MSC-Share

> huanghongxun

CSP 部分题目解析

题目类型

- T1: 简单题（贪心、简单的公式推导）
- T2: 简单题
- T3: 字符串模拟
- T4: 图论题
- T5: 防 AK 题

CSP-201709-3

CSP-201809-3

CSP-201703-3

Templates

```
template <bool Cond, typename T = void> struct enable_if {};
template <typename T> struct enable_if<true, T>
{ typedef T type; };

template <typename T, typename U>
struct is_same : std::false_type {};
template <typename T>
struct is_same<T, T> : std::true_type {};
```

Templates

```
template <typename T>
typename enable_if<is_integral<T>::value, bool>::type
is_odd(T i) { return bool(i % 2); }
```

C++20 Concepts

```
template <typename T>
concept Integral = is_integral<T>::value;

template <Integral T>
bool is_odd(T i) { return bool(i % 2); }
```

Example

```
template<typename T>
concept Addable =
requires (T a, T b) {
    a + b;
};

template<typename T> concept C2 =
requires(T x) {
    {*x} -> typename T::inner;
    {x + 1} -> std::Same<int>;
    {x * 1} -> T;
};
```

```
template <class T>
concept Semiregular = DefaultConstructible<T> &&
    CopyConstructible<T> && Destructible<T> && CopyAssignable<T
requires(T a, size_t n) {
    requires Same<T*, decltype(&a)>;
    { a.~T() } noexcept;  // compound: "a.~T()" is a valid expr
    requires Same<T*, decltype(new T)>;
    requires Same<T*, decltype(new T[n])>;
    { delete new T };
    { delete new T[n] };
};
```