	I	J	K	L
4	10	13	17	26	29	31	37	38	43	46	50

⇒



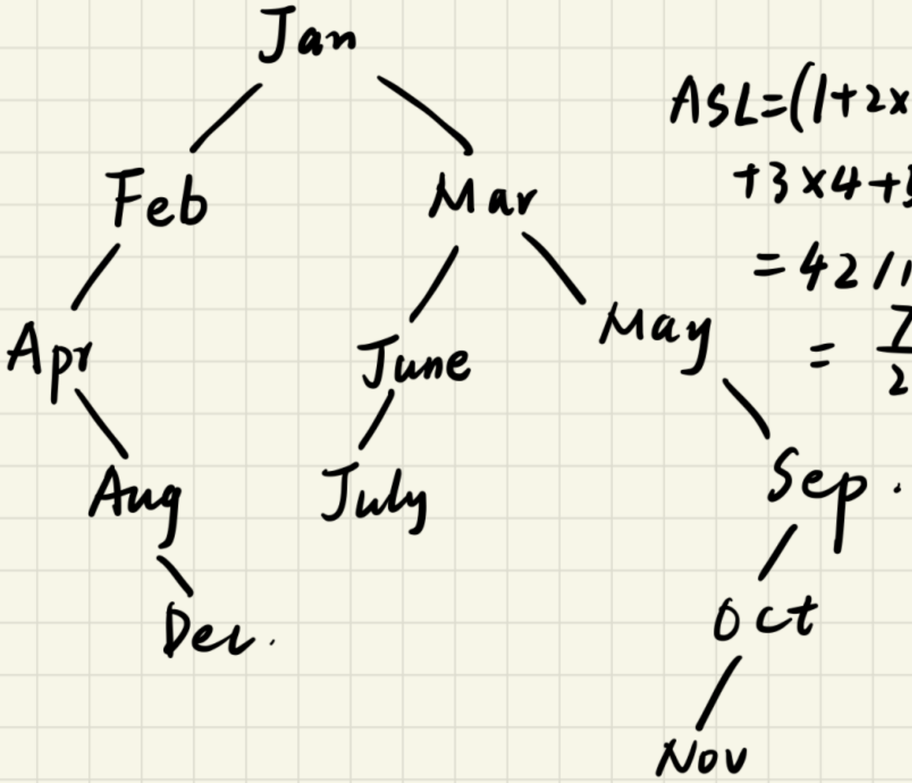
$ASL = (9 + 7\times 2 + 13\times 3 + 21\times 4)/12$

$= (9 + 14 + 39 + 84)/12$

$= 146/12$

$= \frac{73}{6}$

2. (1).

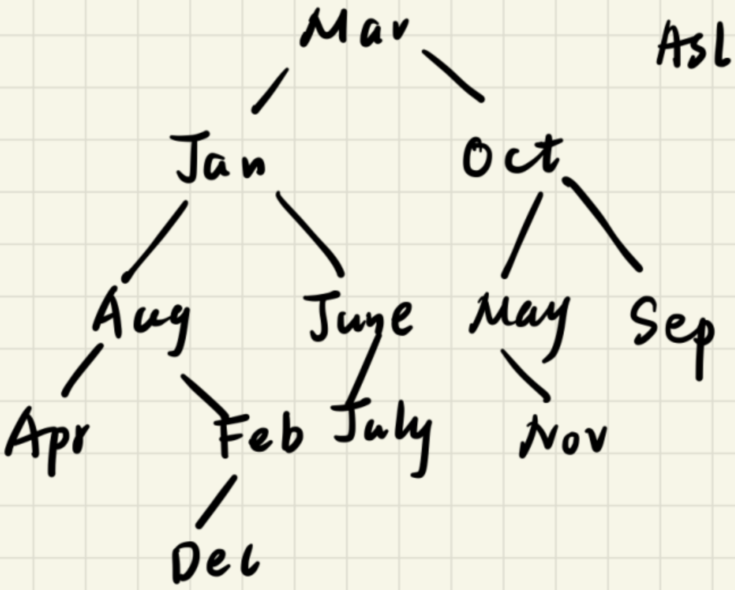


$ASL=(1+2\times2+3\times3+3\times4+5\times2+6)/12$

$=42/12$

$=\frac{7}{2}$

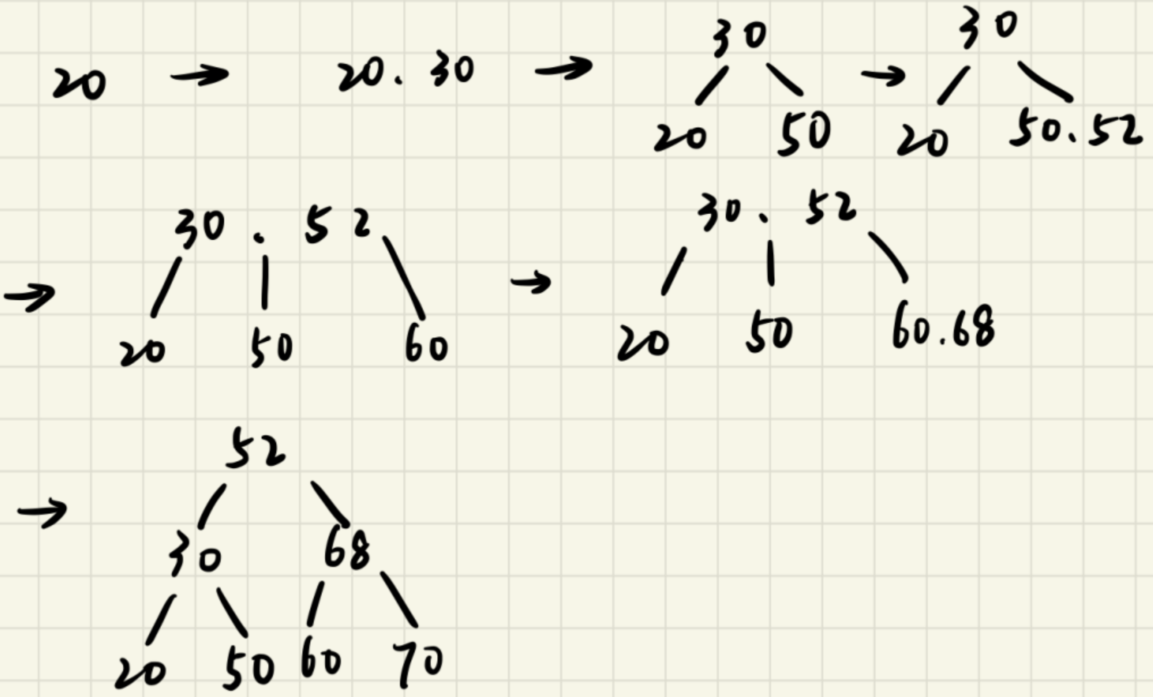
(2)



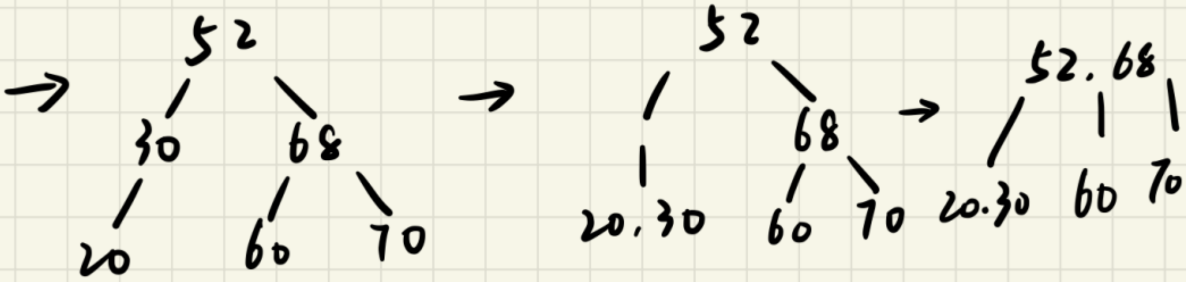
$ASL=(1+2\times2+4\times3+4\times4+5)/12=38/12$

$=\frac{19}{6}$

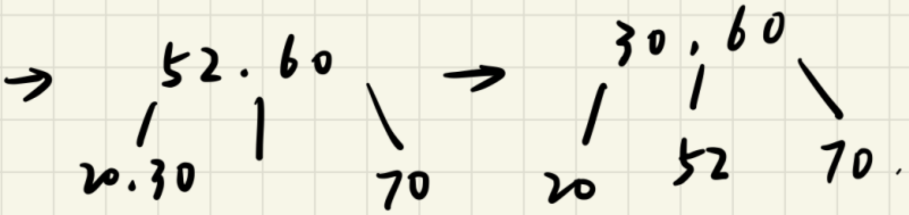
3.



删 50



删68



40.

K	22	41	53	46	30	13	01	67
3K mod 11	0	2	5	6	2	6	3	3

(1).

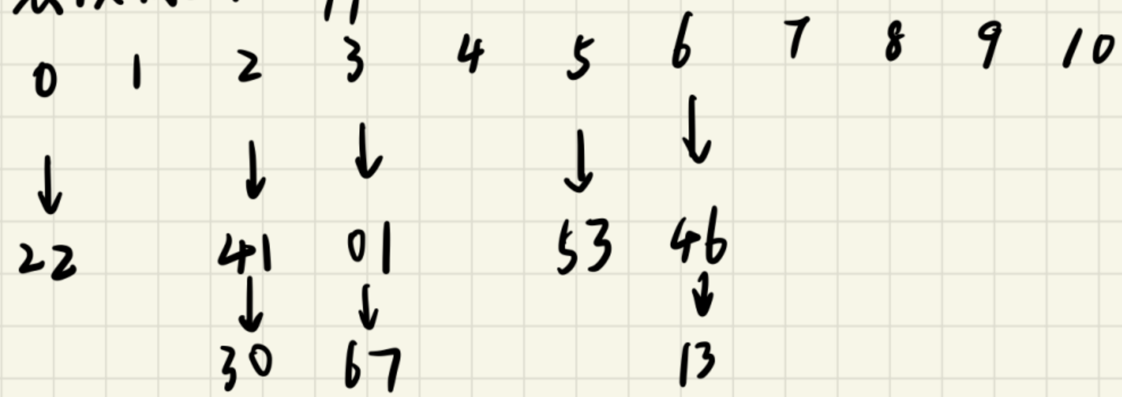
0	1	2	3	4	5	6	7	8	9	10
22		41	30	01	53	46	13	67		

成功ASL =(1+1+1+1+2+2+2+6)/8= 16/8 = 2

失败ASL $=(2+1+8+7+6+5+4+3+2+1+1)/11=\frac{40}{11}$

装填因子：$\frac{8}{11}$

(2).



成功ASL $=(5+3\times2)/8=\frac{11}{8}$

失败ASL $=(1+2+2+1+2)/11=\frac{8}{11}$

装填因子 $=\frac{5}{11}$

三、算法题

1.

```
int binary_search_last_not_greater(int arr[], int size, int value) {
    int left = 0;
    int right = size - 1;
    int result = -1;
    while (left <= right) {
        int mid = left + (right - left) / 2;

        if (arr[mid] <= value) {
            result = mid; // 更新结果为当前中点，但继续在右侧查找可能的更大值
            left = mid + 1;
        } else {
            right = mid - 1;
        }
    }
    return result;
}
```

2.

```c
void insert_in_order(int arr[], int* size, int x) {
    int left = 0;
    int right = *size - 1;
    int pos = *size; // 默认插入位置是数组末尾

    while (left <= right) {
        int mid = left + (right - left) / 2;
        if (arr[mid] < x) {
            left = mid + 1;
        } else {
            pos = mid; // 更新插入位置为中点
            right = mid - 1;
        }
    }

    // 将元素从 pos 位置开始向后移动
    for (int i = *size; i > pos; --i) {
        arr[i] = arr[i - 1];
    }

    arr[pos] = x; // 在正确位置插入元素 x
    (*size)++; // 增加数组的大小
}
```

3.

```c
void print_if_not_less_than_x(TreeNode *root, int x) {
    if (root == NULL) {
        return;
    }
    // 先递归访问右子树
    print_if_not_less_than_x(root->right, x);
    // 如果当前节点的值小于x，则不需要再访问左子树
    if (root->val < x) {
        return;
    }
    // 输出当前节点的值
    printf("%d ", root->val);
    // 最后递归访问左子树
    print_if_not_less_than_x(root->left, x);
}
```