词法分析器测试报告

随机选取leetcode上的题解进行测试

```
int arrangeCoins2(int n){
    int i;
    long sum = 0;
    for (i = 1; i <= n; i++) {
        sum += i;
        if (sum > n) {
            return i - 1;
        }
    }
    return 1;
}

int arrangeCoins(int n){
    int left, right, mid;
    long sum;
    left = 1;
    right = n;
    while (left <= right) {
        mid = left + (right - left) / 2;
        sum = (long)mid * (1 + mid) / 2;
        if (sum <= n) {
            left = mid + 1;
        } else {
            right = mid - 1;
        }
    }
    return left - 1;
}
```

🤗 词法分析结果为：

```
<int, KEY, 17>
<arrangeCoins2, ID, 0>
<(, SPE, 0>
<int, KEY, 17>
<n, ID, 0>
<), SPE, 1>
<{, SYM, 0>
<int, KEY, 17>
<i, ID, 0>
<;, SYM, 5>
<long, KEY, 18>
<sum, ID, 0>
<=, ASS, 0>
<0, NUMI, 0>
<;, SYM, 5>
<for, KEY, 13>
<(, SPE, 0>
<i, ID, 0>
<=, ASS, 0>
<1, NUMI, 1>
<;, SYM, 5>
<i, ID, 0>
<<=, REL, 1>
<n, ID, 0>
<;, SYM, 5>
<i, ID, 0>
<++, ARI, 5>
<+, ARI, 1>
<), SPE, 1>
<{, SYM, 0>
<sum, ID, 0>
<+=, ASS, 1>
<=, ARI, 1>
<i, ID, 0>
<;, SYM, 5>
<if, KEY, 15>
<(, SPE, 0>
<sum, ID, 0>
<>, REL, 2>
<n, ID, 0>
<), SPE, 1>
<{, SYM, 0>
<return, KEY, 21>
<i, ID, 0>
<-, ARI, 1>
<1, NUMI, 1>
<;, SYM, 5>
<}, SYM, 0>
<}, SYM, 0>
<return, KEY, 21>
<1, NUMI, 1>
<;, SYM, 5>
<}, SYM, 0>
<int, KEY, 17>
<arrangeCoins, ID, 0>
<(, SPE, 0>
<int, KEY, 17>
<n, ID, 0>
<), SPE, 1>
<{, SYM, 0>
<int, KEY, 17>
<left, ID, 0>
<,, SYM, 6>
<right, ID, 0>
<,, SYM, 6>
<mid, ID, 0>
<;, SYM, 5>
<long, KEY, 18>
<sum, ID, 0>
<;, SYM, 5>
<left, ID, 0>
<=, ASS, 0>
<1, NUMI, 1>
<;, SYM, 5>
<right, ID, 0>
<=, ASS, 0>
<n, ID, 0>
<;, SYM, 5>
<while, ID, 0>
<(, SPE, 0>
<left, ID, 0>
<<=, REL, 1>
<right, ID, 0>
<), SPE, 1>
<{, SYM, 0>
<mid, ID, 0>
<=, ASS, 0>
<left, ID, 0>
<+, ARI, 0>
< , ARI, 1>
<(, SPE, 0>
<right, ID, 0>
<-, ARI, 1>
<left, ID, 0>
<), SPE, 1>
</, ARI, 3>
<2, NUMI, 2>
<;, SYM, 5>
<sum, ID, 0>
<=, ASS, 0>
<(, SPE, 0>
<long, KEY, 18>
<), SPE, 1>
<mid, ID, 0>
<*, ARI, 2>
<(, SPE, 0>
<1, NUMI, 1>
<+, ARI, 0>
< , ARI, 1>
<mid, ID, 0>
<), SPE, 1>
</, ARI, 3>
<2, NUMI, 2>
<;, SYM, 5>
<if, KEY, 15>
<(, SPE, 0>
<sum, ID, 0>
<<=, REL, 1>
<n, ID, 0>
<), SPE, 1>
<{, SYM, 0>
```

```
        <left, ID, 0>
        <=, ASS, 0>
        <mid, ID, 0>
        <+, ARI, 0>
        < , ARI, 1>
        <1, NUMI, 1>
        <;, SYM, 5>
        <}, SYM, 0>
        <else, KEY, 9>
        <{, SYM, 0>
        <right, ID, 0>
        <=, ASS, 0>
        <mid, ID, 0>
        <-, ARI, 1>
        <1, NUMI, 1>
        <;, SYM, 5>
        <}, SYM, 0>
        <}, SYM, 0>
        <return, KEY, 21>
        <left, ID, 0>
        <-, ARI, 1>
        <1, NUMI, 1>
        <;, SYM, 5>
        <}, SYM, 0>


    该程序共有 28 行
    该程序的字符总数为 280 / 526
    各种记号的的个数为：
        KEY:    16
        ID:     39
        NUMI:   11
        NUMF:   0
        REL:    4
        ASS:    9
        BIT:  0
        LOG:    0
        ARI: 16
        SPE:    18
        NOT:    0
```

```
SYM:    32
SEL:    0
```

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     struct TreeNode *left;
 *     struct TreeNode *right;
 * };
 */
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     struct ListNode *next;
 * };
 */


/**
 * Note: The returned array must be malloced, assume caller calls free().
 */

int depth(struct TreeNode* tree)
{
    if (tree == NULL) {
        return 0;
    }
    int left = depth(tree->left);
    int right = depth(tree->right);
    return left > right ? (left + 1) : (right + 1);
}

struct ListNode** listOfDepth(struct TreeNode* tree, int* returnSize){
    *returnSize = 0;
    if (tree == NULL) {
        return NULL;
    }
    /* 层序遍历 队列 */
    /* 先计算深度 */
    int dep = depth(tree);
    struct ListNode** ret = (struct ListNode**)malloc(sizeof(struct ListNode*) * dep);

    int max_size = pow(2, dep - 1);
    struct TreeNode* queue[max_size * dep];
    queue[0] = tree;
    int start = 0; /* 队首 指向第一个元素 */
    int end = 1; /* 队尾 指向最后一个元素的下一个 */

    while (end > start) { /* 队列不为空就有值 */
        int old_start = start;
        int old_end = end;
        start = end; /* 更新队列 */
        /* 先把当前层入队，再把上一次出队 */
        for (int i = old_start; i < old_end; i++) {
            if (queue[i] == NULL) {
                continue;
            }
            if (queue[i]->left != NULL) {
                queue[end++] = queue[i]->left;
            }
            if (queue[i]->right != NULL) {
                queue[end++] = queue[i]->right;
            }
        }
        ret[*returnSize] = (struct ListNode*)malloc(sizeof(struct ListNode)); /* head */
        struct ListNode* head = ret[*returnSize];
        head->val = queue[old_start]->val;
        head->next = NULL;
        for (int i = old_start + 1; i < old_end; i++) {
            struct ListNode* current = (struct ListNode*)malloc(sizeof(struct ListNode));
            current->val = queue[i]->val;
            current->next = NULL;
            head->next = current;
            head = head->next;
        }
        (*returnSize)++;
    }

    return ret;
}
```

🤗 词法分析结果为：

```
<//*, NOT, 0>
<//*, NOT, 0>
<//*, NOT, 0>
<int, KEY, 17>
<depth, ID, 0>
<(, SPE, 0>
<struct, KEY, 26>
<TreeNode, ID, 0>
<*, ARI, 2>
<tree, ID, 0>
<), SPE, 1>
<{, SYM, 0>
<if, KEY, 15>
<(, SPE, 0>
<tree, ID, 0>
<==, REL, 4>
<NULL, ID, 0>
<), SPE, 1>
<{, SYM, 0>
<return, KEY, 21>
<0, NUMI, 0>
<;, SYM, 5>
<}, SYM, 0>
<int, KEY, 17>
<left, ID, 0>
<=, ASS, 0>
<depth, ID, 0>
<(, SPE, 0>
<tree, ID, 0>
<->, SPE, 4>
<left, ID, 0>
<), SPE, 1>
<;, SYM, 5>
<int, KEY, 17>
<right, ID, 0>
<=, ASS, 0>
<depth, ID, 0>
<(, SPE, 0>
<tree, ID, 0>
<->, SPE, 4>
<right, ID, 0>
<), SPE, 1>
<;, SYM, 5>
<return, KEY, 21>
<left, ID, 0>
<>, REL, 2>
<right, ID, 0>
<?, SEL, 0>
<(, SPE, 0>
<left, ID, 0>
<+, ARI, 0>
< , ARI, 1>
<1, NUMI, 1>
<), SPE, 1>
<:, SYM, 2>
<(, SPE, 0>
<right, ID, 0>
<+, ARI, 0>
< , ARI, 1>
<1, NUMI, 1>
<), SPE, 1>
<;, SYM, 5>
<}, SYM, 0>
<struct, KEY, 26>
<ListNode, ID, 0>
<*, ARI, 2>
<*, ARI, 2>
<listOfDepth, ID, 0>
<(, SPE, 0>
<struct, KEY, 26>
<TreeNode, ID, 0>
<*, ARI, 2>
<tree, ID, 0>
<,, SYM, 6>
<int, KEY, 17>
<*, ARI, 2>
<returnSize, ID, 0>
<), SPE, 1>
<{, SYM, 0>
<*, ARI, 2>
```

```
<returnSize, ID, 0>
<=, ASS, 0>
<0, NUMI, 0>
<;, SYM, 5>
<if, KEY, 15>
<(, SPE, 0>
<tree, ID, 0>
<==, REL, 4>
<NULL, ID, 0>
<), SPE, 1>
<{, SYM, 0>
<return, KEY, 21>
<NULL, ID, 0>
<;, SYM, 5>
<}, SYM, 0>
</*, NOT, 0>
</*, NOT, 0>
<int, KEY, 17>
<dep, ID, 0>
<=, ASS, 0>
<depth, ID, 0>
<(, SPE, 0>
<tree, ID, 0>
<), SPE, 1>
<;, SYM, 5>
<struct, KEY, 26>
<ListNode, ID, 0>
<*, ARI, 2>
<*, ARI, 2>
<ret, ID, 0>
<=, ASS, 0>
<(, SPE, 0>
<struct, KEY, 26>
<ListNode, ID, 0>
<*, ARI, 2>
<*, ARI, 2>
<), SPE, 1>
<malloc, ID, 0>
<(, SPE, 0>
<sizeof, KEY, 24>
<(, SPE, 0>
<struct, KEY, 26>
<ListNode, ID, 0>
<*, ARI, 2>
<), SPE, 1>
<*, ARI, 2>
<dep, ID, 0>
<), SPE, 1>
<;, SYM, 5>
<int, KEY, 17>
<max, ID, 0>
<_size, ID, 0>
<=, ASS, 0>
<pow, ID, 0>
<(, SPE, 0>
<2, NUMI, 2>
<,, SYM, 6>
<dep, ID, 0>
<-, ARI, 1>
<1, NUMI, 1>
<), SPE, 1>
<;, SYM, 5>
<struct, KEY, 26>
<TreeNode, ID, 0>
<*, ARI, 2>
<queue, ID, 0>
<[, SPE, 2>
<max, ID, 0>
<_size, ID, 0>
<*, ARI, 2>
<dep, ID, 0>
<], SPE, 3>
<;, SYM, 5>
<queue, ID, 0>
<[, SPE, 2>
<0, NUMI, 0>
<], SPE, 3>
<=, ASS, 0>
<tree, ID, 0>
<;, SYM, 5>
<int, KEY, 17>
<start, ID, 0>
```

```
<=, ASS, 0>
<0, NUMI, 0>
<;, SYM, 5>
</*, NOT, 0>
<int, KEY, 17>
<end, ID, 0>
<=, ASS, 0>
<1, NUMI, 1>
<;, SYM, 5>
</*, NOT, 0>
<while, ID, 0>
<(, SPE, 0>
<end, ID, 0>
<>, REL, 2>
<start, ID, 0>
<), SPE, 1>
<{, SYM, 0>
</*, NOT, 0>
<int, KEY, 17>
<old, ID, 0>
<_start, ID, 0>
<=, ASS, 0>
<start, ID, 0>
<;, SYM, 5>
<int, KEY, 17>
<old, ID, 0>
<_end, ID, 0>
<=, ASS, 0>
<end, ID, 0>
<;, SYM, 5>
<start, ID, 0>
<=, ASS, 0>
<end, ID, 0>
<;, SYM, 5>
</*, NOT, 0>
</*, NOT, 0>
<for, KEY, 13>
<(, SPE, 0>
<int, KEY, 17>
<i, ID, 0>
<=, ASS, 0>
```

```
<old, ID, 0>
<_start, ID, 0>
<;, SYM, 5>
<i, ID, 0>
<<, REL, 0>
<old, ID, 0>
<_end, ID, 0>
<;, SYM, 5>
<i, ID, 0>
<++, ARI, 5>
<+, ARI, 1>
<), SPE, 1>
<{, SYM, 0>
<if, KEY, 15>
<(, SPE, 0>
<queue, ID, 0>
<[, SPE, 2>
<i, ID, 0>
<], SPE, 3>
<==, REL, 4>
<NULL, ID, 0>
<), SPE, 1>
<{, SYM, 0>
<continue, KEY, 5>
<;, SYM, 5>
<}, SYM, 0>
<if, KEY, 15>
<(, SPE, 0>
<queue, ID, 0>
<[, SPE, 2>
<i, ID, 0>
<], SPE, 3>
<->, SPE, 4>
<left, ID, 0>
<!=, REL, 5>
<NULL, ID, 0>
<), SPE, 1>
<{, SYM, 0>
<queue, ID, 0>
<[, SPE, 2>
<end, ID, 0>
<++, ARI, 5>
<+, ARI, 1>
<], SPE, 3>
<=, ASS, 0>
<queue, ID, 0>
<[, SPE, 2>
<i, ID, 0>
<], SPE, 3>
<->, SPE, 4>
<left, ID, 0>
<;, SYM, 5>
<}, SYM, 0>
<if, KEY, 15>
<(, SPE, 0>
<queue, ID, 0>
<[, SPE, 2>
<i, ID, 0>
<], SPE, 3>
<->, SPE, 4>
<right, ID, 0>
<!=, REL, 5>
<NULL, ID, 0>
<), SPE, 1>
<{, SYM, 0>
<queue, ID, 0>
<[, SPE, 2>
<end, ID, 0>
<++, ARI, 5>
<+, ARI, 1>
<], SPE, 3>
<=, ASS, 0>
<queue, ID, 0>
<[, SPE, 2>
<i, ID, 0>
<], SPE, 3>
<->, SPE, 4>
<right, ID, 0>
<;, SYM, 5>
<}, SYM, 0>
<}, SYM, 0>
<ret, ID, 0>
```

```
<[, SPE, 2>
<*, ARI, 2>
<returnSize, ID, 0>
<], SPE, 3>
<=, ASS, 0>
<(, SPE, 0>
<struct, KEY, 26>
<ListNode, ID, 0>
<*, ARI, 2>
<), SPE, 1>
<malloc, ID, 0>
<(, SPE, 0>
<sizeof, KEY, 24>
<(, SPE, 0>
<struct, KEY, 26>
<ListNode, ID, 0>
<), SPE, 1>
<), SPE, 1>
<;, SYM, 5>
</*, NOT, 0>
<struct, KEY, 26>
<ListNode, ID, 0>
<*, ARI, 2>
<head, ID, 0>
<=, ASS, 0>
<ret, ID, 0>
<[, SPE, 2>
<*, ARI, 2>
<returnSize, ID, 0>
<], SPE, 3>
<;, SYM, 5>
<head, ID, 0>
<->, SPE, 4>
<val, ID, 0>
<=, ASS, 0>
<queue, ID, 0>
<[, SPE, 2>
<old, ID, 0>
<_start, ID, 0>
<], SPE, 3>
<->, SPE, 4>
```

```
<val, ID, 0>
<;, SYM, 5>
<head, ID, 0>
<->, SPE, 4>
<next, ID, 0>
<=, ASS, 0>
<NULL, ID, 0>
<;, SYM, 5>
<for, KEY, 13>
<(, SPE, 0>
<int, KEY, 17>
<i, ID, 0>
<=, ASS, 0>
<old, ID, 0>
<_start, ID, 0>
<+, ARI, 0>
< , ARI, 1>
<1, NUMI, 1>
<;, SYM, 5>
<i, ID, 0>
<<, REL, 0>
<old, ID, 0>
<_end, ID, 0>
<;, SYM, 5>
<i, ID, 0>
<++, ARI, 5>
<+, ARI, 1>
<), SPE, 1>
<{, SYM, 0>
<struct, KEY, 26>
<ListNode, ID, 0>
<*, ARI, 2>
<current, ID, 0>
<=, ASS, 0>
<(, SPE, 0>
<struct, KEY, 26>
<ListNode, ID, 0>
<*, ARI, 2>
<), SPE, 1>
<malloc, ID, 0>
<(, SPE, 0>
<sizeof, KEY, 24>
<(, SPE, 0>
<struct, KEY, 26>
<ListNode, ID, 0>
<), SPE, 1>
<), SPE, 1>
<;, SYM, 5>
<current, ID, 0>
<->, SPE, 4>
<val, ID, 0>
<=, ASS, 0>
<queue, ID, 0>
<[, SPE, 2>
<i, ID, 0>
<], SPE, 3>
<->, SPE, 4>
<val, ID, 0>
<;, SYM, 5>
<current, ID, 0>
<->, SPE, 4>
<next, ID, 0>
<=, ASS, 0>
<NULL, ID, 0>
<;, SYM, 5>
<head, ID, 0>
<->, SPE, 4>
<next, ID, 0>
<=, ASS, 0>
<current, ID, 0>
<;, SYM, 5>
<head, ID, 0>
<=, ASS, 0>
<head, ID, 0>
<->, SPE, 4>
<next, ID, 0>
<;, SYM, 5>
<}, SYM, 0>
<(, SPE, 0>
<*, ARI, 2>
<returnSize, ID, 0>
<), SPE, 1>
```

```
        <++, ARI, 5>
        <+, ARI, 1>
        <;, SYM, 5>
        <}, SYM, 0>
        <return, KEY, 21>
        <ret, ID, 0>
        <;, SYM, 5>
        <}, SYM, 0>


    该程序共有 79 行
    该程序的字符总数为 1514 / 2215
    各种记号的的个数为：
        KEY:    40
        ID:     133
        NUMI:   10
        NUMF:   0
        REL:    9
        ASS:    25
        BIT:  0
        LOG:    0
        ARI: 38
        SPE:    92
        NOT:    11
        SYM:    57
        SEL:    1
```

```c
#include <stdlib.h>
#include <stdio.h>
#include <stdbool.h>
#include <string.h>
#include <math.h>
#include <limits.h>

#define MMAX(a, b)        ((a) > (b)? (a) : (b))
#define MMIN(a, b)        ((a) < (b)? (a) : (b))

int coins[4] = {1, 5, 10, 25};

//【算法思路】dp。
int waysToChange(int n){
    int *dp = (int *)calloc(n + 1, sizeof(int));

    dp[0] = 1;
    for(int i = 0; i < 4; i++) {
        int coin = coins[i];

        for(int j = 1; j <= n; j++) {
            if(j - coin >= 0) {
                dp[j] = (dp[j] + dp[j - coin]) % 1000000007;
            }
        }
    }

    return dp[n];
}
```

🤗 词法分析结果为：

```
<#, SYM, 7>
<include, ID, 0>
<<, REL, 0>
<stdlib, ID, 0>
<., SPE, 5>
<h, ID, 0>
<>, REL, 2>
<#, SYM, 7>
<include, ID, 0>
<<, REL, 0>
<stdio, ID, 0>
<., SPE, 5>
<h, ID, 0>
<>, REL, 2>
<#, SYM, 7>
<include, ID, 0>
<<, REL, 0>
<stdbool, ID, 0>
<., SPE, 5>
<h, ID, 0>
<>, REL, 2>
<#, SYM, 7>
<include, ID, 0>
<<, REL, 0>
<string, ID, 0>
<., SPE, 5>
<h, ID, 0>
<>, REL, 2>
<#, SYM, 7>
<include, ID, 0>
<<, REL, 0>
<math, ID, 0>
<., SPE, 5>
<h, ID, 0>
<>, REL, 2>
<#, SYM, 7>
<include, ID, 0>
<<, REL, 0>
<limits, ID, 0>
```

```
<., SPE, 5>
<h, ID, 0>
<>, REL, 2>
<#, SYM, 7>
<define, ID, 0>
<MMAX, ID, 0>
<(, SPE, 0>
<a, ID, 0>
<,, SYM, 6>
<b, ID, 0>
<), SPE, 1>
<(, SPE, 0>
<(, SPE, 0>
<a, ID, 0>
<), SPE, 1>
<>, REL, 2>
<(, SPE, 0>
<b, ID, 0>
<), SPE, 1>
<?, SEL, 0>
<(, SPE, 0>
<a, ID, 0>
<), SPE, 1>
<:, SYM, 2>
<(, SPE, 0>
<b, ID, 0>
<), SPE, 1>
<), SPE, 1>
<#, SYM, 7>
<define, ID, 0>
<MMIN, ID, 0>
<(, SPE, 0>
<a, ID, 0>
<,, SYM, 6>
<b, ID, 0>
<), SPE, 1>
<(, SPE, 0>
<(, SPE, 0>
<a, ID, 0>
<), SPE, 1>
<<, REL, 0>
```

```
<(, SPE, 0>
<b, ID, 0>
<), SPE, 1>
<?, SEL, 0>
<(, SPE, 0>
<a, ID, 0>
<), SPE, 1>
<:, SYM, 2>
<(, SPE, 0>
<b, ID, 0>
<), SPE, 1>
<), SPE, 1>
<int, KEY, 17>
<coins, ID, 0>
<[, SPE, 2>
<4, NUMI, 4>
<], SPE, 3>
<=, ASS, 0>
<{, SYM, 0>
<1, NUMI, 1>
<,, SYM, 6>
<5, NUMI, 5>
<,, SYM, 6>
<10, NUMI, 10>
<,, SYM, 6>
<25, NUMI, 25>
<}, SYM, 0>
<;, SYM, 5>
<//, NOT, 1>
<int, KEY, 17>
<waysToChange, ID, 0>
<(, SPE, 0>
<int, KEY, 17>
<n, ID, 0>
<), SPE, 1>
<{, SYM, 0>
<int, KEY, 17>
<*, ARI, 2>
<dp, ID, 0>
<=, ASS, 0>
<(, SPE, 0>
<int, KEY, 17>
<*, ARI, 2>
<), SPE, 1>
<calloc, ID, 0>
<(, SPE, 0>
<n, ID, 0>
<+, ARI, 0>
<1, NUMI, 1>
<,, SYM, 6>
<sizeof, KEY, 24>
<(, SPE, 0>
<int, KEY, 17>
<), SPE, 1>
<), SPE, 1>
<;, SYM, 5>
<dp, ID, 0>
<[, SPE, 2>
<0, NUMI, 0>
<], SPE, 3>
<=, ASS, 0>
<1, NUMI, 1>
<;, SYM, 5>
<for, KEY, 13>
<(, SPE, 0>
<int, KEY, 17>
<i, ID, 0>
<=, ASS, 0>
<0, NUMI, 0>
<;, SYM, 5>
<i, ID, 0>
<<, REL, 0>
<4, NUMI, 4>
<;, SYM, 5>
<i, ID, 0>
<++, ARI, 5>
<), SPE, 1>
<{, SYM, 0>
<int, KEY, 17>
<coin, ID, 0>
<=, ASS, 0>
<coins, ID, 0>
```

```
<[, SPE, 2>
<i, ID, 0>
<], SPE, 3>
<;, SYM, 5>
<for, KEY, 13>
<(, SPE, 0>
<int, KEY, 17>
<j, ID, 0>
<=, ASS, 0>
<1, NUMI, 1>
<;, SYM, 5>
<j, ID, 0>
<<=, REL, 1>
<n, ID, 0>
<;, SYM, 5>
<j, ID, 0>
<++, ARI, 5>
<), SPE, 1>
<{, SYM, 0>
<if, KEY, 15>
<(, SPE, 0>
<j, ID, 0>
<-, ARI, 1>
<coin, ID, 0>
<>=, REL, 3>
<0, NUMI, 0>
<), SPE, 1>
<{, SYM, 0>
<dp, ID, 0>
<[, SPE, 2>
<j, ID, 0>
<], SPE, 3>
<=, ASS, 0>
<(, SPE, 0>
<dp, ID, 0>
<[, SPE, 2>
<j, ID, 0>
<], SPE, 3>
<+, ARI, 0>
<dp, ID, 0>
<[, SPE, 2>
```

```
        <j, ID, 0>
        <-, ARI, 1>
        <coin, ID, 0>
        <], SPE, 3>
        <), SPE, 1>
        <%, ARI, 4>
        <1000000007, NUMI, 1000000007>
        <;, SYM, 5>
        <}, SYM, 0>
        <}, SYM, 0>
        <}, SYM, 0>
        <return, KEY, 21>
        <dp, ID, 0>
        <[, SPE, 2>
        <n, ID, 0>
        <], SPE, 3>
        <;, SYM, 5>
        <}, SYM, 0>



    该程序共有 30 行
    该程序的字符总数为 412 / 611
    各种记号的的个数为：
        KEY:    14
        ID:     62
        NUMI:   13
        NUMF:   0
        REL:    17
        ASS:    7
        BIT:  0
        LOG:    0
        ARI: 9
        SPE:    60
        NOT:    1
        SYM:    36
        SEL:    2
```

```
int g_allNumbers[26] = {0};
int g_total = 0;
int g_score[16] = {0};
int g_wordNumbers[16][26] = {0};

bool check(int i)
{
    for (int j = 0; j < 26; j++) {
        if (g_wordNumbers[i][j] > g_allNumbers[j]) {
            return false;
        }
    }
    return true;
}
void minus(int i)
{
    for (int j = 0; j < 26; j++) {
        g_allNumbers[j] -= g_wordNumbers[i][j];
    }
}
void add(int i)
{
    for (int j = 0; j < 26; j++) {
        g_allNumbers[j] += g_wordNumbers[i][j];
    }
}

void dfs(int *visited, int* score, char ** words, int wordsSize, int total, int curPos)
{
    if (total > g_total) {
        g_total = total;
    }
    int i = curPos;
    while (i < wordsSize) {
        if (check(i) && visited[i] == 0) {
            minus(i);
            visited[i] = 1;
            // printf("i->%d total->%d cur->%d\n", i, total, g_score[i]);
            dfs(visited, score, words, wordsSize, total + g_score[i], i + 1);
            visited[i] = 0;
            add(i);
        }
        i++;
    }
}

int maxScoreWords(char ** words, int wordsSize, char* letters, int lettersSize, int* score, int scoreSize)
{
    int visited[15] = {0};
    g_total = 0;
    memset(g_score, 0, sizeof(g_score));
    memset(g_allNumbers, 0, sizeof(g_allNumbers));
    memset(g_wordNumbers, 0, sizeof(g_wordNumbers));

    for (int i = 0; i < lettersSize; i++) {
        g_allNumbers[letters[i] - 'a']++;
    }
    for (int i = 0; i < wordsSize; i++) {
        for (int j = 0; j < strlen(words[i]); j++) {
            g_score[i] += score[words[i][j] - 'a'];
        }
    }
    for (int i = 0; i < wordsSize; i++) {
        for (int j = 0; j < strlen(words[i]); j++) {
            g_wordNumbers[i][words[i][j] - 'a']++;
        }
    }
    dfs(visited, score, words, wordsSize, 0, 0);

    return g_total;
}
```

🤗 词法分析结果为：

```
<int, KEY, 17>
<g, ID, 0>
<_allNumbers, ID, 0>
<[, SPE, 2>
<26, NUMI, 26>
<], SPE, 3>
<=, ASS, 0>
<{, SYM, 0>
<0, NUMI, 0>
<}, SYM, 0>
<;, SYM, 5>
<int, KEY, 17>
<g, ID, 0>
<_total, ID, 0>
<=, ASS, 0>
<0, NUMI, 0>
<;, SYM, 5>
<int, KEY, 17>
<g, ID, 0>
<_score, ID, 0>
<[, SPE, 2>
<16, NUMI, 16>
<], SPE, 3>
<=, ASS, 0>
<{, SYM, 0>
<0, NUMI, 0>
<}, SYM, 0>
<;, SYM, 5>
<int, KEY, 17>
<g, ID, 0>
<_wordNumbers, ID, 0>
<[, SPE, 2>
<16, NUMI, 16>
<], SPE, 3>
<[, SPE, 2>
<26, NUMI, 26>
<], SPE, 3>
<=, ASS, 0>
<{, SYM, 0>
```

```
<0, NUMI, 0>
<}, SYM, 0>
<;, SYM, 5>
<bool, ID, 0>
<check, ID, 0>
<(, SPE, 0>
<int, KEY, 17>
<i, ID, 0>
<), SPE, 1>
<{, SYM, 0>
<for, KEY, 13>
<(, SPE, 0>
<int, KEY, 17>
<j, ID, 0>
<=, ASS, 0>
<0, NUMI, 0>
<;, SYM, 5>
<j, ID, 0>
<<, REL, 0>
<26, NUMI, 26>
<;, SYM, 5>
<j, ID, 0>
<++, ARI, 5>
<), SPE, 1>
<{, SYM, 0>
<if, KEY, 15>
<(, SPE, 0>
<g, ID, 0>
<_wordNumbers, ID, 0>
<[, SPE, 2>
<i, ID, 0>
<], SPE, 3>
<[, SPE, 2>
<j, ID, 0>
<], SPE, 3>
<>, REL, 2>
<g, ID, 0>
<_allNumbers, ID, 0>
<[, SPE, 2>
<j, ID, 0>
<], SPE, 3>
<), SPE, 1>
<{, SYM, 0>
<return, KEY, 21>
<false, ID, 0>
<;, SYM, 5>
<}, SYM, 0>
<}, SYM, 0>
<return, KEY, 21>
<true, ID, 0>
<;, SYM, 5>
<}, SYM, 0>
<void, KEY, 31>
<minus, ID, 0>
<(, SPE, 0>
<int, KEY, 17>
<i, ID, 0>
<), SPE, 1>
<{, SYM, 0>
<for, KEY, 13>
<(, SPE, 0>
<int, KEY, 17>
<j, ID, 0>
<=, ASS, 0>
<0, NUMI, 0>
<;, SYM, 5>
<j, ID, 0>
<<, REL, 0>
<26, NUMI, 26>
<;, SYM, 5>
<j, ID, 0>
<++, ARI, 5>
<), SPE, 1>
<{, SYM, 0>
<g, ID, 0>
<_allNumbers, ID, 0>
<[, SPE, 2>
<j, ID, 0>
<], SPE, 3>
<-=, ASS, 2>
<g, ID, 0>
<_wordNumbers, ID, 0>
```

```
<[, SPE, 2>
<i, ID, 0>
<], SPE, 3>
<[, SPE, 2>
<j, ID, 0>
<], SPE, 3>
<;, SYM, 5>
<}, SYM, 0>
<}, SYM, 0>
<void, KEY, 31>
<add, ID, 0>
<(, SPE, 0>
<int, KEY, 17>
<i, ID, 0>
<), SPE, 1>
<{, SYM, 0>
<for, KEY, 13>
<(, SPE, 0>
<int, KEY, 17>
<j, ID, 0>
<=, ASS, 0>
<0, NUMI, 0>
<;, SYM, 5>
<j, ID, 0>
<<, REL, 0>
<26, NUMI, 26>
<;, SYM, 5>
<j, ID, 0>
<++, ARI, 5>
<), SPE, 1>
<{, SYM, 0>
<g, ID, 0>
<_allNumbers, ID, 0>
<[, SPE, 2>
<j, ID, 0>
<], SPE, 3>
<+=, ASS, 1>
<g, ID, 0>
<_wordNumbers, ID, 0>
<[, SPE, 2>
<i, ID, 0>
<], SPE, 3>
<[, SPE, 2>
<j, ID, 0>
<], SPE, 3>
<;, SYM, 5>
<}, SYM, 0>
<}, SYM, 0>
<void, KEY, 31>
<dfs, ID, 0>
<(, SPE, 0>
<int, KEY, 17>
<*, ARI, 2>
<visited, ID, 0>
<,, SYM, 6>
<int, KEY, 17>
<*, ARI, 2>
<score, ID, 0>
<,, SYM, 6>
<char, KEY, 3>
<*, ARI, 2>
<*, ARI, 2>
<words, ID, 0>
<,, SYM, 6>
<int, KEY, 17>
<wordsSize, ID, 0>
<,, SYM, 6>
<int, KEY, 17>
<total, ID, 0>
<,, SYM, 6>
<int, KEY, 17>
<curPos, ID, 0>
<), SPE, 1>
<{, SYM, 0>
<if, KEY, 15>
<(, SPE, 0>
<total, ID, 0>
<>, REL, 2>
<g, ID, 0>
<_total, ID, 0>
<), SPE, 1>
<{, SYM, 0>
```

```
<g, ID, 0>
<_total, ID, 0>
<=, ASS, 0>
<total, ID, 0>
<;, SYM, 5>
<}, SYM, 0>
<int, KEY, 17>
<i, ID, 0>
<=, ASS, 0>
<curPos, ID, 0>
<;, SYM, 5>
<while, ID, 0>
<(, SPE, 0>
<i, ID, 0>
<<, REL, 0>
<wordsSize, ID, 0>
<), SPE, 1>
<{, SYM, 0>
<if, KEY, 15>
<(, SPE, 0>
<check, ID, 0>
<(, SPE, 0>
<i, ID, 0>
<), SPE, 1>
<&&, LOG, 0>
<visited, ID, 0>
<[, SPE, 2>
<i, ID, 0>
<], SPE, 3>
<==, REL, 4>
<0, NUMI, 0>
<), SPE, 1>
<{, SYM, 0>
<minus, ID, 0>
<(, SPE, 0>
<i, ID, 0>
<), SPE, 1>
<;, SYM, 5>
<visited, ID, 0>
<[, SPE, 2>
<i, ID, 0>
<], SPE, 3>
<=, ASS, 0>
<1, NUMI, 1>
<;, SYM, 5>
<//, NOT, 1>
<dfs, ID, 0>
<(, SPE, 0>
<visited, ID, 0>
<,, SYM, 6>
<score, ID, 0>
<,, SYM, 6>
<words, ID, 0>
<,, SYM, 6>
<wordsSize, ID, 0>
<,, SYM, 6>
<total, ID, 0>
<+, ARI, 0>
<g, ID, 0>
<_score, ID, 0>
<[, SPE, 2>
<i, ID, 0>
<], SPE, 3>
<,, SYM, 6>
<i, ID, 0>
<+, ARI, 0>
<1, NUMI, 1>
<), SPE, 1>
<;, SYM, 5>
<visited, ID, 0>
<[, SPE, 2>
<i, ID, 0>
<], SPE, 3>
<=, ASS, 0>
<0, NUMI, 0>
<;, SYM, 5>
<add, ID, 0>
<(, SPE, 0>
<i, ID, 0>
<), SPE, 1>
<;, SYM, 5>
<}, SYM, 0>
```

```
<i, ID, 0>
<++, ARI, 5>
<;, SYM, 5>
<}, SYM, 0>
<}, SYM, 0>
<int, KEY, 17>
<maxScoreWords, ID, 0>
<(, SPE, 0>
<char, KEY, 3>
<*, ARI, 2>
<*, ARI, 2>
<words, ID, 0>
<,, SYM, 6>
<int, KEY, 17>
<wordsSize, ID, 0>
<,, SYM, 6>
<char, KEY, 3>
<*, ARI, 2>
<letters, ID, 0>
<,, SYM, 6>
<int, KEY, 17>
<lettersSize, ID, 0>
<,, SYM, 6>
<int, KEY, 17>
<*, ARI, 2>
<score, ID, 0>
<,, SYM, 6>
<int, KEY, 17>
<scoreSize, ID, 0>
<), SPE, 1>
<{, SYM, 0>
<int, KEY, 17>
<visited, ID, 0>
<[, SPE, 2>
<15, NUMI, 15>
<], SPE, 3>
<=, ASS, 0>
<{, SYM, 0>
<0, NUMI, 0>
<}, SYM, 0>
<;, SYM, 5>
```

```
<g, ID, 0>
<_total, ID, 0>
<=, ASS, 0>
<0, NUMI, 0>
<;, SYM, 5>
<memset, ID, 0>
<(, SPE, 0>
<g, ID, 0>
<_score, ID, 0>
<,, SYM, 6>
<0, NUMI, 0>
<,, SYM, 6>
<sizeof, KEY, 24>
<(, SPE, 0>
<g, ID, 0>
<_score, ID, 0>
<), SPE, 1>
<), SPE, 1>
<;, SYM, 5>
<memset, ID, 0>
<(, SPE, 0>
<g, ID, 0>
<_allNumbers, ID, 0>
<,, SYM, 6>
<0, NUMI, 0>
<,, SYM, 6>
<sizeof, KEY, 24>
<(, SPE, 0>
<g, ID, 0>
<_allNumbers, ID, 0>
<), SPE, 1>
<), SPE, 1>
<;, SYM, 5>
<memset, ID, 0>
<(, SPE, 0>
<g, ID, 0>
<_wordNumbers, ID, 0>
<,, SYM, 6>
<0, NUMI, 0>
<,, SYM, 6>
<sizeof, KEY, 24>
```

```
<(, SPE, 0>
<g, ID, 0>
<_wordNumbers, ID, 0>
<), SPE, 1>
<), SPE, 1>
<;, SYM, 5>
<for, KEY, 13>
<(, SPE, 0>
<int, KEY, 17>
<i, ID, 0>
<=, ASS, 0>
<0, NUMI, 0>
<;, SYM, 5>
<i, ID, 0>
<<, REL, 0>
<lettersSize, ID, 0>
<;, SYM, 5>
<i, ID, 0>
<++, ARI, 5>
<), SPE, 1>
<{, SYM, 0>
<g, ID, 0>
<_allNumbers, ID, 0>
<[, SPE, 2>
<letters, ID, 0>
<[, SPE, 2>
<i, ID, 0>
<], SPE, 3>
<-, ARI, 1>
<', SYM, 3>
<a, ID, 0>
<', SYM, 3>
<], SPE, 3>
<++, ARI, 5>
<;, SYM, 5>
<}, SYM, 0>
<for, KEY, 13>
<(, SPE, 0>
<int, KEY, 17>
<i, ID, 0>
<=, ASS, 0>
```

```
<0, NUMI, 0>
<;, SYM, 5>
<i, ID, 0>
<<, REL, 0>
<wordsSize, ID, 0>
<;, SYM, 5>
<i, ID, 0>
<++, ARI, 5>
<), SPE, 1>
<{, SYM, 0>
<for, KEY, 13>
<(, SPE, 0>
<int, KEY, 17>
<j, ID, 0>
<=, ASS, 0>
<0, NUMI, 0>
<;, SYM, 5>
<j, ID, 0>
<<, REL, 0>
<strlen, ID, 0>
<(, SPE, 0>
<words, ID, 0>
<[, SPE, 2>
<i, ID, 0>
<], SPE, 3>
<), SPE, 1>
<;, SYM, 5>
<j, ID, 0>
<++, ARI, 5>
<), SPE, 1>
<{, SYM, 0>
<g, ID, 0>
<_score, ID, 0>
<[, SPE, 2>
<i, ID, 0>
<], SPE, 3>
<+=, ASS, 1>
<score, ID, 0>
<[, SPE, 2>
<words, ID, 0>
<[, SPE, 2>
```

```
<i, ID, 0>
<], SPE, 3>
<[, SPE, 2>
<j, ID, 0>
<], SPE, 3>
<-, ARI, 1>
<', SYM, 3>
<a, ID, 0>
<', SYM, 3>
<], SPE, 3>
<;, SYM, 5>
<}, SYM, 0>
<}, SYM, 0>
<for, KEY, 13>
<(, SPE, 0>
<int, KEY, 17>
<i, ID, 0>
<=, ASS, 0>
<0, NUMI, 0>
<;, SYM, 5>
<i, ID, 0>
<<, REL, 0>
<wordsSize, ID, 0>
<;, SYM, 5>
<i, ID, 0>
<++, ARI, 5>
<), SPE, 1>
<{, SYM, 0>
<for, KEY, 13>
<(, SPE, 0>
<int, KEY, 17>
<j, ID, 0>
<=, ASS, 0>
<0, NUMI, 0>
<;, SYM, 5>
<j, ID, 0>
<<, REL, 0>
<strlen, ID, 0>
<(, SPE, 0>
<words, ID, 0>
<[, SPE, 2>
```

```
<i, ID, 0>
<], SPE, 3>
<), SPE, 1>
<;, SYM, 5>
<j, ID, 0>
<++, ARI, 5>
<), SPE, 1>
<{, SYM, 0>
<g, ID, 0>
<_wordNumbers, ID, 0>
<[, SPE, 2>
<i, ID, 0>
<], SPE, 3>
<[, SPE, 2>
<words, ID, 0>
<[, SPE, 2>
<i, ID, 0>
<], SPE, 3>
<[, SPE, 2>
<j, ID, 0>
<], SPE, 3>
<-, ARI, 1>
<', SYM, 3>
<a, ID, 0>
<', SYM, 3>
<], SPE, 3>
<++, ARI, 5>
<;, SYM, 5>
<}, SYM, 0>
<}, SYM, 0>
<dfs, ID, 0>
<(, SPE, 0>
<visited, ID, 0>
<,, SYM, 6>
<score, ID, 0>
<,, SYM, 6>
<words, ID, 0>
<,, SYM, 6>
<wordsSize, ID, 0>
<,, SYM, 6>
<0, NUMI, 0>
```

```
          <,, SYM, 6>
          <0, NUMI, 0>
          <), SPE, 1>
          <;, SYM, 5>
          <return, KEY, 21>
          <g, ID, 0>
          <_total, ID, 0>
          <;, SYM, 5>
          <}, SYM, 0>



      该程序共有 71 行
      该程序的字符总数为 1229 / 1801
      各种记号的的个数为：
          KEY:      50
          ID:       164
          NUMI:     31
          NUMF:     0
          REL:      12
          ASS:      21
          BIT:  0
          LOG:      1
          ARI: 24
          SPE:      120
          NOT:      1
          SYM:      116
          SEL:      0
```

经过测试，各种情况下该程序都表现良好