/\*\*

 \* Return an array of arrays of size \*returnSize.

 \* The sizes of the arrays are returned as \*returnColumnSizes array.

 \* Note: Both returned array and \*columnSizes array must be malloced, assume caller calls free().

 \*/

int compare(const void \*a, const void \*b){

    return (\*(int \*)a - \*(int \*)b);

}

int\*\* threeSum(int\* nums, int numsSize, int\* returnSize, int\*\* returnColumnSizes){

    qsort(nums, numsSize, sizeof(int), compare);

    \*returnSize = 0;

    int unit = 512;

    int \*\*ans = (int \*\*)malloc(sizeof(int \*) \* unit);

    \*returnColumnSizes = (int \*)malloc(sizeof(int \*) \* unit);

    for (int i = 0; i < numsSize - 2; i++){

        int j = i + 1, k = numsSize - 1;

        if (i > 0 && nums[i] == nums[i - 1])

            continue;

        while (j < k){

            if (nums[i] + nums[j] + nums[k] == 0){

                (\*returnSize)++;

                if ((\*returnSize) % unit == 0){

                ans = realloc(ans, sizeof(int \*) \* (\*returnSize + 512));

                \*returnColumnSizes = realloc(\*returnColumnSizes, sizeof(int \*) \* (\*returnSize + unit));

                }

                ans[\*returnSize - 1] = (int \*)malloc(sizeof(int) \* 3);

                ans[\*returnSize - 1][0] = nums[i];

                ans[\*returnSize - 1][1] = nums[j];

                ans[\*returnSize - 1][2] = nums[k];

                (\*returnColumnSizes)[\*returnSize - 1] = 3;

                while (j < k && nums[k] == nums[k - 1])

                    k--;

                while (j < k && nums[j] == nums[j + 1])

                    j++;

                j++;

                k--;

            }

            else if (nums[i] + nums[j] + nums[k] < 0)

                j++;

            else

                k--;

        }

    }

    return ans;

}