实验1 算法时间复杂度、空间复杂度

**1．实验目的**

(1）通过实验，让学生理解算法的基本概念，包括算法的定义、特性、分类等，培养对算法的基本认识。

(2）掌握复杂度分析方法：实验可能包括对算法时间复杂度和空间复杂度的分析，帮助学生掌握复杂度分析的方法和技巧。

(3）培养编程能力：通过实验，培养学生的编程能力，让他们能够将算法实现为具体的程序，并进行测试和分析。

**2．实验内容和步骤**

编程序并上机调试运行

（1）设计一个尽可能高校的算法求1+++......+ ,其中n≥1。

#include <iostream>

using namespace std;

double sum(int n);//声明函数

int main() {

int n = 10 ;

double b = sum(n);

cout << b;

return 0;

}

double sum(int n) {

double s = 1.0;

int f = 1;

for (int i = 2; i <= n; i++) {

f \*= i;

s += 1.0 / f;

}

return s;

}

2.有一个数组a包含n（n>1)个整数元素，设计一个尽可能高效的算法将后面k(0≤K≤n)个元素循环右移。例如a=（1，2，3，4，5）,k=3，结果a=（3，4，5，1，2）。

方法一：普通数组实现

#include <iostream>

using namespace std;

//void swapst(int array[], int s, int t);

//void crightk(int array[], int n, int k);

//void swap(int &a, int &b) {

// int temp = a;

// a = b;

// b = temp;

//}

//反转

void reverse(int arr[], int s, int t) {

int i = s, j = t;

while (i < j) {

swap(arr[i], arr[j]);

i++;

j--;

}

// for (int i = 0; i < 5; i++) {

// cout << arr[i];

// }

// cout << endl;

}

//循环右移

void rotate(int a[], int n, int k) {

k = k % n; // 如果k大于n，则取模得到实际要移动的步数

reverse(a, 0, n - k - 1);

reverse(a, n - k, n - 1);

reverse(a, 0, n - 1);

}

//打印数组

void printArray(int arr[], int n) {

for (int i = 0; i < n; i++) {

cout << arr[i] << " ";

}

cout << endl;

}

int main() {

int a[] = {1, 2, 3, 4, 5};

int length = sizeof(a) / sizeof(a[0]);//利用sizeof计算数组的长度

int k = 3;

rotate(a, length, k);

printArray(a, length);

return 0;

}

方法二：用vector容器实现

#include <iostream>

#include <vector>

//#include <algorithm>

using namespace std;

void reverse(vector<int> &arr, int star, int end);

void rotate(vector<int> &arr, int k);

int main() {

vector<int> a = {1, 2, 3, 4, 5};

int k = 3;

rotate(a, k);

//输出结果

for (int num : a) {

cout << num << " ";

}

cout << endl;

return 0;

}

void reverse(vector<int> &arr, int star, int end) {

while (star < end) {

swap(arr[star], arr[end]);

star++;

end--;

}

}

void rotate(vector<int> &arr, int k) {

int n = arr.size();

reverse(arr, 0, n - k - 1); //反转数组前n-k个元素

reverse(arr, n - k, n - 1); //反转数组剩余k个元素

reverse(arr, 0, n - 1); //反转整个数组

}