Tools 工具集

IO卡常

快读

仅整数读入。

```
namespace fast_io {
    constexpr int MAXBUF = 1<<20;
    char buf[MAXBUF],*pl,*pr;

    char gc() {
        if(pl==pr) pr=(pl=buf)+fread(buf,1,MAXBUF,stdin);
        return pl==pr?EOF:*pl++;
    }

    template<typename T> T rd(T &x) {
        x = 0;
        T f = 1;
        char c = gc();
        while (!isdigit(c)) {
            if (c == '-') f = -1;
            c = gc();
        }
        while (isdigit(c)) x = x * 10 + (c ^ 48), c = gc();
        return x = x * f;
    }

    template<typename... T> auto read(T&... x) { return (rd(x),...); }
```

```
    #undef gc
} using fast_io::read;
```

快写

仅整数和字符。

```
namespace fast_io {
    constexpr int MAXPBUF = 1 << 20;
    char pbuf[MAXPBUF], *pp = pbuf;

    void clear_buffer() {
        fwrite(pbuf, 1, pp-pbuf, stdout);
        pp = pbuf;
    }

    void push(const char &c) {
        if (pp - pbuf == MAXPBUF) clear_buffer();
        *pp++ = c;
    }

    template<typename T> void wt(T x) {
        if (x < 0) push('-'), x = -x;
        static int sta[40];
        int top = 0;
        do {
            sta[top++] = x % 10;
        } while (x/=10);
        while (top) push(sta[--top] + '0');
    }

    void wt(const char &c) { push(c); }
    void wt(const string &s) { for (auto &x:s) push(x); }
    void wt(const char *p) { while (*p != '\0') push(*p++); }

    template<typename... T> void write(const T&... x) { (wt(x),...); }

    struct IO {
        ~IO() { clear_buffer(); }
    } static io;
} using fast_io::write;
```

读写

```
namespace fast_io {
    constexpr int MAXBUF = 1 << 20, MAXLEN = 1 << 20;
    char buf[MAXBUF], *pl, *pr;
    char str[MAXLEN];
```

```
    #define gc() \
    (pl == pr && (pr = (pl = buf) + fread(buf, 1, MAXBUF, stdin), pl == pr)
\
    ? EOF : *pl++)

    template<typename T> T rd(T &x) {
        x = 0;
        T f = 1;
        char c = gc();
        while (!isdigit(c)) {
            if (c == '-') f = -1;
            c = gc();
        }
        while (isdigit(c)) x = x * 10 + (c ^ 48), c = gc();
        if (c != '.') return x = x * f;
        for (double t = 0.1; c = gc(), isdigit(c); t *= 0.1) x += (c - '0')
* t;
        return x = x * f;
    }

    char* rd(char *p = str) {
        char c = gc(), *h = p;
        while (!isgraph(c)) c = gc();
        while (isgraph(c)) *p++ = c, c = gc();
        *p = '\0';
        return h;
    }

    char rd(char &c) {
        c = gc();
        while (!isgraph(c)) c = gc();
        return c;
    }

    string rd(string &s) { return s = rd(str); }

    template<typename... T> auto read(T&... x) { return (rd(x),...); }
    #undef gc

    constexpr int MAXPBUF = 1 << 20, PRECISION = 7;
    char pbuf[MAXPBUF], *pp = pbuf;

    void clear_buffer() { fwrite(pbuf, 1, pp-pbuf, stdout), pp = pbuf; }

    void push(const char &c) {
        if (pp - pbuf == MAXPBUF) clear_buffer();
        *pp++ = c;
    }

    template<typename T> void wt(T x) {
        if (x < 0) push('-'), x = -x;
        static int sta[40];
        int top = 0;
        do {
```

```
            sta[top++] = x % 10;
        } while (x/=10);
        while (top) push(sta[--top] + '0');
    }

    template<typename T> void wt_f(T x,int p) {
        if (x < 0) push('-'), x = -x;
        long long pre = (long long)x;
        wt(pre);
        x -= pre;
        if (p) push('.');
        while (p--) {
            x *= 10;
            int t = (int)x;
            x -= t;
            push(t + '0');
        }
    }

    void wt(const char &c) { push(c); }
    void wt(const string &s) { for (auto &x:s) push(x); }
    void wt(const char *p) { while (*p != '\0') push(*p++); }

    void wt(const float &x, int p = PRECISION) { wt_f(x,p); }
    void wt(const double &x, int p = PRECISION) { wt_f(x,p); }
    void wt(const long double &x, int p = PRECISION) { wt_f(x,p); }

    template<typename... T> void write(const T&... x) { (wt(x),...); }
    template<typename T> void writef(const T &x,const int &p) { wt_f(x,p);
}

    struct IO {
        template<typename T> friend IO&
        operator>>(IO &io, T& x) { rd(x); return io; }

        template<typename T> friend IO&
        operator<<(IO &io, const T& x) { wt(x); return io; }

        ~IO() { clear_buffer(); }
    } static io;

} using fast_io::read,fast_io::write,fast_io::writef,fast_io::io;
```

编译运行

实用编译参数

- `-std=c++20`,`-std=gun++20` 指定C++标准
- `-O1`,`-O2`,`-O3`,`-Ofast` 指定优化等级
- `-DONLINE_JUDGE`,`-DTRIPLE_XOR_SUM` 预定义宏
- `-Wall`,`-Wextra` 指定警告等级

- `-fsanitize=address,-fsanitize=undefined` 启用内存检查（需要库支持）

手写 code runner (sh环境)

借助sh(bash)函数快速运行C++代码。

使用 `$0,$1,$2,...` 来访问函数参数。

```
function run {
    g++ -std=c++20 -Wall -Wextra -fsanitize=address -fsanitize=undefined \
    -O1 -o$1 $1.cpp && ./$1
}

function runo2 {
    g++ -std=c++20 -Wall -Wextra -O2 -o$1 $1.cpp && ./$1
}
```

使用函数。

```
run A
run A <1.in >1.out
runo2 A <1.in 1>1.out 2>1.err
```

手写 code runner (win32环境)

定义 powershell function

```
function run {
    g++ -std=c++20 -Wall -Wextra -fsanitize=address -fsanitize=undefined \
    -O1 "-o$Args" "$Args.cpp" && & "./$Args"
}

function runo2 {
    g++ -std=c++20 -Wall -Wextra -O2 "-o$Args" "$Args.cpp" && & "./$Args"
}
```

使用函数

```
run A
runo2 A
Get-Content 1.in | run A > 1.out
Get-Content 1.in | run A 1> 1.out 2> 1.err
```

powershell 计算校验和

```
Get-FileHash file -Algorithm SHA256 | Format-List
```

- `Algorithm`，指定算法，默认为 `SHA256`
- `Format-List`，可选，格式化输出。

计算程序运行时间

运行时间

```
chrono::system_clock clock;
auto t0=clock.now();
/* code */
chrono::duration<double,milli> dur=clock.now()-t0;
cout<<"Executed in "<<dur.count()<<" ms"<<endl;
```

使用 `clock()` 的一行式风格

```
cerr << "Time elapsed: " << 1.0 * clock() / CLOCKS_PER_SEC << " s." <<
endl;
// or
cerr << 1.0 * clock() << endl;
```

调试

debugger (200B version)

支持变长参数的极致精简版，仅能debug可输出类型。

```
template<class... T> void dbg(T... x) { ((cerr<<x<<' '),...),cerr<<endl; }
#define __pos__ string{}+"["+__func__+" "+to_string(__LINE__)+"] "
#define debug(...) cerr<<__pos__<<#__VA_ARGS__" = ",dbg(__VA_ARGS__)
// #define debug(...) 1 交之前别忘了define掉debug
```

debugger (1kiB version)

简易debuger加长版，如果有时间可以抄。

在编译命令中指定宏 `TRIPLE_XOR_SUM` 启用调试。

可以把代码写进文件然后include，减少打印代码的长度。

```

```

```
#ifdef TRIPLE_XOR_SUM
namespace debug {
    using S=string;
    using std::to_string;

    S to_string(const char &x) { return S(1,x); }
    S to_string(const string &x) { return "\'"+x+"\'"; }

    template<class T,class U> S to_string (const pair<T,U> &x) {
        return "("+to_string(x.first)+","+to_string(x.second)+")";
    }

    template<class T> S to_string(const T& x) {
        S res="{";
        for(auto &i:x) res+=to_string(i)+",";
        res.back()='}';
        return res;
    }

    template<class... T> S get(S name,const T&... x) {
        S res=name+" = ";
        ((res+=to_string(x),res+=" | "),...);
        res.pop_back(),res.pop_back();
        return res;
    }

    template<class T> S geta(S name,const T& arr,int l,int r) {
        S res=name+"["+to_string(l)+"->"+to_string(r)+"] = [";
        for(int i=l;i<=r;i++) res+=to_string(arr[i])+",";
        res.back()=']';
        return res;
    }
}
#define __pos__ string{}+"["+__func__+" "+to_string(__LINE__)+"] "
#define debug(...) cerr<<debug::get(__pos__+#__VA_ARGS__ __VA_OPT__(,)
__VA_ARGS__)<<endl;
#define debug_arr(x,l,r) cerr<<debug::geta(__pos__+#x,x,l,r)<<endl;
#else
#define debug(...) 1
#define debug_arr(...) 1
#endif
```

对拍

通常来说一份假的代码能很容易被找到问题，此时写完gen和std之后可以先手动运行几次，之后再考虑抄对拍代码。

简易数据生成器

```
mt19937 gen=mt19937(random_device{}());
// or
mt19937
gen=mt19937(chrono::system_clock().now().time_since_epoch().count());

int randint(int l,int r) {
    return gen()%(r-l+1)+l;
}
```

暴力对拍器 (sh 环境)

```
void hack() {
    auto fin=[&](string s) {
        cerr<<"[hack] "<<s<<endl;
        exit(0);
    };

    if(system("g++ gen.cpp -o gen -O2 -std=c++20")) fin("gen CE");
    if(system("g++ std.cpp -o std -O2 -std=c++20")) fin("std CE");
    if(system("g++ test.cpp -o test -O2 -std=c++20")) fin("test CE");

    for(int i=1;;i++) {
        cerr<<"[hack] #"<<i<<endl;
        if(system("./gen > 1.in")) fin("gen RE");
        if(system("./std < 1.in > 1.ans")) fin("std RE");
        if(system("./test < 1.in > 1.out")) fin("test RE");

        fstream x("1.out"),y("1.ans");
        while(x&&y) {
            string s,t;
            x>>s,y>>t;
            if(s!=t) fin("success!");
        }
    }
}
```

暴力对拍器 (win32 环境)

```
void hack() {
    auto fin=[&](string s) {
        cerr<<"[hack] "<<s<<endl;
        exit(0);
    };

    if(system("g++ gen.cpp -o gen -O2 -std=c++20")) fin("gen CE");
    if(system("g++ std.cpp -o std -O2 -std=c++20")) fin("std CE");
    if(system("g++ test.cpp -o test -O2 -std=c++20")) fin("test CE");
```

```
    for(int i=1;;i++) {
        cerr<<"[hack] #"<<i<<endl;
        if(system(".\\gen > 1.in")) fin("gen RE");
        if(system("type 1.in | .\\std > 1.ans")) fin("std RE");
        if(system("type 1.in | .\\test > 1.out")) fin("test RE");

        fstream x("1.out"),y("1.ans");
        while(x&&y) {
            string s,t;
            x>>s,y>>t;
            if(s!=t) fin("success!");
        }
    }
}
```