HW4

一. 选择题

1. C;
2. A;
3. C;
4. B;
5. C, B。

二. 算法设计题

1.

```c
//
// 以下是 set.h 文件的内容。
//

#ifndef SET_H
#define SET_H

#include <stdbool.h>

typedef struct Node
{
    int value;
    struct Node* next;
} SetNode;

typedef struct
{
    SetNode** node_array;
    int size;
} Set;

Set* create_set(int size);
bool is_contained(const Set set, int value);
void insert_node(Set* set, int value);
void free_set(Set* set);

bool is_unique(int m, int n, const int arr[m][n]);

#endif //SET_H
```

```c
//
// 以下是 set.c 文件的内容。
//

#include <stdio.h>
#include <stdlib.h>
#include "set.h"
```

```
8
9  static bool is_contained_in_node(const Set set, int index, int value)
10 {
11     if (index < 0 || index >= set.size) return false;
12
13     SetNode* current = set.node_array[index];
14     while (current != NULL)
15     {
16         if (current->value == value) return true;
17         current = current->next;
18     }
19
20     return false;
21 }
22
23 static void free_node(SetNode* node)
24 {
25     SetNode* current = node;
26     while (current != NULL)
27     {
28         SetNode* temp = current;
29         current = current->next;
30         free(temp);
31     }
32 }
33
34 Set* create_set(int size)
35 {
36     if (size <= 0) return NULL;
37
38     Set* set = (Set*) malloc(sizeof(Set));
39     if (!set) return NULL;
40
41     set->size = size;
42     set->node_array = (SetNode**) calloc(size, sizeof(SetNode*));
43     if (!set->node_array)
44     {
45         free(set);
46         return NULL;
47     }
48
49     return set;
50 }
51
52 bool is_contained(const Set set, int value)
53 {
54     if (set.size <= 0 || set.node_array == NULL) return false;
55
56     int index = value % set.size;
57     if (index < 0) index += set.size;
58     return is_contained_in_node(set, index, value);
59 }
60
61 void insert_node(Set* set, int value)
62 {
63     if (set == NULL || set->size <= 0 || set->node_array == NULL)
    return;
64
```

```
    int index = value % set->size;
    if (index < 0) index += set->size;
    if (is_contained_in_node(*set, index, value)) return;

    SetNode* new_node = (SetNode*) malloc(sizeof(SetNode));
    if (!new_node) exit(1);
    new_node->value = value;
    new_node->next = set->node_array[index];
    set->node_array[index] = new_node;
}

void free_set(Set* set)
{
    if (set == NULL) return;
    for (int i = 0; i < set->size; i++)
    {
        free_node(set->node_array[i]);
    }

    free(set->node_array);
    free(set);
}

bool is_unique(int m, int n, const int arr[m][n])
{
    Set* set = create_set(m * n);
    if (set == NULL) exit(1);

    for (int i = 0; i < m; i++)
    {
        for (int j = 0; j < n; j++)
        {
            if (is_contained(*set, arr[i][j]))
            {
                free_set(set);
                printf("no\n");
                return false;
            }

            insert_node(set, arr[i][j]);
        }
    }

    free_set(set);
    printf("yes\n");
    return true;
}
```

2.
   - 平均时间复杂度：$mn \times O(1) = O(mn)$;
   - 最坏时间复杂度：$mn \times O(m \times n) = O((m \times n)^2)$。