算法第三周作业

> 所有程序源代码附在该pdf结尾

算法分析题

2-7 多项式算法设计和效率分析

> 算法设计：
>
> 通过分治思想将一个d次的多项式转换为两个d/2次多项式相乘
> 用$T(n)$表示计算时间
> 则$T(n)$满足：
> $T(d) = O(1), d = 1$
> $T(d) = 2T(d/2) + O(d\,log\,d), d > 1$
>
> 效率分析：
>
> 解得$T(d) = O(dlog^2d)$

2-15 循环赛日程表

> 算法设计：
>
> 当选手个数是奇数时，补充一名虚拟选手，将问题转换为偶数的情形
> 下面进入递归部分：
> 用n表示某个子问题的人数，
> (1)
> 如果n等于2，将左上角的元素复制到右下角，将左下角的元素复制到右上角（递归终止）
> (2)
> 如果n/2是偶数，将问题分解为2*2的子块；
> 然后对左边的子块继续递归，
> 然后将左上角的子块复制到右下角，左下角的子块复制到右上角，
> (3)
> 如果n/2是奇数，添加一个虚拟选手，先将问题转换为偶数时的情形，
> 然后重新对这些虚拟选手赋值
>
> 结束所有递归时，如果选手个数是奇数，将虚拟选手设为轮空，即解决问题

算法设计题

2-6 排列的字典序问题

> 算法设计：
>
> 字典序问题类似于全排列问题，
> n个元素的全排列问题可以分解为n个(n-1)个元素的全排列问题
> 直到分解为只剩一个元素的时候返回此时的排列
> 由此可以得到递归的函数：
> $r(n) = n*(n-1)! + r(n-1)$
>
> (1)由排列计算字典序值
> 找到每一个元素对应的小于自己的元素个数，乘以当前位数的阶乘
> 将每一个积相加，即得到对应的字典序值
> (2)由字典序值计算序列
> 将字典序值除以(n-1)的阶乘作为每个位置元素对应的小于自己的元素个数
> 直到n-1等于1以后结束递归

2-9 双色汉诺塔问题

> 算法设计：
>
> 根据汉诺塔的规则，可以将问题分解为
> 如果n=1，将圆盘移动到终点柱
>
> (1)上方n-1个圆盘移动到暂存柱，
> (2)底部编号为n的圆盘移动到终点柱，
> (3)上方n-1个圆盘移动到终点柱
>
> 每次递归时移动到一起的圆盘都是相邻的，也就是说盘子的奇偶性不一样的，
> 不会出现奇偶不同的圆盘叠在一起的情况。

> C++所有源代码如下：

```
#define _CRT_SECURE_NO_WARNINGS
#include<iostream>
#include<vector>
#include<algorithm>
using namespace std;
class Solution {
private:
        //辅助函数
        void arrUpSideDown(vector<int>& arr, const int left, const int right) {
                for (int i = left; i <= (right - left) / 2 + left; i++)
                        swap(arr[i], arr[right - i + left]);
        }
        string numVectorToString(const vector<int>arr) {
                string res = "";
                char ch_buf[10];
                int s = arr.size();
                for (int i = 0; i < s; i++) {
                        res += _itoa(arr[i], ch_buf, 10);
                        res += ' ';
                }
                return res;
        }
        string numVectorToMatrix(const vector<vector<int>>arr) {
                int s1 = arr.size(),//number of rows
                        s2 = arr[0].size();//number of cols
                string res;
                char ch_buf[10];
                for (int i = 0; i < s1; i++) {
                        for (int j = 0; j < s2; j++) {
                                res += '\t';
                                res += _itoa(arr[i][j], ch_buf, 10);
                        }
                        res += '\n';
                }
                return res;
        }
        int factorial(const int n) {
                if (n < 1)
                        return 0;
                if (n == 1)
                        return 1;
                return factorial(n - 1) * n;
        }
        void arrRefill(vector<vector<int>>& arr, const int n) {
                vector<int>temp1(n - 1);
                vector<int>temp2(n - 1);
                for (int i = 0; i < n - 1; i++)
                        temp1[i] = n + i;
                for (int j = 1; j < n; j++)
                        temp2[j - 1] = arr[n - 1][j];
                for (int i = 0; i < n - 1; i++)
                        for (int j = 0; j < n; j++)
                                if (arr[i][j] >= n) {
                                        arr[i][j] = temp1[i];
                                arr[arr[i][j] - 1][j] = i + 1;
                        }
                        else
                                arr[i + (n - 1)][j] = arr[i][j] + (n - 1);
        for (int i = 0; i < n - 1; i++) {
                for (int j = n; j < 2 * (n - 1); j++) {
                        arr[i][j] = temp1[(1 + i + (j - n)) % (n - 1)];
                        arr[arr[i][j] - 1][j] = i + 1;
                }
        }
}
void arrRestruct(vector<vector<int>>& arr, const int n) {
        arr.resize(n);
        int s = arr.size();
        for (int i = 0; i < s; i++)
                arr[i].resize(n);
}
void arrReturnBack(vector<vector<int>>& arr, const int n) {
        arr.pop_back();
        int s1 = arr.size(),
                s2 = arr[0].size();
        for (int i = 0; i < s1; i++)
                for (int j = 0; j < s2; j++)
                        if (arr[i][j] == n)
                                arr[i][j] = 0;
}
void matrixInitial(vector<vector<int>>& arr) {//initialize first line using 1~n
        int s = arr.size();
        for (int i = 0; i < s; i++)
                arr[i][0] = i + 1;
}
void matrixCopyPaste(vector<vector<int>>& arr, const int start_x, const int start_y, cor
        for (int i = 0; i < n; i++)
                for (int j = 0; j < n; j++) {
                        arr[start_y + n + i][start_x + n + j] = arr[start_y + i][start_>
                        arr[start_y + i][start_x + n + j] = arr[start_y + n + i][start_>
                }
}
//循环赛日程表
void tableRecursion(vector<vector<int>>& arr, const int start_x, const int start_y, cons
        if (n < 2)
                return;
        if (n == 2) {
                arr[start_y][start_x + 1] = arr[start_y + 1][start_x];
                arr[start_y + 1][start_x + 1] = arr[start_y][start_x];
                return;
        }
        tableRecursion(arr, start_x, start_y, n / 2);
        tableRecursion(arr, start_x, start_y + n / 2, n / 2);
        matrixCopyPaste(arr, start_x, start_y, n / 2);
}
void tableRecursionGeneral(vector<vector<int>>& arr, const int start_x, const int start_
        if (n < 2)
                return;
        if (n == 2) {
```

```
				arr[start_y][start_x + 1] = arr[start_y + 1][start_x];
				arr[start_y + 1][start_x + 1] = arr[start_y][start_x];
				return;
			}
			if ((n / 2) % 2) {
				tableRecursionGeneral(arr, start_x, start_y, (n + 1) / 2);
				matrixCopyPaste(arr, start_x, start_y, (n + 1) / 2);
				//arrReturnBack(arr, n + 1);
				arrRefill(arr, n + 1);
				//start here...
				return;
			}
			tableRecursionGeneral(arr, start_x, start_y, n / 2);
			tableRecursionGeneral(arr, start_x, start_y + n / 2, n / 2);
			matrixCopyPaste(arr, start_x, start_y, n / 2);
		}
	public:
		//字典序
		int dictionaryIndex(vector<int>arr, const int n) {
			int ans = 0;

			for (int i = 0; i < n; i++) {
				int pos = 0;
				for (int j = i + 1; j < n; j++)
					if (arr[j] < arr[i])
						pos++;
				ans += pos * factorial(n - i - 1);
			}
			return ans;
		}
		string dictionaryNextOrder(vector<int>arr, const int n) {
			int s = arr.size();
			int i = s - 2;
			bool flag = false;
			for (; i >= 0; i--) {
				if (arr[i] < arr[i + 1]) {
					flag = true;
					break;
				}
			}
			if (flag) {
				int j = s - 1;
				for (; arr[j] < arr[i] && j >= 0; j--)
					;
				swap(arr[i], arr[j]);
				arrUpSideDown(arr, i + 1, s - 1);
				return numVectorToString(arr);
			}
			return "error_overflow";
		}
		//汉诺塔
		void hanoiCore(const int n, const char src, const char tmp, const char dst) {
			if (n == 1) {
				cout << n << ' ' << src << ' ' << dst << endl;
				return;
```

```
                }
                hanoiCore(n - 1, src, dst, tmp);
                cout << n << ' ' << src << ' ' << dst << endl;
                hanoiCore(n - 1, tmp, src, dst);
        }

        string tenisTimeTable(vector<vector<int>>& arr, const int n) {
                matrixInitial(arr);
                tableRecursion(arr, 0, 0, n);
                return numVectorToMatrix(arr);
        }
        string tenisTimeTableGeneral(vector<vector<int>>& arr, const int n) {
                if (n % 2) {
                        arrRestruct(arr, n + 1);
                        matrixInitial(arr);
                        tableRecursionGeneral(arr, 0, 0, n + 1);
                        arrReturnBack(arr, n + 1);
                        return numVectorToMatrix(arr);
                }
                matrixInitial(arr);
                tableRecursionGeneral(arr, 0, 0, n);
                return numVectorToMatrix(arr);
        }
};

class testData {
private:
        //检验是否为2的k次幂
        bool checkPowerOf2(int n) {
                if (n <= 1)
                        return false;
                int count = 0;
                while (n) {
                        count += (n % 2);
                        n /= 2;
                }
                if (count > 1)
                        return false;
                return true;
        }
public:
        //检验程序——字典序
        void dicOrder(vector<int>arr, const int n) {
                Solution test;
                cout << "The index of this is: " << endl << test.dictionaryIndex(arr, n) << endl
                cout << "The next order of this order is: " << endl << test.dictionaryNextOrder(
                cout << endl;
        }
        //检验程序——汉诺塔
        void hanoi(const int n) {
                Solution test;
                cout << "The steps to finish the task is: " << endl;
                test.hanoiCore(n, 'A', 'B', 'C');
                cout << endl;
        }
```

```
	//检验程序—循环赛日程表
	void schedule(vector<vector<int>>& arr, const int n) {
		Solution test;
		if (checkPowerOf2(n) == false) {
			cout << "Error, the test data is illegal." << endl;
			return;
		}
		cout << "The schedule of this tenis match is: " << endl << test.tenisTimeTable(a
		cout << endl;
	}
	void scheduleGeneral(vector<vector<int>>& arr, const int n) {
		Solution test;
		cout << "The schedule of this tenis match is: " << endl << test.tenisTimeTableGe
		cout << endl;
	}
};

int main() {
	testData test;
	//test of dictionary order
	if (1) {
		vector<int>arr = { 2,6,4,5,8,1,7,3 };//edit here
		test.dicOrder(arr, arr.size());
	}
	//test of hanoi tower
	if (1) {
		test.hanoi(3);//edit here
	}
	//test of schedule
	if (1) {
		int n = 8;
		vector<vector<int>>arr(n, vector<int>(n, 0));
		test.schedule(arr, n);
	}
	//test of schedule(General)
	if (1) {
		int n = 5;
		vector<vector<int>>arr(n, vector<int>(n, 0));
		test.scheduleGeneral(arr, n);
	}
	return 0;
}
```