程序设计实训 HW2

姓名：王瀚森　　学号：2024013325

作业一：24点游戏求解

算法分析

本算法解决24点游戏问题，采用穷举策略确保找到所有可能的解，主要特点如下：

1. **穷举策略**：枚举所有可能的数字排列、运算符组合和括号结构，确保不会遗漏任何可能的解
2. **数字排列处理**：
   - 使用`std::sort`对输入数字进行排序
   - 使用`std::next_permutation`生成4个数字的全排列（$4! = 24$种）
   - 确保考虑所有数字顺序组合的可能性
3. **运算符组合处理**：
   - 3个运算符位置各4种可能：+、-、×、÷
   - 共$4^3 = 64$种运算符组合
   - 使用三重嵌套循环枚举所有组合
4. **括号结构处理**：
   - 定义5种不同的运算顺序（括号组合），覆盖所有可能的优先级组合：

     (1) `((a ⊙ b) ⊙ c) ⊙ d` - 从左到右依次计算

     (2) `d ⊙ ((a ⊙ b) ⊙ c)` - 右侧先计算

     (3) `((c ⊙ (a ⊙ b)) ⊙ d` - 中间先计算

     (4) `d ⊙ (c ⊙ (a ⊙ b))` - 从右到左依次计算

     (5) `(a ⊙ b) ⊙ (d ⊙ c)` - 分组计算
   - 每种括号结构独立实现为一个method函数
5. **浮点数处理**：
   - 使用1e-6容差处理浮点精度问题：`fabs(val - 24.0) < 1e-6`
   - 严格检查除零错误：`if (fabs(b) < 1e-6) return false`
   - 支持中间运算存在小数的情况
6. **去重机制**：
   - 使用`std::set<std::string>`存储表达式

- 确保输出结果不包含完全相同的表达式

7. **表达式构建：**
   - 动态构建带括号的表达式字符串
   - 使用`std::to_string`将数字转换为字符串
   - 根据运算顺序添加括号

时间复杂度分析：

- 数字排列：$O(4!) = 24$ 种
- 运算符组合：$O(4^3) = 64$ 种
- 括号结构：$O(5)$ 种
- 总计算量：$24 \times 64 \times 5 = 7680$ 次表达式计算

文件结构

- `Game24Solver.h` - 类声明
- `Game24Solver.cpp` - 类实现
- `main.cpp` - 主程序

Visual Studio 编译运行步骤（之后的代码同理）

1. 创建新项目：文件→ 新建→ 项目→ 空项目
2. 添加源文件：右键”源文件” → 添加→ 现有项→ 选择 Game24Solver.cpp, Main.cpp
3. 添加头文件：右键”头文件” → 添加→ 现有项→ 选择 Game24Solver.h
4. 编译运行：按 Ctrl+F5

核心代码（请以源程序代码为准）

Game24Solver.h

```
#pragma once
#include <vector>
#include <set>
#include <string>

class Game24Solver {
    public:
    Game24Solver(const std::vector<int>& numbers);
    ~Game24Solver();
    void solve();
    void printSolutions() const;

    private:
    std::vector<double> nums;
    std::set<std::string> solutions;
    const char ops[4] = { '+', '-', '*', '/' };

    bool calculate(double a, double b, int op, double& result)
        ↪ const;
    bool method1(int op1, int op2, int op3, double& finalVal, std
        ↪ ::string& expr);
    bool method2(int op1, int op2, int op3, double& finalVal, std
        ↪ ::string& expr);
    bool method3(int op1, int op2, int op3, double& finalVal, std
        ↪ ::string& expr);
    bool method4(int op1, int op2, int op3, double& finalVal, std
        ↪ ::string& expr);
    bool method5(int op1, int op2, int op3, double& finalVal, std
        ↪ ::string& expr);
};
```

main.cpp

```cpp
#include "Game24Solver.h"
#include <iostream>
#include <vector>

int main() {
    std::vector<int> numbers;
    for (int i = 0; i < 4; i++) {
        int num;
        std::cin >> num;
        numbers.push_back(num);
    }

    Game24Solver solver(numbers);
    solver.solve();
    solver.printSolutions();

    return 0;
}
```

运行结果

测试用例1:

输入: 5 5 5 1

输出:

```
(5-(1/5))*5
5*(5-(1/5))
```

测试用例2:

输入: 3 3 8 8

输出:

```
8/(3-8/3)
```

测试用例3:

输入: 1 1 1 1

输出:

```
no
```

作业二：大数表达式求解

算法分析

本算法解决大数表达式求值问题，主要特点如下：

1. **大数表示**：使用逆序存储的整数向量（低位在前）
2. **运算符枚举**：枚举所有运算符组合（2^{N-1}种可能）
3. **表达式计算**：严格按顺序计算（无运算符优先级）
4. **结果处理**：
   - 优先寻找等于目标值D的表达式
   - 若无解，寻找大于D的最小值
   - 若无大于D的值，输出-1
5. **高效比较**：实现大数比较算法

时间复杂度：$O(2^{N-1} \times N \times L)$，L为数字长度

文件结构

- `BigInt.h` - 大数运算声明
- `BigInt.cpp` - 大数运算实现
- `Expression.h` - 表达式类声明
- `Expression.cpp` - 表达式类实现
- `Solver.h` - 求解器类声明
- `Solver.cpp` - 求解器类实现
- `main.cpp` - 主程序

核心代码

BigInt.h

```
#pragma once
#include <vector>
#include <string>

std::vector<int> add(const std::vector<int>& a, const std::vector
    ↪ <int>& b);
std::vector<int> multiply(const std::vector<int>& a, int b);
bool compare(const std::vector<int>& a, const std::vector<int>& b
    ↪ );
bool equals(const std::vector<int>& a, const std::vector<int>& b)
    ↪ ;
std::string to_string(const std::vector<int>& num);
std::vector<int> stringToBigInt(const std::string& str);
```

Expression.h

```
#pragma once
#include "BigInt.h"
#include <vector>
#include <string>

class Expression {
    private:
    std::vector<std::vector<int>> numbers;
    std::vector<char> operators;
    std::vector<int> result;
    std::string exprStr;

    public:
    Expression(const std::vector<std::vector<int>>& nums);
    void setOperators(const std::vector<char>& ops);
    std::vector<int> getResult() const;
    std::string getExpressionString() const;

    private:
    void calculateResult();
    void buildExpressionString();
};
```

Solver.h

```
#pragma once
#include "BigInt.h"
#include "Expression.h"
#include <vector>

class Solver {
    private:
    std::vector<std::vector<int>> numbers;
    int n;
    std::vector<int> target;

    public:
    Solver(const std::vector<std::vector<int>>& nums, const std::
        ↪ vector<int>& d);
    void solve();

    private:
    void handleSingleNumberCase();
    void processResults(const std::vector<Expression>&
        ↪ expressions);
};
```

main.cpp

```
#include <iostream>
#include <vector>
#include <string>
#include "Solver.h"
#include "BigInt.h"

int main() {
    int N;
    std::string D;
    std::cin >> N >> D;

    std::vector<std::vector<int>> numbers(N);
    for (int i = 0; i < N; i++) {
        std::string numStr;
        std::cin >> numStr;
        numbers[i] = stringToBigInt(numStr);
    }

    std::vector<int> target = stringToBigInt(D);
    Solver solver(numbers, target);
    solver.solve();

    return 0;
}
```

运行结果

测试用例1：

输入:

```
4 235
34 12 5 5
```

输出:

```
34+12*5+5
```

测试用例2：

输入:

```
2 100
99 99
```

输出:

```
No
198
```

总结

本次作业实现了两个经典算法问题：

1. **24点游戏求解器：**通过穷举法高效解决24点问题
2. **大数表达式求解器：**实现大数运算和表达式求解