1.

(1)

```
Node* search(Node* head, Node*& p, int K) {
    if (p == nullptr) p = head;
    if (p->key < K) {
        while (p != nullptr && p->key < K) {
            p = p->next;
        }
    }
    else if (p->key > K) {
        while (p != nullptr && p->key > K) {
            p = p->prev;
        }
    }
    if (p != nullptr && p->key == K) {
        return p;
    }
    return nullptr;
}
```

(2)
注意到:

$$ASP_{succ} = \frac{1}{n}\sum_{i\in[n]}\frac{1}{n}\sum_{j\in[n]}|i-j| = \frac{n}{3} - \frac{1}{3n}$$

2.

1. **排序集合：**
   - 首先对集合 S_1 和 S_2 排序，时间复杂度分别为 $O(N \log N)$ 和 $O(M \log M)$。
   - 排序的目的是便于在交集中快速查找公共元素。
2. **双指针遍历：**
   - 初始化两个指针 i 和 j，分别指向 S_1 和 S_2 的起始位置。
   - 根据以下规则移动指针：
     - 如果 $S_1[i] = S_2[j]$，将该元素加入交集，两个指针同时向前移动。
     - 如果 $S_1[i] < S_2[j]$，指针 i 向前移动。
     - 如果 $S_1[i] > S_2[j]$，指针 j 向前移动。
3. **终止条件：**

- 当任一指针超过对应集合长度时，停止遍历。

4. **结果返回：**
   - 返回保存交集结果的集合。

```
Function Intersection(S1, S2):
    sort S1  // 时间复杂度 O(N log N)
    sort S2  // 时间复杂度 O(M log M)
    i ← 0
    j ← 0
    Result ← Empty Set
    while i < N and j < M: // 时间复杂度 O(N + M)
        if S1[i] == S2[j]:
            Add S1[i] to Result
            i ← i + 1
            j ← j + 1
        else if S1[i] < S2[j]:
            i ← i + 1
        else:
            j ← j + 1
    return Result
```

综上, 总时间复杂度为:

$$O(N \log N) + O(\log N \log \log N) + O(N + \log N) = O(N \log N)$$

3.

(1)

HT	0	1	2	3	4	5	6	7	8	9
Key	21	14	3	5	20		9	37		

检索成功的平均长度:

$$\frac{1}{7}(1+2+1+1+1+2+3) \approx 1.57$$

(2)

HT	0	1	2	3	4	5	6	7	8	9
Key	21	5	3	14	20		9		37	

检索成功的平均长度:

$$\frac{1}{7}(1+2+1+1+1+2+1) \approx 1.29$$