EPITA ICPC Team Notebook (2017-18)

Contents

1 Miscellaneous

1.1 check

```
#!/bin/sh

clear
if g++ -g -O2 -Wall -Wextra -std=gnu++14 -static "$@" -lm code.cpp -o elf; then
    for f in *.in; do
        f=${f%.in}
        printf "Test $f: "
        ./elf check < $f.in > .tmp.out
        if [ -f $f.out ]; then
            if diff -q -b .tmp.out $f.out >/dev/null; then
                echo "PASS"
            else
                echo "FAIL"
                diff -y .tmp.out $f.out
            fi
            printf "\n"
        else
            echo "No $f.out file"
            cat .tmp.out
        fi
    done
fi
```

1.2 template.cpp

```
#include <iostream>
#include <vector>
#include <string>
#include <algorithm>
#include <tuple>

#include <cmath>
#include <cstdio>

#define FOR(i, n) for(lli i = 0; i < (lli) n; ++i)
#define ALL(x) (x).begin(), (x).end()

#define pb push_back
#define mt make_tuple
#define mp make_pair

#define fi first
#define se second

#define X(A) get<0>(A)
#define Y(A) get<1>(A)
#define Z(A) get<2>(A)
#define W(A) get<3>(A)

using namespace std;

using lli = long long int;

using vi = vector<lli>;
using vvi = vector<vi>;

using vb = vector<bool>;
using vvb = vector<vb>;

using ii = pair<lli, lli>;
using iii = pair<ii, lli>;

int main(int argc, char**)
{
    ios_base::sync_with_stdio(0);
    if(argc == 1) {freopen("PB_NAME.in","r",stdin);freopen("PB_NAME.out","w",stdout);}

    return 0;
}
```

1.3 gen

```
#!/bin/sh
PB_ID=$1
PB_NAME=$2
mkdir $PB_ID
cp template.cpp $PB_ID/code.cpp
if [ -z "$PB_NAME" ]; then
    sed -i "/PB_NAME/d" $PB_ID/code.cpp
    sed -i "/argc/c\int main(void)" $PB_ID/code.cpp
else
    sed -i "s/PB_NAME/$PB_NAME/g" $PB_ID/code.cpp
fi
cp check $PB_ID
cd $PB_ID
vim -u ../vimrc -p 1.in 1.out 2.in 2.out 3.in 3.out code.cpp
```

1.4 .vimrc

```
set nocompatible "helpful to test this vimrc

syntax on
filetype plugin on
set cc=80     "colorcolum
set bg=dark   "background

set nu        "number
set ts=4      "tabstop
set sw=4      "shiftwidth
set et        "expandtab
set so=5      "scrolloff
set sm        "showmatch

set ai        "autoindent
set si        "smartindent
set sr        "shiftround
set bs=2      "backspace=indent,eol,start

set hls       "hlsearch
set ic        "ignorecase

set nobk      "nobackup
set nowb      "nowritebackup
set noswf     "noswapfile

"fast save/quit
let mapleader="\<Space>"
nmap <leader>w :w<CR>
nmap <leader>q :q<CR>

"custom check script
nnoremap <CR> :w<CR>:!./check<CR>

map 0 ^
```

2 Math

2.1 Fast Fourier Transform

```
using cpx = complex<double>;
const double PI = acos(-1);

void fillPrimRoots(cpx* vec, lli n, bool conjugate)
{
    double s = conjugate ? -1 : 1;
    FOR(i, n / 2)
        vec[i] = polar(1., s * 2 * PI * i / n);
}

struct FFT
{
    lli n;
    vector<cpx> rt, rtc;
    vi rev;

    FFT(lli base) : n(1 << base), rt(n, 0), rtc(n), rev(n)
    {
        FOR(i, n)
            rev[i] = (rev[i >> 1] >> 1) + ((i & 1) << (base - 1));
        fillPrimRoots(rt.data() + n / 2, n, false);
        fillPrimRoots(rtc.data() + n / 2, n, true);
        FORD(i, 0, n / 2)
        {
            rt[i] = rt[2 * i];
            rtc[i] = rtc[2 * i];
        }
    }

    void fft(cpx* a, bool inv = false) const
    {
        const cpx* roots = inv ? rtc.data() : rt.data();
        FOR(i, n)
            if(i < rev[i])
                swap(a[i], a[rev[i]]);
        for(lli k = 1; k < n; k <<= 1)
            for(lli i = 0; i < n; i += 2 * k)
                FOR(j, k)
                {
                    cpx z = a[i + j + k] * roots[j + k];
                    a[i + j + k] = a[i + j] - z;
                    a[i + j] = a[i + j] + z;
                }
        if(inv)
        {
            cpx invn = cpx(1) / cpx(n);
            FOR(i, n)
                a[i] *= invn;
        }
    }
};

vi multFFT(const FFT& fft, const vi& a, const vi& b)
{
    lli n = fft.n;
    assert(a.size() == n && b.size() == n);
    vector<cpx> c(n);
    FOR(i, n)
        c[i] = cpx(a[i], b[i]);
    fft.fft(c.data());
    vector<cpx> f = c;
    FOR(i, n)
    {
        lli j = (n - i) & (n - 1);
        c[i] = (f[j] * f[j] - conj(f[i] * f[i])) * cpx(0, -.25 / n);
    }
    fft.fft(c.data());
    vi res(n);
    FOR(i, n)
        res[i] = (lli) round(c[i].real());
    return res;
}
```