REPORT

Data Structures

Student ID : 1771008
Name : Minjeong Kim

Homework9_1

변수(variable) 분석

element	
Type	**name**
char[KEY_SIZE]	key

ListNode	
Type	**name**
element	item
ListNode*	link

ListNode *hash_table[table_size]

13의 bucket number을 갖고있는 hash_table을 생성한다.
각 hashtable의 인자로는 element가 들어가고 이들은 Linked list로 연결되어있다.

함수(function) 분석

int transform(char *key) / int hash_function(char *key)
hash_table에 넣을 node의 key값을 ASCII 코드를 이용해서 숫자로 변환하고, table의 bucket 숫자에 맞게 숫자를 재 변환하는 함수이다.

void hash_chain_add(element item, ListNode *ht[])
transform, has_function을 이용해서 인자로 들어온 item을 숫자로 변환하고, 해당하는 숫자의 index를 가진 hash_table의 bucekt으로 element를 linked list에 연결한다.
즉, 인자로 들어온 item이 hash_table이 가진 13개의 bucket중 어디에 들어갈지를 정하고(encoding과 비슷한 개념, 대응의 개념) 해당하는 bucket에 연결한다.
항상 말하지만 linked list는 linked list의 첫 시작일 때와 일반적일 때를 잘 구분해서 넣도록 하자!

void hash_chain_search(element item, ListNode *ht[])
인자로 들어온 item의 bucket num을 hash_function을 이용해서 확인하고, 해당하는 bucket의 linked list를 쭉 따라 가면서 item과 bucket의 각 node의 item과 같은 것이 있는지 조건문으로 확인한다. 만약 있다면 search success이고, 아닐 경우 search failed이다.

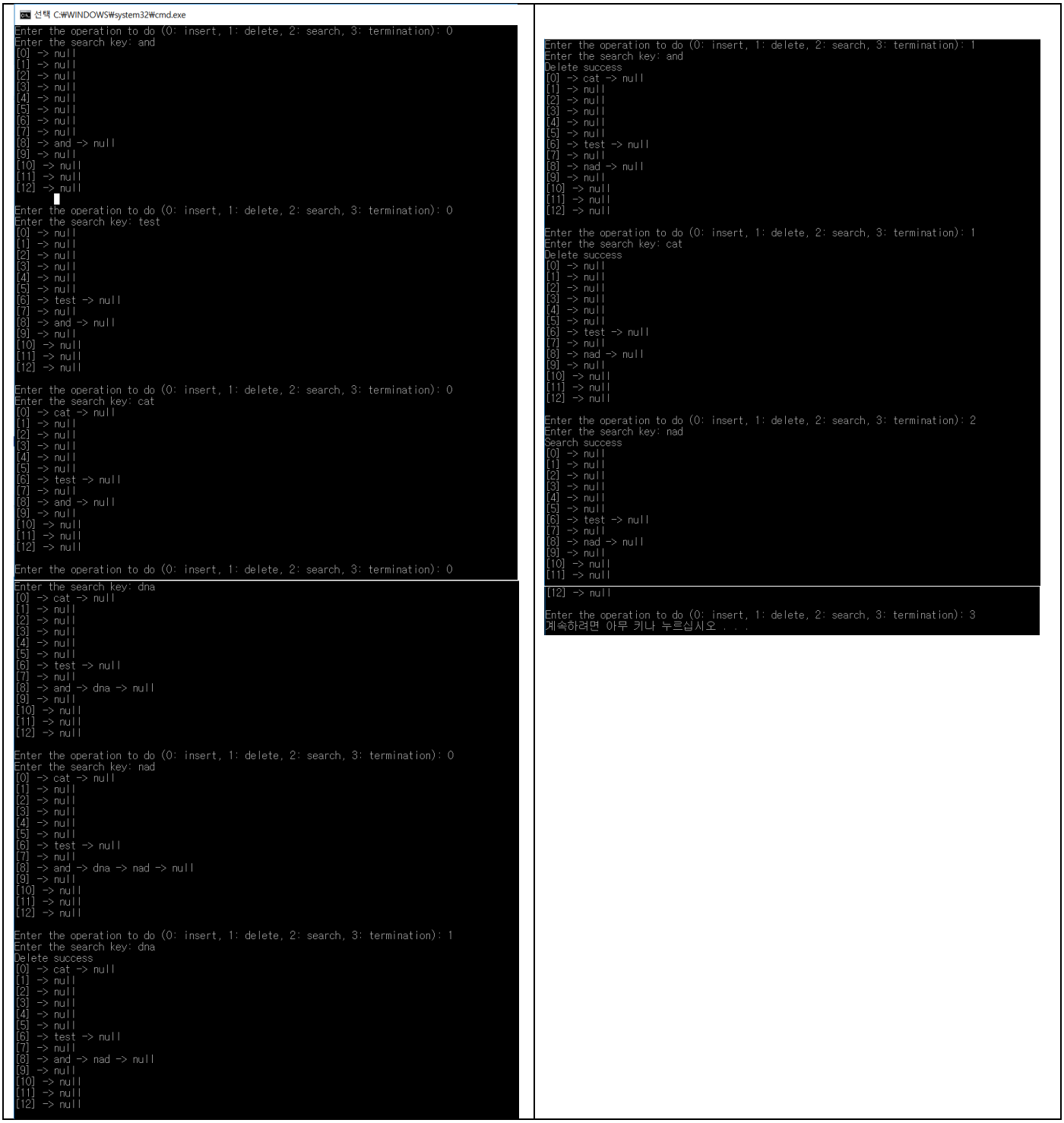
void hash_chain_delete(element item, ListNode *ht[])
delete를 하기 위해서는 먼저, 해당하는 item이 hash_table에 있는지 확인해야한다. 따라서 코드가 search와 유사한 부분이 있다. search와 같은 과정으로 인자로 들어오 item의 bucket num을 확인하고, bucket의 linked list를 따라가면서 item과 bucket의 각 node의 item과 같은 것이 있는지 조건문으로 확인한다. 이때 일치한 것이 있으면 삭제하고, 없으면 delete 가 실패했음을 알려준다.
삭제를 할 때도 추가할 때와 마찬가지로 linked list의 첫 부분과 일반적일 때를 잘 구분해서 삭제 해야 하기 때문에 조건문을 사용한다.

전체 코드(full code)

```c
#define KEY_SIZE 10
#define TABLE_SIZE 13

#include <stdio.h>
#include <string.h>
#include <stdlib.h>

typedef struct element {
    char key[KEY_SIZE];
} element;

typedef struct ListNode {
    element item;
    ListNode *link;
} ListNode;

ListNode *hash_table[TABLE_SIZE]; //13 mod

// Transform the string key into an integer by summing ASCII codes
int transform(char *key) {
    int number = 0;
    while (*key)
        number += *key++;
    return number;
} // Division function (key mod TABLE_SIZE )

int hash_function(char *key) {
    return transform(key) % TABLE_SIZE;
}


bool equal(element e1,element e2) {
    return !strcmp(e1.key, e2.key);
}


void hash_chain_add(element item, ListNode *ht[])
{
    int hash_value = hash_function(item.key); //mod 한 것!
    ListNode *ptr;
    ListNode *node_before = NULL;
    ListNode *node = ht[hash_value];

    for ( ; node; node_before = node, node = node->link){ //같은키가 있으면 안된다. //ㄱ접근한 node가 null이 아닐 때 까지.
        if (equal(node->item, item)) {
            fprintf(stderr, "Duplicate search key\n");
            return;
        }
    }

    ptr =(ListNode *)malloc(sizeof(ListNode));
    ptr->item = item;
    ptr->link = NULL;
    if (node_before)
        node_before->link = ptr;
    else //첫 시작일 때.
        ht[hash_value] = ptr;
}

void hash_chain_search(element item, ListNode *ht[])
{
    ListNode *node;
    int hash_value = hash_function(item.key);
    for (node = ht[hash_value]; node; node = node->link) {
        if (equal(node->item, item)) {
            printf("Search success\n");
            return;
        }
    }
    printf("Search failed\n");
}

void hash_chain_delete(element item, ListNode *ht[]) {


    int hast_value = hash_function(item.key);
    ListNode *node;
    ListNode *node_before = NULL;
    for (node = ht[hast_value]; node; node_before = node, node = node->link) {
        if (equal(node->item, item)) { //일치한 것이 있을 때.
            if (node_before == NULL) { //만약 제일 앞에있는 것을 삭제할 때.
                ht[hast_value] = node->link;
            }
            else {
```

```
85                  node_before->link = node->link;
86              }
87              printf("Delete success\n");
88              return;
89          }
90      }
91      //일치한 것이 없을 때.
92      fprintf(stderr, "there are not delete key\n");
93  }
94
95  void hash_chain_print(ListNode *ht[])
96  {
97      ListNode *node;
98      for (int i = 0; i < TABLE_SIZE; i++) {
99          printf("[%d]", i);
100         for (node = ht[i]; node; node = node->link)
101             printf(" -> %s", node->item.key);
102         printf(" -> null\n");
103     }
104 }
105
106 void init_table(ListNode *ht[])
107 {
108     for (int i = 0; i < TABLE_SIZE; i++)
109         ht[i] = NULL;//each node is initialized as null
110 }
111
112 int main()
113 {
114     element tmp;
115     int op;
116     init_table(hash_table);
117
118     while (1) {
119         printf("Enter the operation to do (0: insert, 1: delete, 2: search, 3: termination): ");
120         scanf_s("%d", &op);
121
122         if (op == 3) break;
123
124         printf("Enter the search key: ");
125         scanf_s("%s", tmp.key, sizeof(tmp.key));
126
127         if (op == 0) {
128             hash_chain_add(tmp, hash_table);
129         }
130         else if (op == 1) {
131             hash_chain_delete(tmp, hash_table);
132         }
133         else if (op == 2) {
134             hash_chain_search(tmp, hash_table);
135         }
136
137         hash_chain_print(hash_table);
138         printf("\n");
139     }
140
141 }
```

결과(output)



```
선택 C:₩WINDOWS₩system32₩cmd.exe
Enter the operation to do (0: insert, 1: delete, 2: search, 3: termination): 0
Enter the search key: and
[0] -> null
[1] -> null
[2] -> null
[3] -> null
[4] -> null
[5] -> null
[6] -> null
[7] -> null
[8] -> and -> null
[9] -> null
[10] -> null
[11] -> null
[12] -> null

Enter the operation to do (0: insert, 1: delete, 2: search, 3: termination): 0
Enter the search key: test
[0] -> null
[1] -> null
[2] -> null
[3] -> null
[4] -> null
[5] -> null
[6] -> test -> null
[7] -> null
[8] -> and -> null
[9] -> null
[10] -> null
[11] -> null
[12] -> null

Enter the operation to do (0: insert, 1: delete, 2: search, 3: termination): 0
Enter the search key: cat
[0] -> cat -> null
[1] -> null
[2] -> null
[3] -> null
[4] -> null
[5] -> null
[6] -> test -> null
[7] -> null
[8] -> and -> null
[9] -> null
[10] -> null
[11] -> null
[12] -> null

Enter the operation to do (0: insert, 1: delete, 2: search, 3: termination): 0
Enter the search key: dna
[0] -> cat -> null
[1] -> null
[2] -> null
[3] -> null
[4] -> null
[5] -> null
[6] -> test -> null
[7] -> null
[8] -> and -> dna -> null
[9] -> null
[10] -> null
[11] -> null
[12] -> null

Enter the operation to do (0: insert, 1: delete, 2: search, 3: termination): 0
Enter the search key: nad
[0] -> cat -> null
[1] -> null
[2] -> null
[3] -> null
[4] -> null
[5] -> null
[6] -> test -> null
[7] -> null
[8] -> and -> dna -> nad -> null
[9] -> null
[10] -> null
[11] -> null
[12] -> null

Enter the operation to do (0: insert, 1: delete, 2: search, 3: termination): 1
Enter the search key: dna
Delete success
[0] -> cat -> null
[1] -> null
[2] -> null
[3] -> null
[4] -> null
[5] -> null
[6] -> test -> null
[7] -> null
[8] -> and -> nad -> null
[9] -> null
[10] -> null
[11] -> null
[12] -> null
```

```
Enter the operation to do (0: insert, 1: delete, 2: search, 3: termination): 1
Enter the search key: and
Delete success
[0] -> cat -> null
[1] -> null
[2] -> null
[3] -> null
[4] -> null
[5] -> null
[6] -> test -> null
[7] -> null
[8] -> nad -> null
[9] -> null
[10] -> null
[11] -> null
[12] -> null

Enter the operation to do (0: insert, 1: delete, 2: search, 3: termination): 1
Enter the search key: cat
Delete success
[0] -> null
[1] -> null
[2] -> null
[3] -> null
[4] -> null
[5] -> null
[6] -> test -> null
[7] -> null
[8] -> nad -> null
[9] -> null
[10] -> null
[11] -> null
[12] -> null

Enter the operation to do (0: insert, 1: delete, 2: search, 3: termination): 2
Enter the search key: nad
Search success
[0] -> null
[1] -> null
[2] -> null
[3] -> null
[4] -> null
[5] -> null
[6] -> test -> null
[7] -> null
[8] -> nad -> null
[9] -> null
[10] -> null
[11] -> null
[12] -> null

Enter the operation to do (0: insert, 1: delete, 2: search, 3: termination): 3
계속하려면 아무 키나 누르십시오 . . .
```

Homework9_2

변수(variable) 분석

TreeNode	
Type	**name**
int	key
TreeNode*	left
TreeNode*	right

TreeType	
Type	**name**
TreeNode*	root

treeType의 root는 하나의 binary search tree의 root node의 주소를 나타낸다.

함수(function) 분석

void insert_node(TreeType *br, int key)

binary search tree의 특성에 맞게, temporary node의 key값이 parameter로 들어온 key값하고 큰지 작은지를 비교해서 새로 추가할 node의 위치를 먼저 탐색하고, 그 위치에, 새로운 treenode를 추가한다.

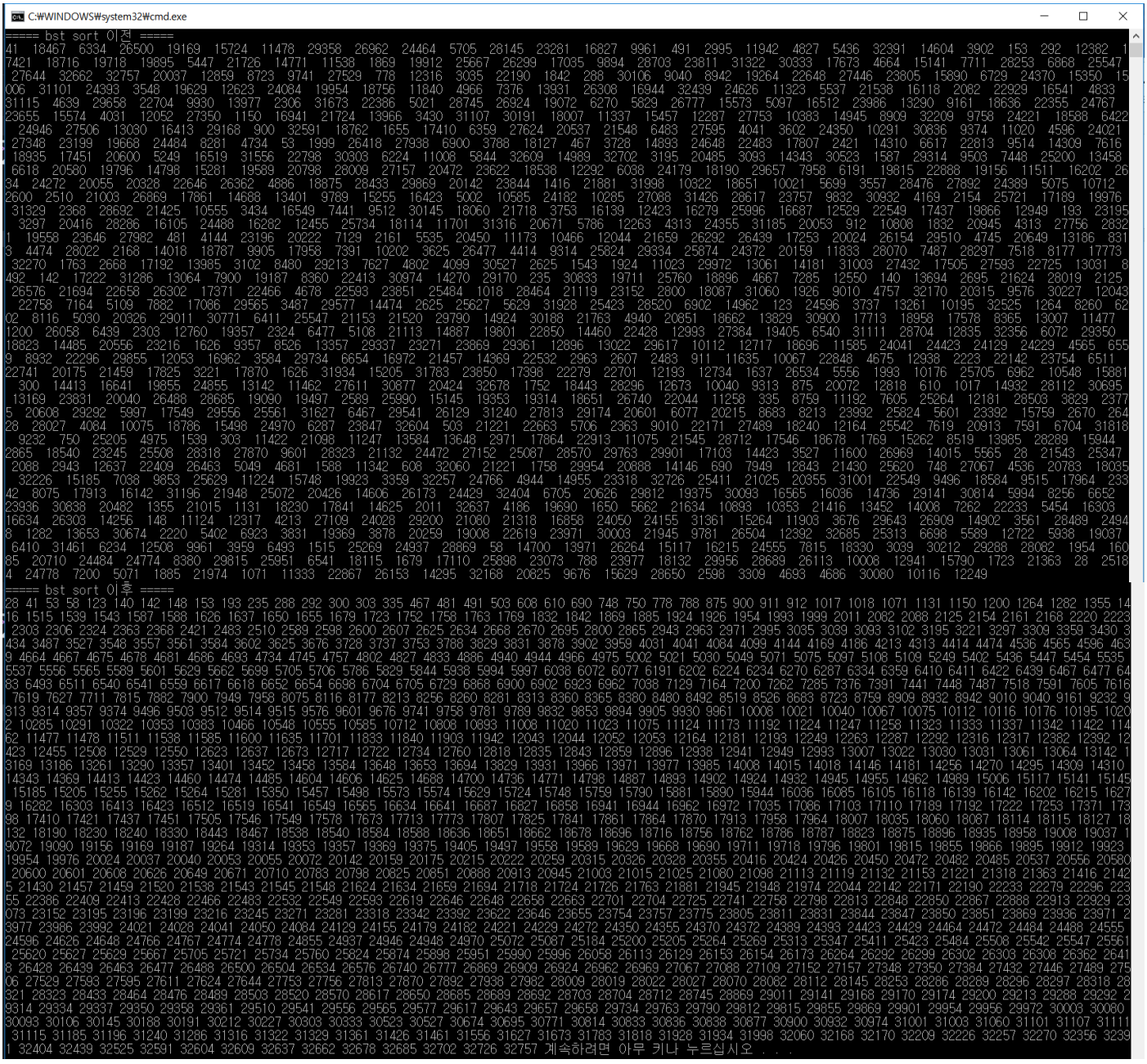
void inorder(TreeNode *node)

recursion을 이용해서 traversal을 구현한다. 이때 binary search tree는 현재 node의 key보다 작으면 왼쪽, 크면 오른쪽이라는 특징을 갖고 있기 때문에 전체적으로 L < V < R 의 형태이다. 따라서 L -> V -> R의 순서로 트리를 탐색한다면 작은 수부터 큰 수까지 sorting을 할 수 있다.

전체 코드(full code)

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

 # define RAND_MAX 10000;

typedef struct TreeNode {
    int key;
    struct TreeNode* left;
    struct TreeNode* right;
}TreeNode;

typedef struct TreeType {
    TreeNode* root;
}TreeType;

void init(TreeType* bt) {
    bt->root = NULL;
}

void insert_node(TreeType *bt, int key) {
    TreeNode *p, *t;
    TreeNode *n;
    t = bt->root;
    p = NULL;

    while (t != NULL) {
        if (key == t->key) {
            return;
        }
        else {
            p = t;
            if (key < t->key)
                t = t->left;
            else
                t = t->right;
        }
    }

    n = (TreeNode *)malloc(sizeof(TreeNode));
    if (n == NULL) return;

    n->key = key;
    n->left = n->right = NULL;

    if (p != NULL) {
        if (key < p->key)
            p->left = n;
        else p->right = n;
    }
    else {
        bt->root = n;
    }
}

void inorder(TreeNode* node) {
    if (node != NULL) {
        inorder(node->left);
        printf("%d ", node->key);
        inorder(node->right);
    }
}

int main() {
    int input_size = 1000;

    // output: sorted result
    int *input = (int *)malloc(sizeof(int)*input_size);

    TreeType* bt = (TreeType *)malloc(sizeof(TreeType));
    init(bt);

    // Generate an input data randomly
    printf("===== bst sort 이전 =====\n");
    for (int i = 0; i < input_size; i++) {
        input[i] = rand();
        printf("%d  ", input[i]);
        insert_node(bt, input[i]);
    }

    printf("\n===== bst sort 이전 =====\n");
    inorder(bt->root);
}
```

결과(output)


C:₩WINDOWS₩system32₩cmd.exe
===== bst sort 이전 =====
41 18467 6334 26500 19169 15724 11478 29358 26962 24464 5705 28145 23281 16827 9961 491 2995 11942 4827 5436 32391 14604 3902 153 292 12382 17421 18716 19718 19895 5447 21726 14771 11538 1869 19912 25667 26299 17035 9894 28703 23811 31322 30333 17673 4664 15141 7711 28253 6868 25547 27644 32662 32757 20037 12859 8723 9741 27529 778 12316 3035 22190 1842 288 30106 9040 8942 19264 22648 27446 23805 15890 6729 24370 15350 15006 31101 24393 3548 19629 12623 24084 19954 18756 11840 4966 7376 13931 26308 16944 32439 24626 11323 5537 21538 16118 2082 22929 16541 4833 31115 4639 29658 22704 9930 13977 2306 31673 22386 5021 28745 26924 19072 6270 5829 26777 15573 5097 16512 23986 13290 9161 18636 22355 24767 23655 15574 4031 12052 27350 1150 16941 21724 13966 3430 31107 30191 18007 11337 15457 12287 27753 10383 14945 8909 32209 9758 24221 18588 6422 24946 27506 13030 16413 29168 900 32591 18762 1655 17410 6359 27624 20537 21548 6483 27595 4041 3602 24350 10291 30836 9374 11020 4596 24021 27348 23199 19668 24484 8281 4734 53 1999 26418 27938 6900 3788 18127 467 3728 14893 24648 22483 17807 2421 14310 6617 22813 9514 14309 7616 18935 17451 20600 5249 16519 31556 22798 30303 6224 11008 5844 32609 14989 32702 3195 20485 3093 14343 30523 1587 29314 9503 7448 25200 13458 6618 20580 19796 14798 15281 19589 20798 28009 27157 20472 23622 18538 12292 6038 24179 18190 29657 7958 6191 19815 22888 19156 11511 16202 26034 24272 20055 20328 22646 26362 4886 18875 28433 29869 20142 23844 1416 21881 31998 10322 18651 10021 5699 3557 28476 27892 24389 5075 10712 2600 2510 21003 26869 17861 14688 13401 9789 15255 16423 5002 10585 24182 10285 27088 31426 28617 23757 9832 30932 4169 2154 25721 17189 19976 31329 2368 28692 21425 10555 3434 16549 7441 9512 30145 18060 21718 3753 16139 12423 16279 25996 16687 12529 22549 17437 19866 12949 193 23195 3297 20416 28286 16105 24488 16282 12455 25734 18114 11701 31316 20671 5786 12263 4313 24355 31185 20053 912 10608 1832 20945 4313 27756 28321 19558 23646 27982 481 4144 23196 20222 7129 2161 5535 20450 11173 10466 12044 21659 26292 26439 17253 20024 26154 29510 4745 20649 13186 8313 4474 28022 2168 14018 18787 9905 17958 7391 10202 3625 26477 4414 9314 25824 29334 25874 24372 20159 11833 28070 7487 28297 7518 8177 17773 32270 1763 2668 17192 13985 3102 8480 29213 7627 4802 4099 30527 2625 1543 1924 11023 29972 13061 14181 31008 27432 17505 27593 22725 13081 8492 142 17222 31286 13064 7900 19187 8360 22413 30974 14270 29170 235 30833 19711 25760 18896 4667 7285 12550 140 13694 2695 21624 28019 2125 26576 21694 22658 26302 17371 22466 4678 22593 23851 25484 1018 28464 21119 23152 2800 18087 31060 1926 9010 4757 32170 20315 9576 30227 12043 22758 7164 5109 7882 17086 29565 3487 29577 14474 2625 25627 5629 31928 25423 28520 6902 14962 123 24596 3737 13261 10195 32525 1264 8260 6202 8116 5030 20326 29011 30771 6411 25547 21153 21520 29790 14924 30188 21763 4940 20851 18662 13829 30900 17713 18958 17578 8365 13007 11477 1200 26058 6439 2303 12760 19357 2324 6477 5108 21113 14887 19801 22850 14460 22428 12993 27384 19405 6540 31111 28704 12835 32356 6072 29350 18823 14485 20556 23216 1626 9357 8526 13357 29337 23271 23869 29361 12896 13022 29617 10112 12717 18696 11585 24041 24423 24129 24229 4565 6559 8932 22296 29855 12053 16962 3584 29734 6654 16972 21457 14369 22532 2963 2607 2483 911 11635 10067 22848 4675 12938 2223 22142 23754 6511 22741 20175 21459 17825 3221 17870 1626 31934 15205 31783 23850 17398 22279 22701 12193 12734 1637 26534 5556 1993 10176 25705 6962 10548 15881 300 14413 16641 19855 24855 13142 11462 27611 30877 20424 32678 1752 18443 28296 12673 10040 9313 875 20072 12818 610 1017 14932 28112 30695 13169 23831 20040 26488 28685 19090 19497 2589 25990 15145 19353 19314 18651 26740 22044 11258 335 8759 11192 7605 25264 12181 28503 3829 23775 20608 29292 5997 17549 29556 25561 31627 6467 29541 26129 31240 27813 29174 20601 6077 20215 8683 8213 23992 25824 5601 23392 15759 2670 26428 28027 4084 10075 18786 15498 24970 6287 23847 32604 503 21221 22663 5706 2363 9010 22171 27489 18240 12164 25542 7619 20913 7591 6704 31818 9232 750 25205 4975 1539 303 11422 21098 11247 13584 13648 2971 17864 22913 11075 21545 28712 17546 18678 1769 15262 8519 13985 28289 15944 2865 18540 23245 25508 28318 27870 9601 28323 21132 24472 27152 25087 28570 29763 29901 17103 14423 3527 11600 26969 14015 5565 28 21543 25347 2088 2943 12637 22409 26463 5049 4681 1588 11342 608 32060 21221 1758 29954 20888 14146 690 7949 12843 21430 25620 748 27067 4536 20783 18035 32226 15185 7038 9853 25629 11224 15748 19923 3359 32257 24766 4944 14955 23318 32726 25411 21025 20355 31001 22549 9496 18584 9515 17964 23342 8075 17913 16142 31196 21948 25072 20426 14606 26173 24429 32404 6705 20626 29812 19375 30093 16565 16036 14736 29141 30814 5994 8256 6652 23936 30838 20482 1355 21015 1131 18230 17841 14625 2011 32637 4186 19690 1650 5662 21634 10893 10353 21416 13452 14008 7262 22233 5454 16308 16634 26303 14256 148 11124 12317 4213 27109 24028 29200 21080 21318 16858 24050 24155 31361 15264 11903 3676 29643 26909 14902 3561 28489 24948 1282 13653 30674 2220 5402 6923 3831 19369 3878 20259 19008 22619 23971 30003 21945 9781 26504 12392 32685 25313 6698 5589 12722 5938 19037 6410 31461 6234 12508 9961 3959 6493 1515 25269 24937 28869 58 14700 13971 26264 15117 16215 24555 7815 18330 3039 30212 29288 28082 1954 16085 20710 24484 24774 8380 29815 25951 6541 18115 1679 17110 25898 23073 788 23977 18132 29956 28689 26113 10008 12941 15790 1723 21363 28 25184 24778 7200 5071 1885 21974 1071 11333 22867 26153 14295 32168 20825 9676 15629 28650 2598 3309 4693 4686 30080 10116 12249
===== bst sort 이후 =====
28 41 53 58 123 140 142 148 153 193 235 288 292 300 303 335 467 481 491 503 608 610 690 748 750 778 788 875 900 911 912 1017 1018 1071 1131 1150 1200 1264 1282 1355 1416 1515 1539 1543 1587 1588 1626 1637 1650 1655 1679 1723 1752 1758 1763 1769 1832 1842 1869 1885 1924 1926 1954 1993 1999 2011 2082 2088 2125 2154 2161 2168 2220 2223 2303 2306 2324 2363 2368 2421 2483 2510 2589 2598 2600 2607 2625 2634 2668 2670 2695 2800 2865 2943 2963 2971 2995 3035 3039 3093 3102 3195 3221 3297 3309 3359 3430 3434 3487 3527 3548 3557 3561 3584 3602 3625 3676 3728 3737 3753 3788 3829 3831 3878 3902 3959 4031 4041 4084 4099 4144 4169 4186 4213 4313 4414 4474 4536 4565 4596 4639 4664 4667 4675 4678 4681 4686 4693 4734 4745 4757 4802 4827 4833 4886 4940 4944 4966 4975 5002 5021 5030 5049 5071 5075 5097 5108 5109 5249 5402 5436 5447 5454 5535 5537 5556 5565 5589 5601 5629 5662 5699 5705 5706 5786 5829 5844 5938 5994 5997 6038 6072 6077 6191 6202 6224 6234 6270 6287 6334 6359 6410 6411 6422 6439 6467 6477 6483 6493 6511 6540 6541 6559 6617 6618 6652 6654 6698 6704 6705 6729 6868 6900 6902 6923 6962 7038 7129 7164 7200 7262 7285 7376 7391 7441 7448 7487 7518 7591 7605 7616 7619 7627 7711 7815 7882 7900 7949 7958 8075 8116 8177 8213 8256 8260 8281 8313 8360 8365 8380 8480 8492 8519 8526 8683 8723 8759 8909 8932 8942 9010 9040 9161 9232 9313 9314 9357 9374 9496 9503 9512 9514 9515 9576 9601 9676 9741 9758 9781 9789 9832 9853 9894 9905 9930 9961 10008 10021 10040 10067 10075 10112 10116 10176 10195 10202 10285 10291 10322 10353 10383 10466 10548 10555 10585 10712 10808 10893 11008 11020 11023 11075 11124 11173 11192 11224 11247 11258 11323 11333 11337 11342 11422 11462 11477 11478 11511 11538 11585 11600 11635 11701 11833 11840 11903 11942 12043 12044 12052 12053 12164 12181 12193 12249 12263 12287 12292 12316 12317 12382 12392 12423 12455 12508 12529 12550 12623 12637 12673 12717 12722 12734 12760 12818 12835 12843 12859 12896 12938 12941 12949 12993 13007 13022 13030 13031 13061 13064 13142 13169 13186 13261 13290 13357 13401 13452 13458 13584 13648 13653 13694 13829 13931 13966 13971 13977 13985 14008 14015 14018 14146 14181 14256 14270 14295 14309 14310 14343 14369 14413 14423 14460 14474 14485 14604 14606 14625 14688 14700 14736 14771 14798 14887 14893 14902 14924 14932 14945 14955 14962 14989 15006 15117 15141 15145 15185 15205 15255 15262 15264 15281 15350 15457 15498 15573 15574 15629 15724 15748 15759 15790 15881 15890 15944 16036 16085 16105 16118 16139 16142 16202 16215 16279 16282 16303 16413 16423 16512 16519 16541 16549 16565 16634 16641 16687 16827 16858 16941 16944 16962 16972 17035 17086 17103 17110 17189 17192 17222 17253 17371 17398 17410 17421 17437 17451 17505 17546 17549 17578 17673 17713 17773 17807 17825 17841 17861 17864 17870 17913 17958 17964 18007 18035 18060 18087 18114 18115 18127 18132 18190 18230 18240 18330 18443 18467 18538 18540 18584 18588 18636 18651 18662 18678 18696 18716 18756 18762 18786 18787 18823 18875 18896 18935 18958 19008 19037 19072 19090 19156 19169 19187 19264 19314 19353 19357 19369 19375 19405 19497 19558 19589 19629 19668 19690 19711 19718 19796 19801 19815 19855 19866 19895 19912 19923 19954 19976 20024 20037 20040 20053 20055 20072 20142 20159 20175 20215 20222 20259 20315 20326 20328 20355 20416 20424 20426 20450 20472 20482 20485 20537 20556 20580 20600 20601 20608 20626 20649 20671 20710 20783 20798 20825 20851 20888 20913 20945 21003 21015 21025 21080 21098 21113 21119 21132 21153 21221 21318 21363 21416 21425 21430 21457 21459 21520 21538 21543 21545 21548 21624 21634 21659 21694 21718 21724 21726 21763 21881 21945 21948 21974 22044 22142 22171 22190 22233 22279 22296 22355 22386 22409 22413 22428 22466 22483 22532 22549 22593 22619 22646 22648 22658 22663 22701 22704 22725 22741 22758 22798 22813 22848 22850 22867 22888 22913 22929 23073 23152 23195 23196 23199 23216 23245 23271 23281 23318 23342 23392 23622 23646 23655 23754 23757 23775 23805 23811 23831 23844 23847 23850 23851 23869 23936 23971 23977 23986 23992 24021 24028 24041 24050 24084 24129 24155 24179 24182 24221 24229 24272 24350 24355 24370 24372 24389 24393 24423 24429 24464 24472 24484 24488 24555 24596 24626 24648 24766 24767 24774 24778 24855 24937 24946 24948 24970 25072 25087 25184 25200 25205 25264 25269 25313 25347 25411 25423 25484 25508 25542 25547 25561 25620 25627 25629 25667 25705 25721 25734 25760 25824 25874 25898 25951 25990 25996 26058 26113 26129 26153 26154 26173 26264 26292 26299 26302 26303 26308 26362 26418 26428 26439 26463 26477 26488 26500 26504 26534 26576 26740 26777 26869 26909 26924 26962 26969 27067 27088 27109 27152 27157 27348 27350 27384 27432 27446 27489 27506 27529 27593 27595 27611 27624 27644 27753 27756 27813 27870 27892 27938 27982 28009 28019 28022 28027 28070 28082 28112 28145 28253 28286 28289 28296 28297 28318 28321 28323 28433 28464 28476 28489 28503 28520 28570 28617 28650 28685 28689 28692 28703 28704 28712 28745 28869 29011 29141 29168 29170 29174 29200 29213 29288 29292 29314 29334 29337 29350 29358 29361 29510 29541 29556 29565 29577 29617 29643 29657 29658 29734 29763 29790 29812 29815 29855 29869 29901 29954 29956 29972 30003 30080 30093 30106 30145 30188 30191 30212 30227 30303 30333 30523 30527 30674 30695 30771 30814 30833 30836 30838 30877 30900 30932 30974 31001 31003 31060 31101 31107 31111 31115 31185 31196 31240 31286 31316 31322 31329 31361 31426 31461 31556 31627 31673 31783 31818 31928 31934 31998 32060 32168 32170 32209 32226 32257 32270 32356 32391 32404 32439 32525 32591 32604 32609 32637 32662 32678 32685 32702 32726 32757 계속하려면 아무 키나 누르십시오 . . .