**Advanced Algorithms**

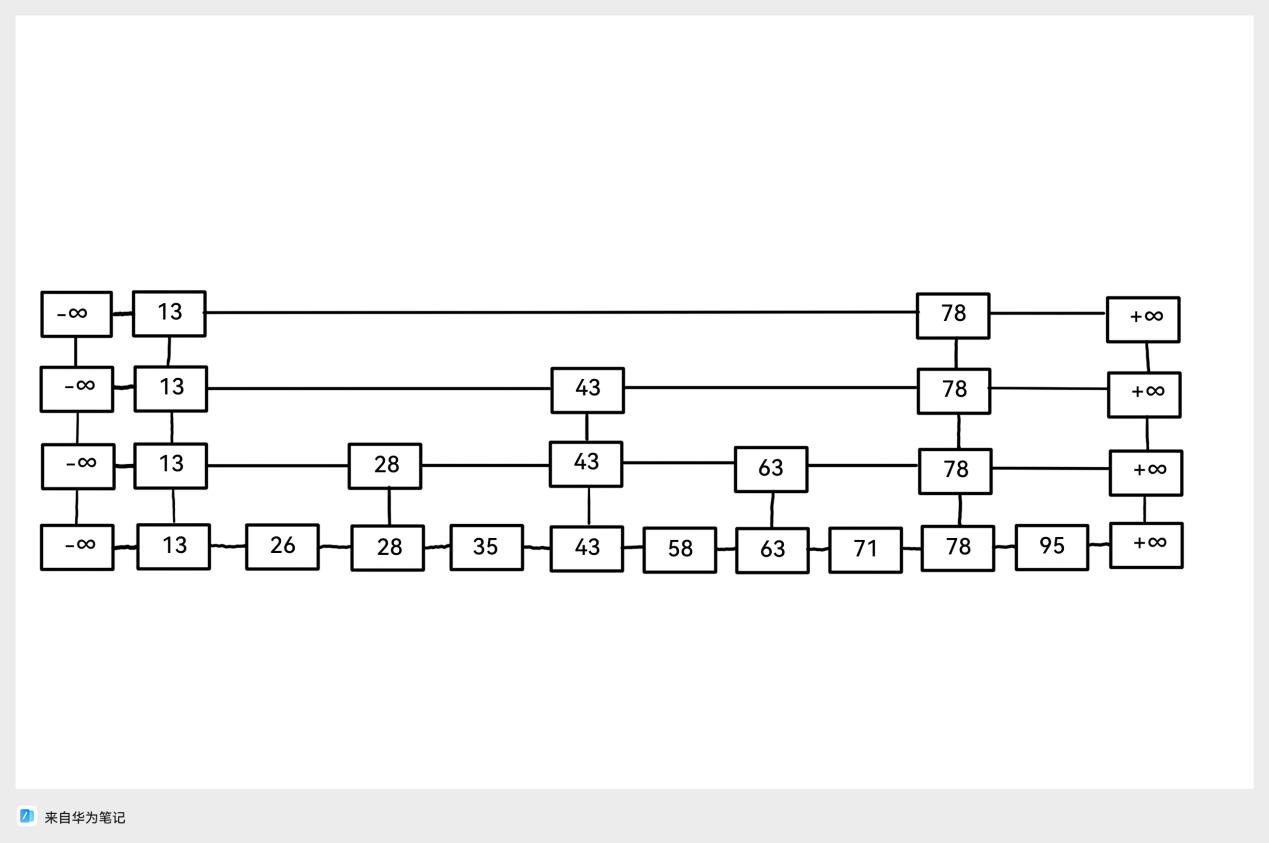
**Exercise for Lecture 9**

|  |  |  |  |
| --- | --- | --- | --- |
| **Student Name** |  | **Student ID** |  |
| **Problem 1** |  | | |
| **Problem 2** |  | | |
| **Problem 3** |  | | |
| **Total Score** |  | | |
| **Notes** | Deadline: **2023-10-13 24:00**  Submission Format: ‘**Lecture9\_Name\_Student ID.docx**’, and please send to: **[chenlq1997@126.com](mailto:algorithms_23fall@163.com)**.  This assignment is meant to be an evaluation of your **individual** understanding coming into the course and should be completed **without collaboration** or outside help. | | |

**Problem 1.[30 points]**

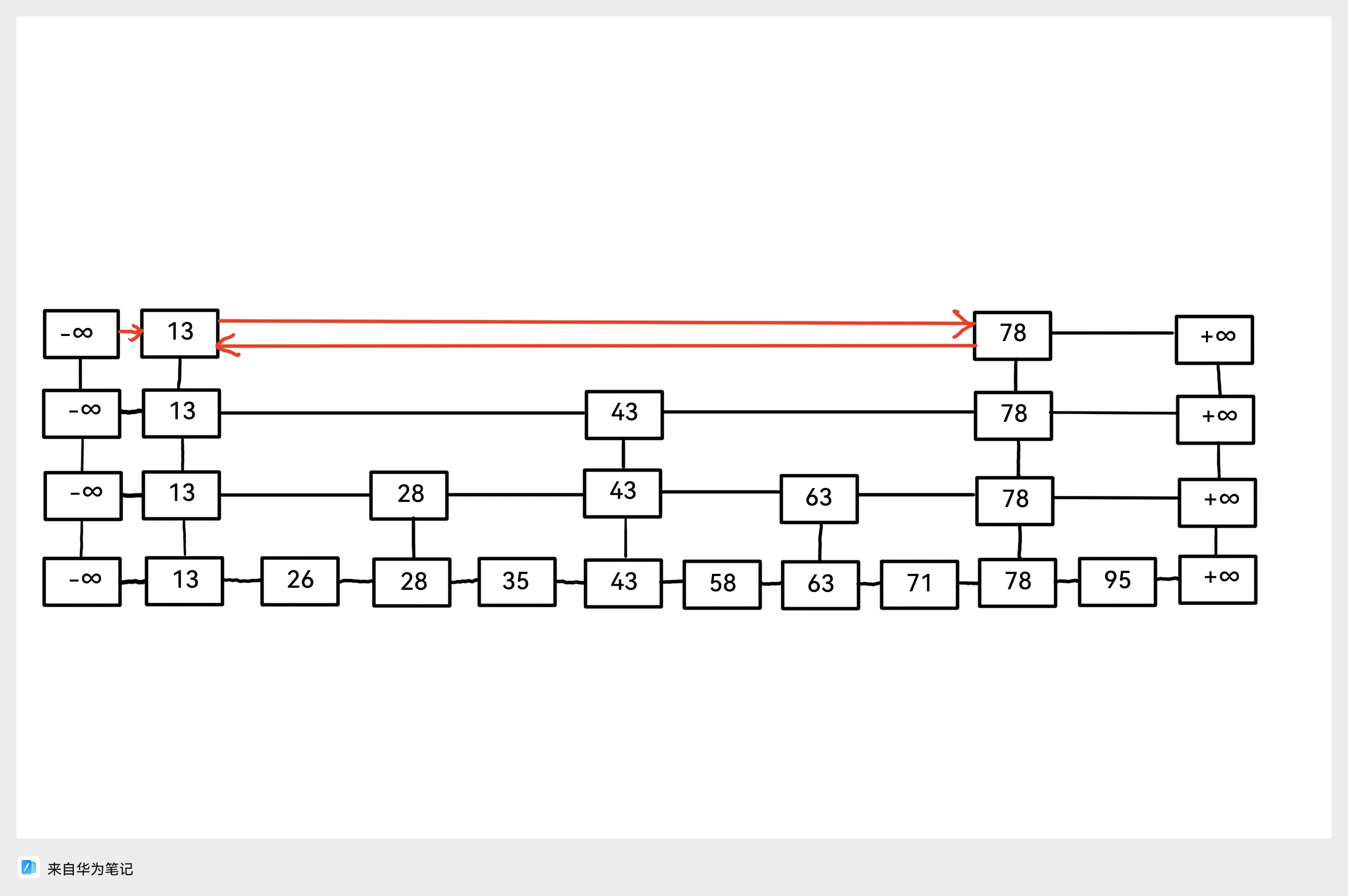
1.1 Please show how to search 35 in the next skip list. Each comparison is an intermediate step, and you need to provide each step. The answer can be shown in one picture and the example is shown as follows:

1.2 Please show how to search 71 in the next skip list.

****

**Picture 1. The skip list**

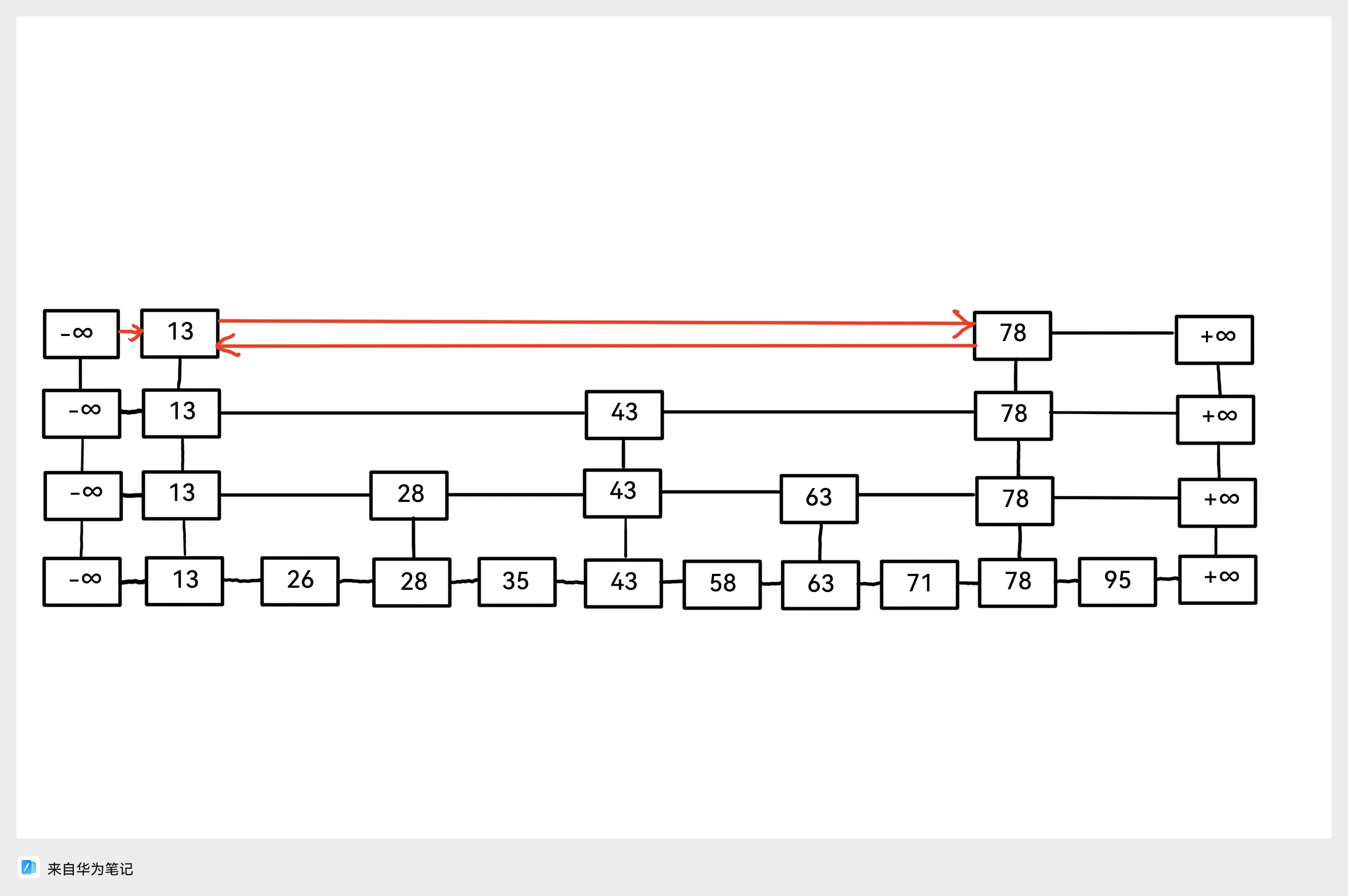
**Search 26**

****

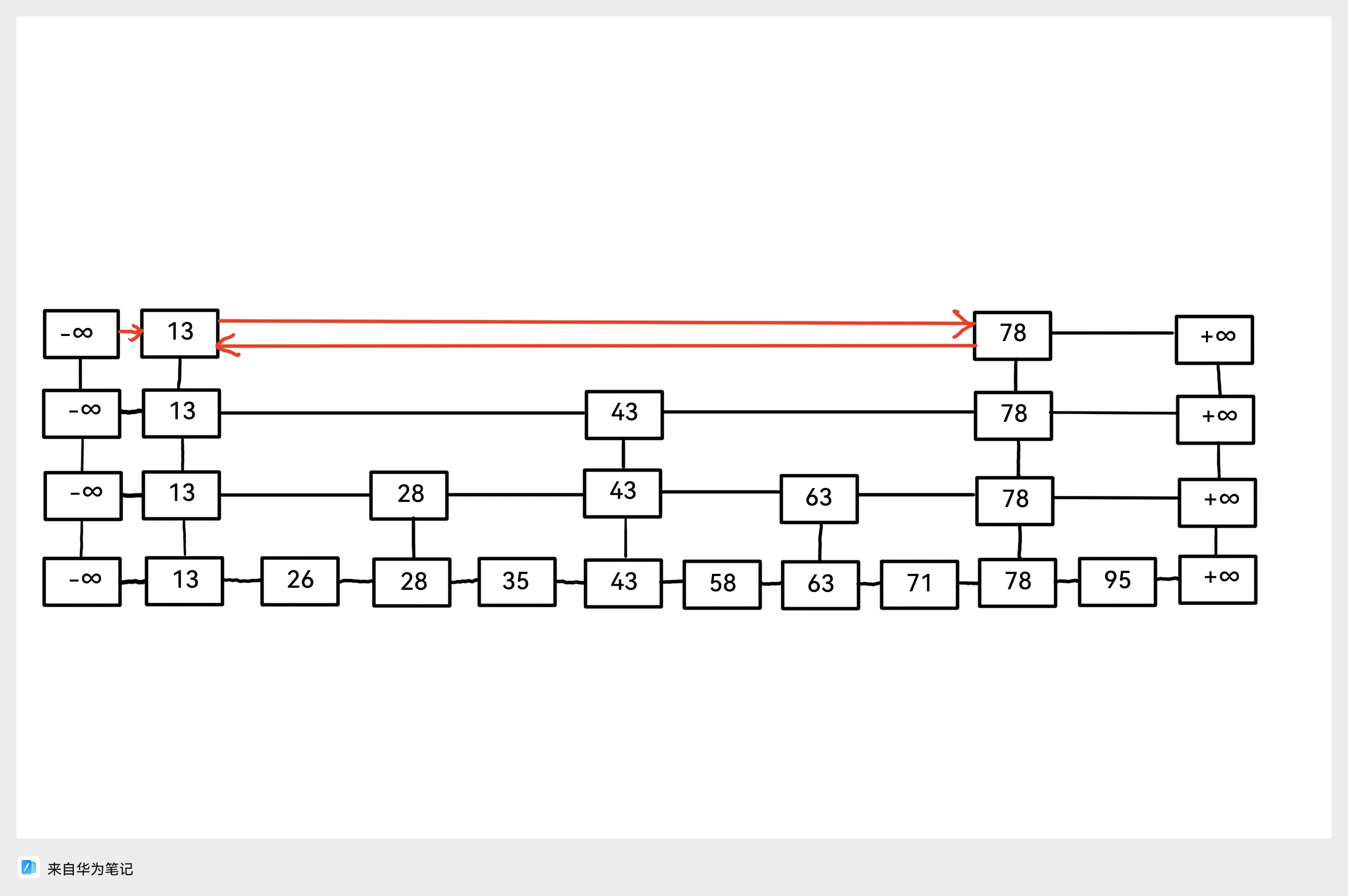
**Picture 2. The example of solution**

**Solution:**

1.1

****

1.2

****

**Problem 2.[30 points]**

2.1 Please design the data structure of the nodes in the skip list and the data structure of skip list separately.

2.2 Please give the code for searching operation in the skip list. Please give the effective C code directly instead of pseudo code whose name should be Skip\_List\_Search().

**Solution:**

2.1 The data structure of nodes in the skip list.

typedef struct node

{

int key;

struct node \*next[1];

} Node;

The data structure of the skip list.

typedef struct skiplist

{

int level;

Node \*head;

} Skiplist;

2.2 The code of Skip\_List\_Search().

Node \*Skip\_List\_Search(Skiplist \*sl, int key)

{

Node \*q=NULL, \*p=sl->head;

for(int i=sl->level-1; i>=0; i--)

{

while((q=p->next[i]) && q->key<key)

p=q;

if(q && q->key==key)

return q;

}

return NULL;

}

**Problem 3.[40 points]**

You are need to give an algorithm to realize the following searching in an skip list: Given the key j and a pointer to the node , and . You need to find the node in expected time and , where k is the distance between and

3.1 You are allowed to add an element to the data structure of the node and you need to explain the purpose of adding the new element.

3.3 Please introduce your algorithm.

3.2 Please analyse the time complexity of your algorithm.

**Solution:**

3.1 To each node, add a pointer up, to point at the parent in the level above.

3.2 Skip list with the numbers [1, 2...n] in the lowest level S1 and (-∞, ∞) in the highest level Sh. Start with the given pointer to element x in level S1, use the 'up' pointers in order to move to a higher level, until the current node’s key is ≤ j and it's next element's key is > j. From this point start to search the element j in a the smallest segment in the Skip List, that includes both elements x and y.

3.3 Complexity: The part of the skip list, where we perform the operations mentioned above is of an approximate size of k. It can be a bit bigger than k in case we found a node with key<j. Therefore, we'll assume that the list is of size 2k in average (just in case). The part of skip list based on the 2k is a tree with height O(logk). So the search on the part of the skip list is like performing the operation on a skip list of size O(k). Thus, the search time is O(logk) time.