# 栈队列并查集

Ben

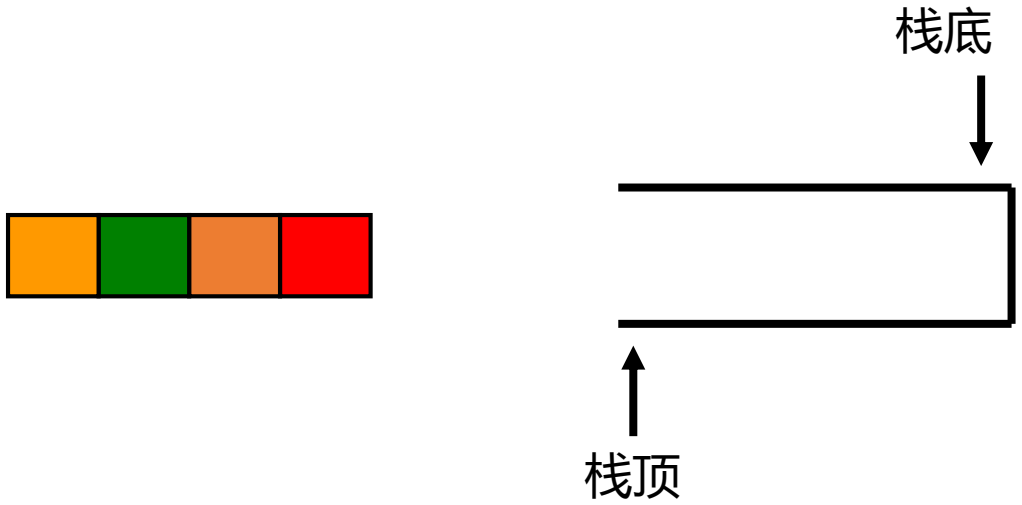
# Outline

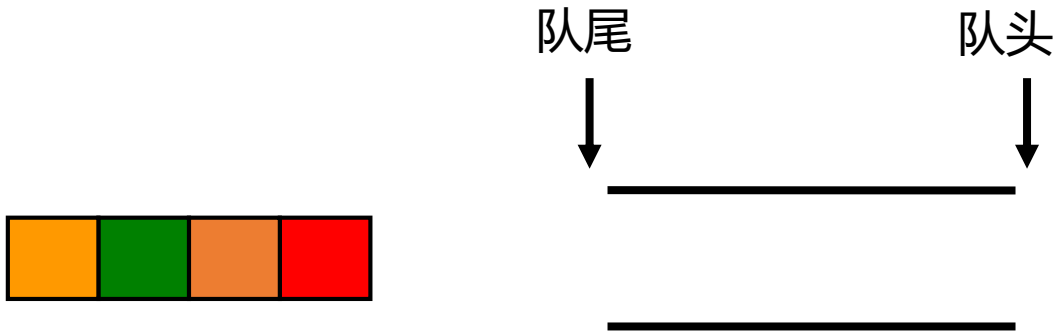
- 栈和队列
- 并查集

# 栈和队列：定义

- 存放数据的线性表
- 操作：入栈/队列、出栈/队列、判断满/空
- 空间复杂度：O(n)
- 单次操作时间复杂度：O(1)
- 区别
  - 先进后出（FILO, First In Last Out）
  - 先进先出（FIFO, First In First Out）

# 栈和队列：定义



# 栈和队列：实现

- 数组和链表皆可（线性表）
- 指针（辅助变量）
  - 栈顶/底指针
  - 队头/尾指针
- 关键：出入元素的同时移动指针

# 栈的应用：括号匹配检测

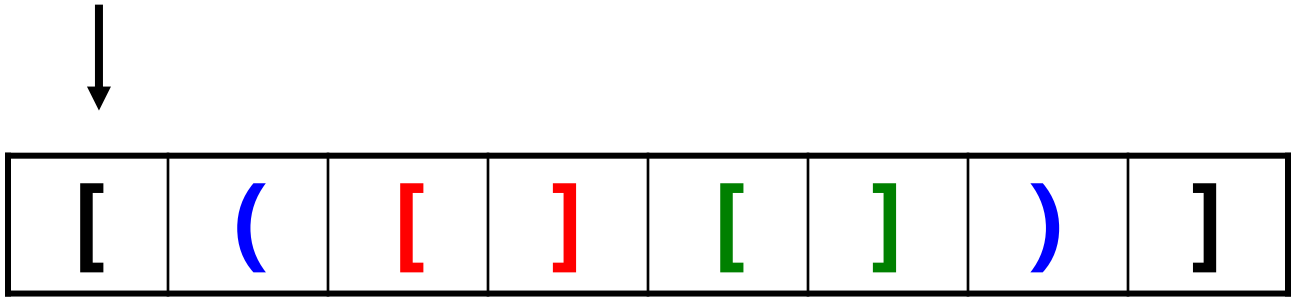
- 括号、引号等符号是成对出现的，必须相互匹配
- 设计一个算法，自动检测输入的字符串中的括号是否匹配
- 比如：
  - {}[([][])]　　匹配
  - [(])　　不匹配
  - (()]　　不匹配

[ ( [ ] [ ] ) ]

# 栈的应用：括号匹配检测

思考

- 从左向右扫描字符串

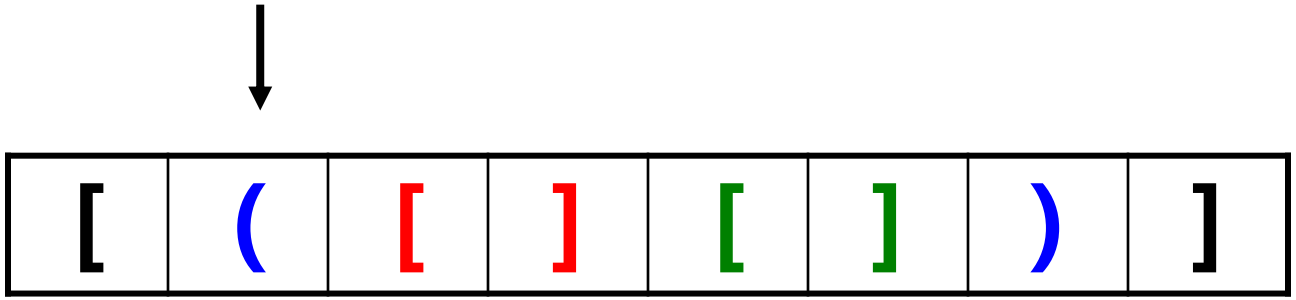



当前是[，期待一个]

# 栈的应用：括号匹配检测

思考

- 从左向右扫描字符串

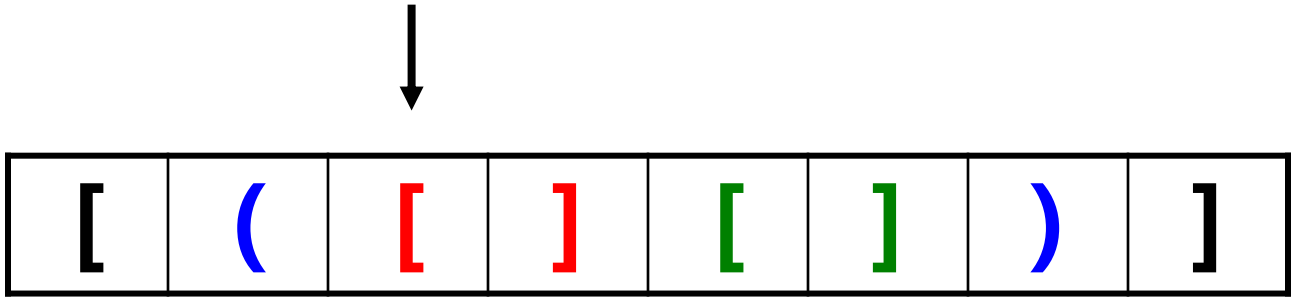



当前是(，和刚才的[不匹配，说明相匹配的符号还在右边，继续扫描

# 栈的应用：括号匹配检测

思考

- 从左向右扫描字符串

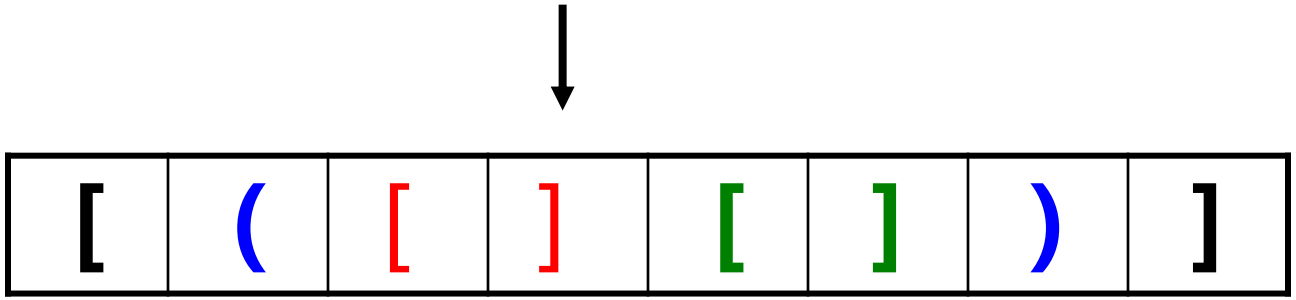



当前是[，和刚才的(不匹配，说明
相匹配的符号还在右边，继续扫描

# 栈的应用：括号匹配检测

思考

- 从左向右扫描字符串

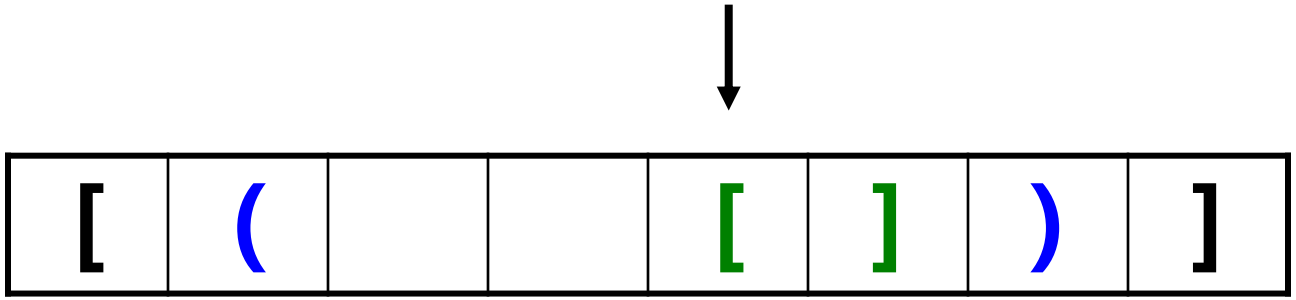



当前是]，和刚才的[正好一对，可以从字符串中“删去”不考虑了

# 栈的应用：括号匹配检测

思考

- 从左向右扫描字符串

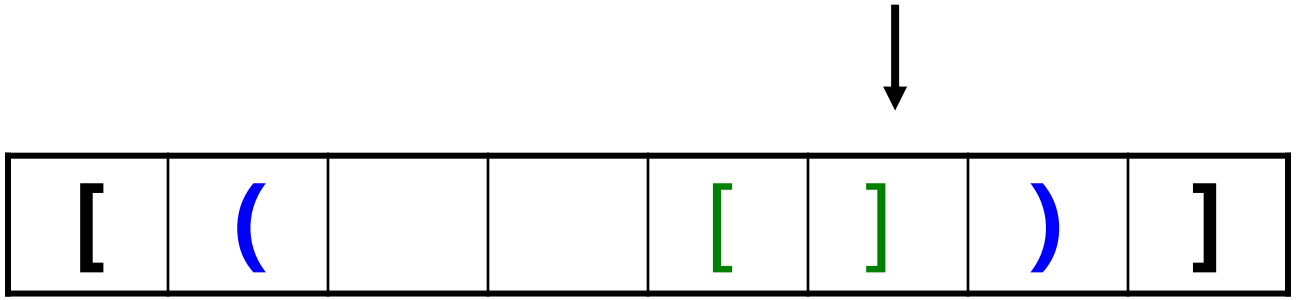



当前是[，目前最近的一个是(，不匹配，继续扫描

# 栈的应用：括号匹配检测

思考

- 从左向右扫描字符串

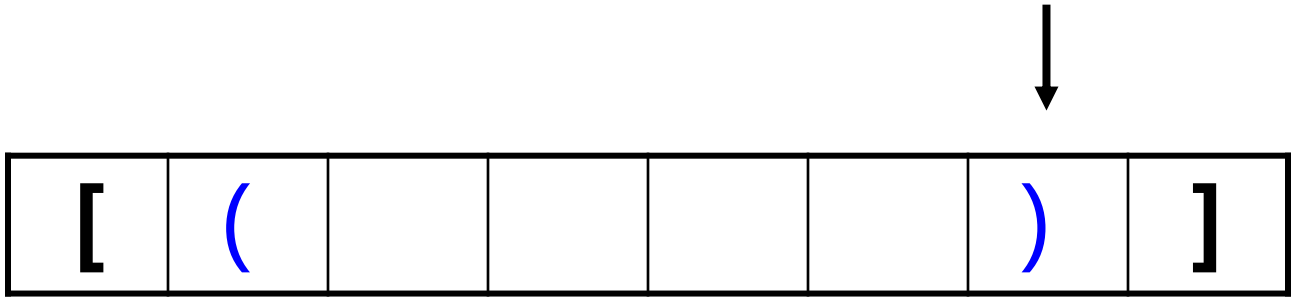



当前是]，和刚才的[正好一对，可以从字符串中“删去”不考虑了

# 栈的应用：括号匹配检测

思考

- 从左向右扫描字符串

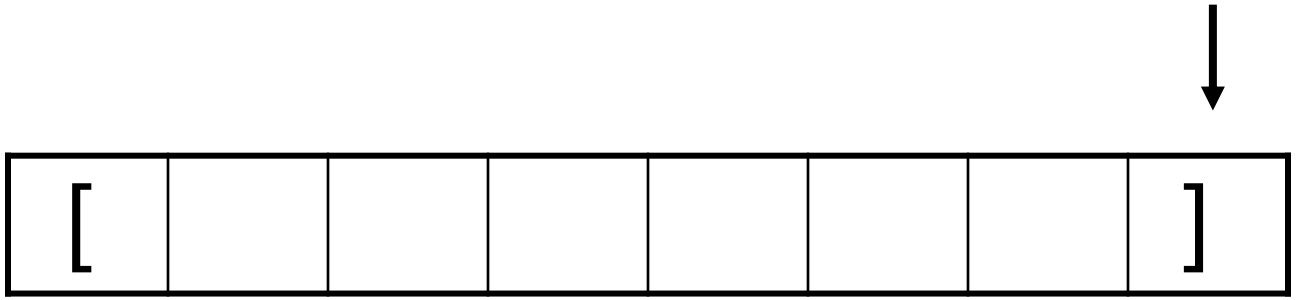



当前是)，目前最近的一个是(，正好一对，可以从字符串中“删去”不考虑了

# 栈的应用：括号匹配检测

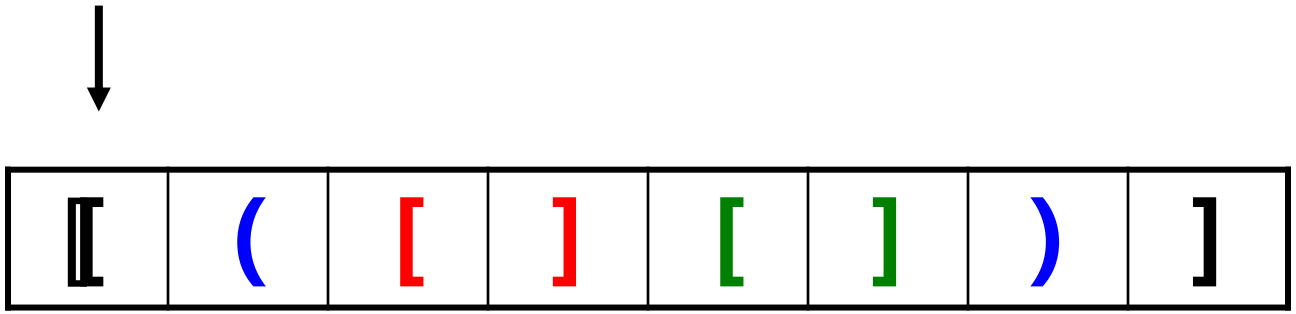
思考

- 从左向右扫描字符串



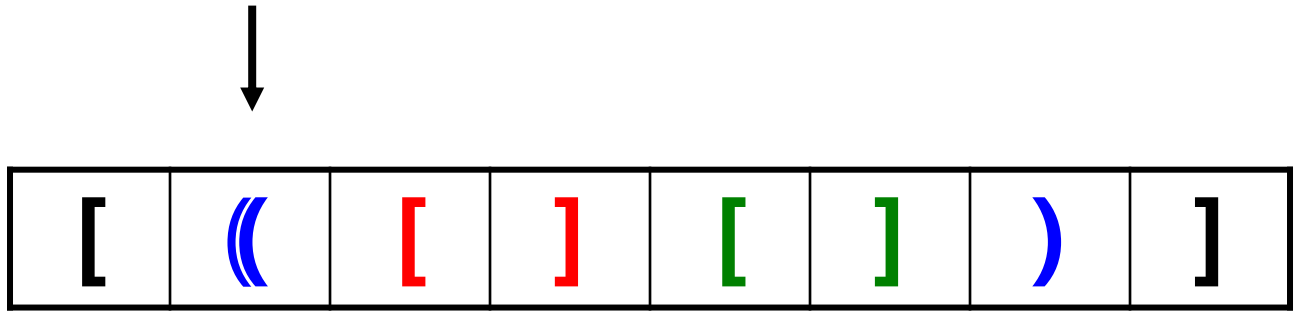

当前是]，目前最近的一个是[，正好一对，可以从字符串中“删去”不考虑了，此时左右的括号都匹配成功

# 栈的应用：括号匹配检测




栈为空，
当前字符直接入栈

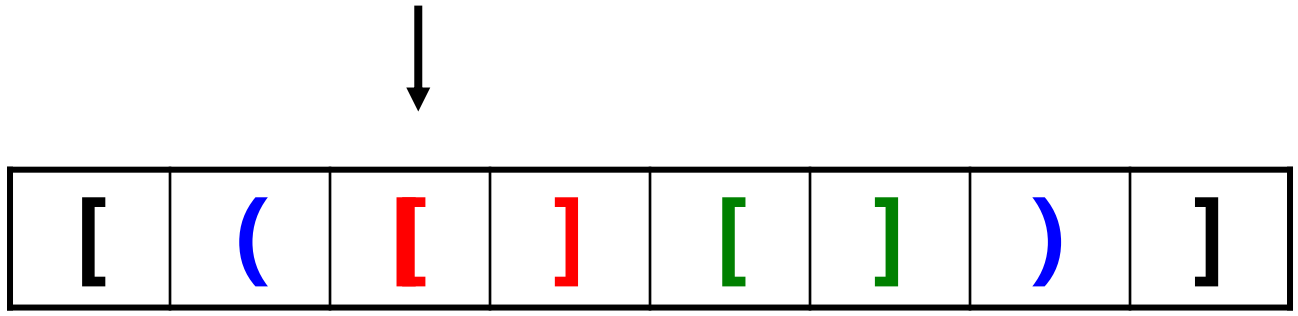
# 栈的应用：括号匹配检测

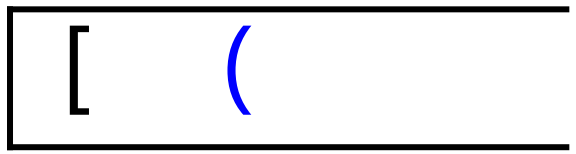




栈顶字符和当前字符不匹配，
当前字符入栈

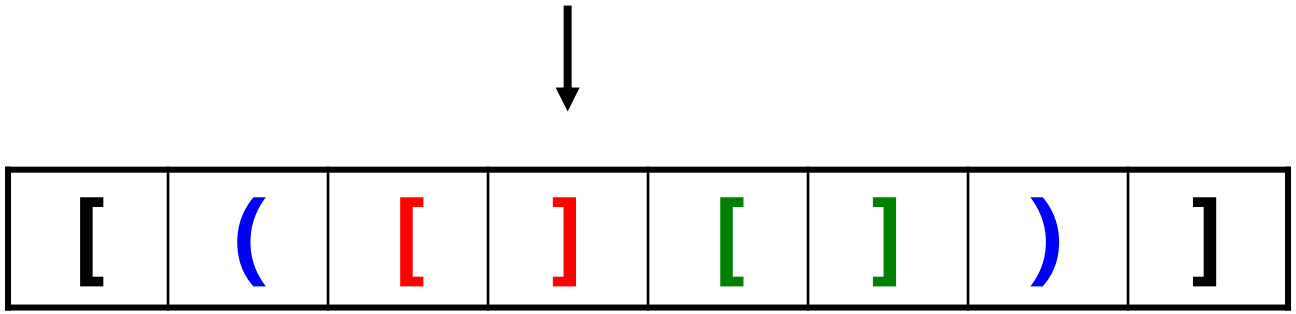
# 栈的应用：括号匹配检测

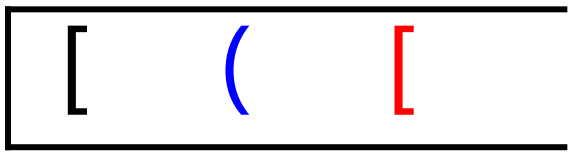



栈顶字符和当前字符不匹配，
当前字符入栈

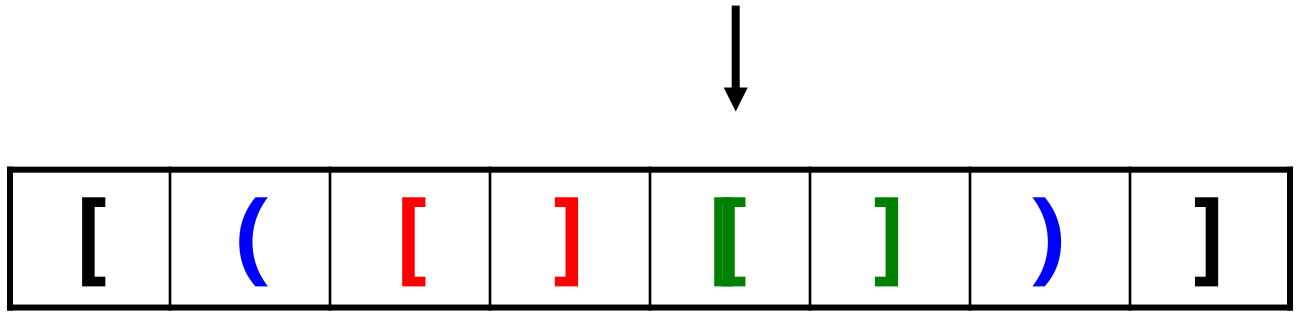
# 栈的应用：括号匹配检测

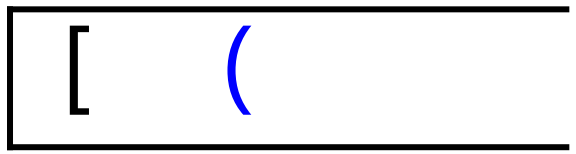



栈顶字符和当前字符匹配，
弹出栈顶字符

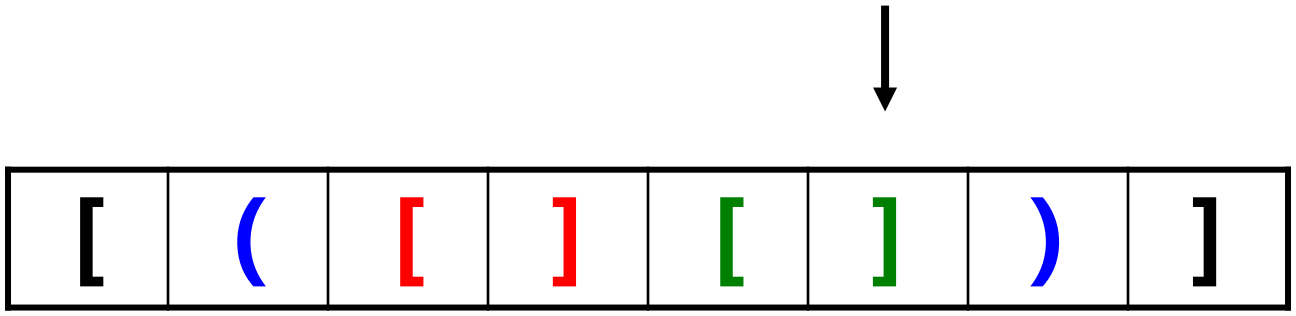
# 栈的应用：括号匹配检测

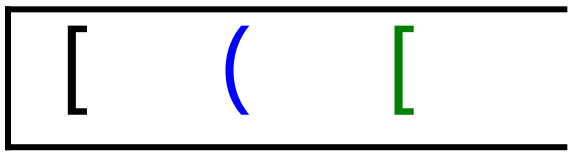



栈顶字符和当前字符不匹配，
当前字符入栈

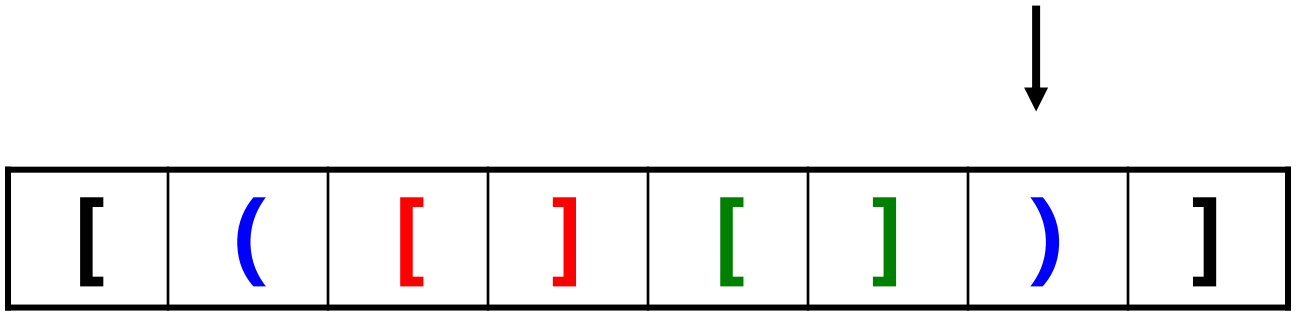
# 栈的应用：括号匹配检测

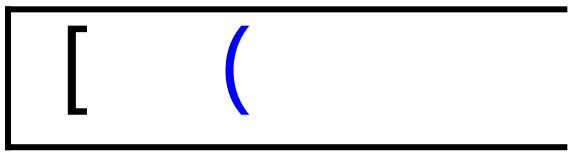



栈顶字符和当前字符匹配，
弹出栈顶字符

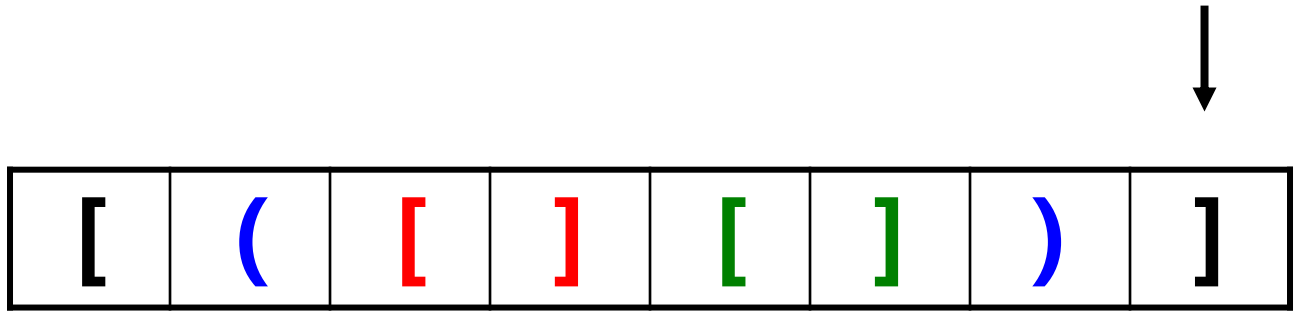
# 栈的应用：括号匹配检测




栈顶字符和当前字符匹配，
弹出栈顶字符

# 栈的应用：括号匹配检测





栈顶字符和当前字符匹配，
弹出栈顶字符

# 栈的应用：模拟系统栈

```
int F(int n) {
        if (n <= 1)
                return 1;
        return n * F(n – 1);
}
```

```
do {
        if (!back) {
                if (n <= 1) {
                        back = true, ret = 1;
                        continue;
                }
                n进栈;
                --n;
        } else { ret *= 出栈； }
} while (栈不为空);
```

# 栈和队列的应用：作业题

- 设计一个队列/栈
- 支持：出，入，求最大元素
- 要求所有操作O(1)
- 一个例子：
- 3 in, 4 in, 2 in, 5 in, out, out, 6 in, out, out, out

# 并查集：定义

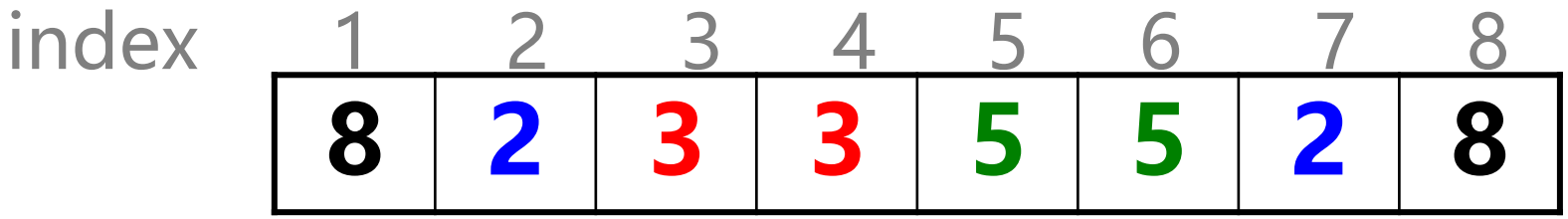
- 存放数据的集合关系，如{1，2}{3，4}{5}
- 支持操作
  - 建立新集合
  - 查找某个元素属于哪个集合
  - 合并两个集合
- 均摊时间复杂度近似O(1)

# 并查集的应用

- 假设n个节点，初始时点与点之间没有连接
- 给出一系列的连接操作
- 一次连接操作不产生环，则接受，否则被抛弃

# 并查集的实现

- 存储：使用数组标记每个元素属于的子集




- 合并：直接将根节点属于的子集改变
- 查找：从待查找节点倒推到根节点
- 优化：路径压缩

# 作业题1

- 编写快速排序（递归版）
- 将上述代码改写为非递归

# 作业题2

- 实现并查集的例题
- 进行正确性测试以及压力测试
- 路径压缩优化前后的效率对比

- MIT教授Erik Demaine：
- If you want to become a good programmer, you can spend 10 years programming, or spend 2 years programming and learning algorithms.
- Q&A