浙江大学2023-2024学年春夏学期《数据结构基础... 已结束

判断题 10 | A. 单选题 20 | 程序填空题 2 | 函数题 1

概览

题目列表

提交列表

排名

答题已结束，仅供题目浏览

graph exactly once. An Euler curcuit is an Euler tour that starts and ends at the same vertex.

Function `Eulerian` is to test if there exists an Euler tour or an Euler curcuit in a given connected `Graph`. The array `Graph->G` stores the adjacency matrix of the undirected graph, and `MGraph` is defined as the following:

```
typedef struct GNode *PtrToGNode;
struct GNode{
     int Nv;  /* number of vertices */
     int Ne;  /* number of edges */
     int G[MaxVertexNum][MaxVertexNum]; /* adjacency matrix */
};
typedef PtrToGNode MGraph;
```

Please fill in the blanks.

```
Type Eulerian( MGraph Graph )
{
    int count_odd, degree;
    Vertex i, j;
    Type ret;

    count_odd = 0;
    for (i=0; i<Graph->Nv; i++) {
        degree = 0;
        for (j=0; j<Graph->Nv; j++) {
            if(Graph->G[i][j]!=0) degree++      3 分 ;
        }
        if (degree%2 == 1) {
            count_odd++;
        }
        if (count_odd > 2) break;
    }
    if (count_odd == 0) {
        ret = EulerCurcuit;
    }
    else if (  count_odd==2      3 分 ) {
        ret = EulerTour;
    }
    else {
        ret = NotEulerian;
    }
```

浙江大学2023-2024学年春夏学期《数据结构基础... 已结束

概览

题目列表

提交列表

排名

评测结果 答案正确

得分 6 分

5-1-1 分数 6 Hashing and rehashing 作者 李松 单位 浙江大学

Let's consider creating a basic hashing program for a list of **nonnegative** numbers with rehashing. We'll use **linear probing** (f(i) = i) to handle collisions. Moreover, rehashing will occur when the table reaches half capacity (capacity > 0.5).

```
typedef struct {
    int *table;
    int size;
    int count;
} HashTable;
void init(HashTable *ht, int size) {
    ht->size = size;
    ht->table = (int*)malloc(sizeof(int) * size);
    ht->count = 0;
    for (int i = 0; i < size; i++) {
        ht->table[i] = -1;
    }
}
void rehash(HashTable *ht);
void insert(HashTable *ht, int key) {
    if ( ht->count/ht->size>0.5 3 分 ) {
        rehash(ht);
    }
    int index = hash_function(key, ht->size);
    while (ht->table[index] != -1) {
        index = index+1 3 分 ; }
    ht->table[index] = key;
    ht->count++;
}
```

评测结果 部分正确

得分 3 分