数算作业10

1

(1)：

```
typedef struct Node {
    int key;
    struct Node *prev, *next;
} Node;

Node* search(Node* head, Node** p, int K) {

    Node *start = *p;
    if (start == NULL) return NULL;
Node *curr = start;
if (curr->key == K) {
    while (curr->next && curr->next->key == K) {
        curr = curr->next;
    }
    *p = curr;
    return curr;
} else if (K < curr->key) {
    // 向前搜索
    while (curr && curr->key > K) {
        curr = curr->prev;
    }
    if (curr && curr->key == K) {
        Node *temp = curr;
        while (temp->next && temp->next->key == K) {
            temp = temp->next;
        }
        *p = temp;
        return temp;
    } else {

        return NULL;
    }
} else {

    while (curr && curr->key < K) {
        curr = curr->next;
    }
    if (curr && curr->key == K) {
        // 找到后同理向后检查重复
        while (curr->next && curr->next->key == K) {
            curr = curr->next;
        }
        *p = curr;
        return curr;
    } else {
        // 没找到
```

```
            return NULL;
        }
    }
        }
```

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

2

算法思想:

借助于堆排序，将S_2从小到大排序，再针对S_1中的每个元素，在排序后的S_2中做二分查找

```
//用privority_queue
#include <iostream>
#include <queue>
#include <vector>
using namespace std;
int main()
{
    priority_queue<int, vector<int>, greater<int>> que;
    vector<int> S1;
    vector<int> S2;
    for (int i = 0; i < S2.size(); i++)
    {
        que.push(S2[i]);
    }
    vector<int> S3;
    for (int i = 0; i < S2.size(); i++)
    {
        S3.push_back(que.top());
        que.pop();
    }
    vector<int> result;
    for (int i = 0; i < S1.size(); i++)
    {
        int left = 0;
        int right = S3.size() - 1;
        while (left < right)
        {
            int mid = (left + right) / 2;
            if (S3[mid]==S2[i])
            {
                result.push_back(S2[i]);
                S3[mid] = S3[S3.size() - 1];//此处优化了一下
                S3.pop_back();
                break;
            }
            else if(S2[i]>S3[mid]){
                left = mid + 1;
            }
            else{
                right = mid - 1;
            }
```

```
        }
    }
}
```

时间复杂度：

堆排序：$O(log\,n(log(log\,n))$

二分找交集: $O(n\times log\,n(log\,n))$

总的时间复杂度 $O(n\times log\,n(log\,n))$

3

（1）

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

检索成功的平均长度为$\frac{1}{7}\times(1+2+1+1+1+2+3)\approx 1.57$

（2）

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

检索成功的平均长度$\frac{1}{7}\times(1+2+1+1+1+2+1)\approx 1.29$