一、选择题

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

*第七题若 `mid = (l + r + 1) / 2`，那么答案是 A

8	9	10	11	12	13	14
A	A	B	C	D	D	A

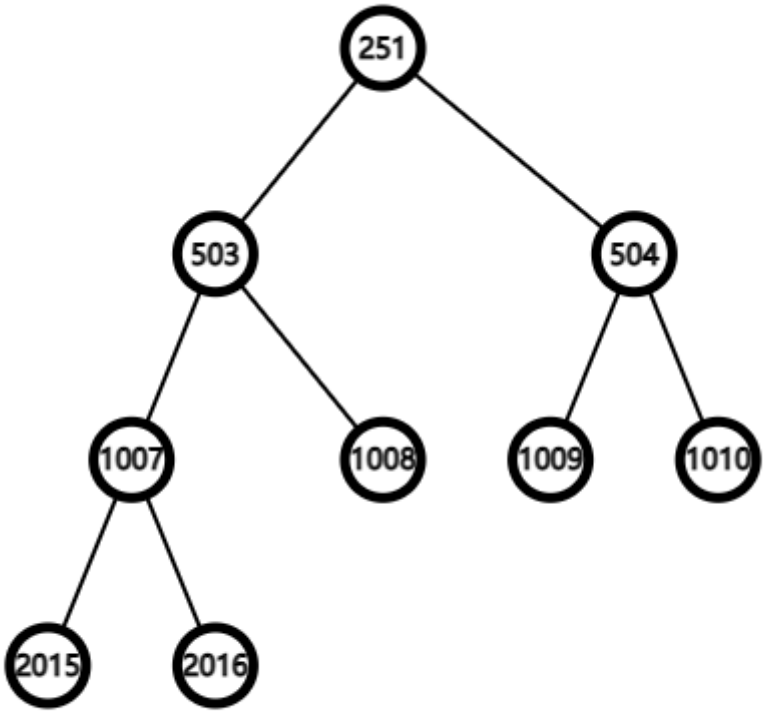
二、填空题

1. `L->next==NULL`
2. 归并
3. 13
4. kruskal
5. $\Theta(n)$
6. 6; 2,3,2
7. XYbXYba
8. 环
9. 之后
10. 4
11. n

三、应用题

T1

1. 在 TR 上（TR: 2, 5~6, 11~14, 23~30, 47~62, 95~126, 191~254, 383~510, 767~1022, 1535~2046）
2. 481（=2016-1535）

3. 

T2

1.

	0	1	2	3	4	5	6	7	8	9	10
HT	11	22	46	13	01	70			41	31	30

2. 查找次数: {11: 1, 46: 1, 13: 2, 01: 5, 70: 2, 41: 1, 31: 1, 30: 3}
   概率: 15/8 = 1.875
3. 查找次数: {0: 7, 1: 6, 2: 5, 3: 4, 4: 3, 5: 2, 6: 1, 7: 1, 8: 10, 9: 9, 10: 8}
   概率: 56/11 = 5.091

T3

1. 85, 72, 49, 65, 34, 40, 43, 58, 20
   (初始堆: 95, 85, 49, 72, 34, 40, 43, 58, 65, 20)
2. 34, 43, 72, 85, 40, 49, 58, 95, 20, 65
3. 20, 58, 40, 34, 85, 43, 49, 95, 65, 72

T4

$$k = \begin{cases} 0 & i = 0, j = n-1 \\ 3i + j - n & otherwise \end{cases}$$

T5

S		D		P					
0		0		0					
0		20		0	3	4	1		
1		19		0	3	4	2		
1		10		0	3				
1		17		0	3	4			
0		25		0	3	4	5		

四、算法设计

T1

1.

```
List init(int A[], int n) {
    List head = new List, p = res;
    for (int i = 1; i <= n; ++i) {
        p->next = new List;
        p = p->next;
        p->val = A[i];
    }
    p->next = NULL;
    return head;
}
```

2.

```
List reverse(List pList) {
    List head = new List;
    head->next = NULL;
    while (pList->next) {
        List current = new List;
        current->val = pList->next->val;
        current->next = head->next;
        head->next = current;
        pList = pList->next;
    }
    return head;
}
```

T2

1.

```
bool checkBST(bTree *pBTree) {
    if (pBTree->lchild && (pBTree->lchild->data >= pBTree->data || !check
BST(pBTree->lchild))) return false;
    if (pBTree->rchild && (pBTree->rchild->data <= pBTree->data || !check
BST(pBTree->rchild))) return false;
    return true;
}
```

2.

```
int countDepthTotal(bTree *pBTree, int depth) {
    int ret = depth;
    if (pBTree->lchild) ret += countDepthTotal(pBTree->lchild, depth +
1);
    if (pBTree->rchild) ret += countDepthTotal(pBTree->rchild, depth +
1);
    return ret;
}
int countNodeTotal(bTree *pBTree) {
    int ret = 1;
    if (pBTree->lchild) ret += countNodeTotal(pBTree->lchild);
    if (pBTree->rchild) ret += countNodeTotal(pBTree->rchild);
    return ret;
}

double calcASL(bTree *pBTree) {
    return (double)countDepthTotal(pBTree) / countNodeTotal(pBTree)
}
```