```
    double total_time = ((double) total*1000) / CLOCKS_PER_SEC;
    fprintf(fptr, "TREAP = %0.4lfms\t(%d rotations)\n", total_time/g_average,
g_rotation_count/g_average);
}

void tree_treap_inorder_print(Treap *treap, idx_t root) {
    if (root == IDX_INVALID) return;
    tree_treap_inorder_print(treap, treap->nodes[root].left);
    printf("%d ", treap->nodes[root].key);
    tree_treap_inorder_print(treap, treap->nodes[root].right);
}

int
main(int argc, char *argv[]) {

    if (argc != 3) {
        puts("aed-prej2 [treesize] [average]");
        exit(EXIT_FAILURE);
    }

    g_treesize = atoi(argv[1]);
    g_average = atoi(argv[2]);

    if (g_treesize < 0) {
        puts("Invalid tree size.");
        exit(EXIT_FAILURE);
    } else if (g_average < 0) {
        puts("Invalid average.");
        exit(EXIT_FAILURE);
    }

    srand(SEED);

    key_t *conjunto_a = arr_gen_conj_a(g_treesize);
    key_t *conjunto_b = arr_gen_conj_b(g_treesize);
    key_t *conjunto_c = arr_gen_conj_c(g_treesize);
    key_t *conjunto_d = arr_gen_conj_d(g_treesize);

    FILE* filelog = fopen("log.txt", "a");
    fprintf(filelog, "\n=== NEW LOG === (Treesize = %d, Average = &d)\n", g_treesize, g_average);

    puts("Testing binary search tree...");
    binary_test_and_log(conjunto_a, filelog);
    binary_test_and_log(conjunto_b, filelog);
    binary_test_and_log(conjunto_c, filelog);
    binary_test_and_log(conjunto_d, filelog);

    puts("Testing AVL tree...");
    avl_test_and_log(conjunto_a, filelog);
    avl_test_and_log(conjunto_b, filelog);
    avl_test_and_log(conjunto_c, filelog);
    avl_test_and_log(conjunto_d, filelog);

    puts("Testing Red-Black tree...");
    rb_test_and_log(conjunto_a, filelog);
    rb_test_and_log(conjunto_b, filelog);
    rb_test_and_log(conjunto_c, filelog);
    rb_test_and_log(conjunto_d, filelog);

    puts("Testing RB search tree...");
```