工具集

快读

仅整数读入。

```
namespace fast_io {
    constexpr int MAXBUF = 1e6;
    char buf[MAXBUF], *pl, *pr;

    #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();
        return x = x * f;
    }

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

    struct IO {
        template<typename T> friend IO&
        operator>>(IO &io, T& x) { rd(x); return io; }
    } static io;
}
using fast_io::read,fast_io::io;
```

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;
```

debug

内存检查编译选项

```
g++ A.cpp -o A -O2 -fsanitize=address -fsanitize=undefined
```

简易debuger

```
// #define ONLINE_JUDGE
#ifndef ONLINE_JUDGE
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<typename T,typename U> S to_string (const pair<T,U> &x) {
        return "("+to_string(x.first)+","+to_string(x.second)+")";
    }

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

    template<typename... 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<typename 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 range(x,l,r) cerr<<debug::geta(__pos__+#x,x,l,r)<<endl;
#else
#define debug(...) 1
#define range(...) 1
#endif
```

对拍

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 rnd(int l,int r) {
    int len=r-l+1;
    LL val=gen()%len;
    return val+l;
}
```

暴力对拍

sh,bash,etc.

```
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!");
        }
    }
}
```

powershell 替换这几行。

```
if(system("./gen > 1.in")) fin("gen RE");
if(system("Get-Content 1.in | ./std > 1.ans")) fin("std RE");
if(system("Get-Content 1.in | ./test > 1.out")) fin("test RE");
```

powershell 重定向标准输入输出

```
Get-Content 1.in | ./A > 1.out
```

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。

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;
```