E5-A题题解

> 作者：余绍函

题目描述

小 A 想给小 B 寄几盒蒜。但是小 B 不喜欢吃蒜。于是小 B 给小 A 一道简单题！小 A 解决题目之后，小 B 才愿意收下小 A 的几盒蒜。题目是这样的：

给定2^n-1次整系数多项式函数$f(x)=\sum_{k=0}^{2^n-1}c_kx^k$，记复数$\omega=cos\frac{2\pi}{2^n}+isin\frac{2\pi}{2^n}$求复数$z=\sum_{k=0}^{2^n-1}cos(k)f(w^k)$

小 A 并不会做。为了给小 B 寄蒜，他找到你并请你帮他计算出z的值。

输入

第一行，一个正整数 n（1 ≤ n ≤ 16）。

第二行，2^n 个整数 $c_0,c_1,\ldots,c_{2n-1}$（$0\le|x|\le100$），表示 $f(x)$的系数。

输出

一行，两个实数，分别表示 z 的实部 $R(z)$ 与 z 的虚部 $I(z)$，保留两位小数。

输入样例

```
4
99 1 1 4 5 1 4 99 1 9 1 9 8 1 0 99
```

输出样例

```
293.94 120.18
```

解题思路

如果你非常了解快速傅里叶变换（FFT），那么你很容易就能发现这道题和DFT非常类似。如需要算出每一个 $f(w^k)$只需要 $O(nlgn)$ 的时间复杂度，如需算出 z 只需要 $O(n)$的时

间复杂度遍历一下所有的$f(w^k)$（注：这里面的n与题干中的n不同）。那么这道题就很好做了！

快速傅里叶变换（FFT）

一些比较好的学习资料：

学习资料	地址	优点	缺点
课程ppt	移步课程群	讲解细致	缺少板子，英文阅读起来困难
一篇博客	点这里	讲解细致	讲解没使用线性代数、板子不一定正确
OI Wiki	点这里	板子很不错	讲解得比较粗糙，不适合初学

DFT板子

对于下面两个版本，我们约定：传入 DFT 函数的数组长度必须为2的幂。这里的板子来源于上文提到的OI-Wiki，差别不会特别大。

递归版本

```
void DFT(complex<double>*a, int n, int op) {
    if (!n) return;
    complex<double>a0[n], a1[n];
    for (int i = 0; i < n; i++)
        a0[i] = a[i << 1], a1[i] = a[i << 1 | 1];
    DFT(a0, n >> 1, op);
    DFT(a1, n >> 1, op);
    complex<double> wn(cos(PI / n), sin(PI / n) * op), w(1, 0);
    for (int i = 0; i < n; i++, w *= wn)
        a[i] = a0[i] + w * a1[i], a[i + n] = a0[i] - w * a1[i];
}
```

非递归版本

```
void change(complex y[], int len) {
    int k;
    for (int i = 1, j = len / 2; i < len - 1; i++) {
        if (i < j) swap(y[i], y[j]);
        k = len / 2;
        while (j >= k)
            j = j - k, k = k / 2;
```

```
        if (j < k) j += k;
    }
}
void DFT(complex y[], int len, int on) {
    change(y, len);
    for (int h = 2; h <= len; h <<= 1) {
        complex wn(cos(2 * PI / h), sin(on * 2 * PI / h));
        for (int j = 0; j < len; j += h) {
            complex w(1, 0);
            for (int k = j; k < j + h / 2; k++) {
                complex u = y[k];
                complex t = w * y[k + h / 2];
                y[k] = u + t, y[k + h / 2] = u - t, w = w * wn;
            }
        }
    }
    if (on == -1) {
        for (int i = 0; i < len; i++)
            y[i].x /= len;
    }
}
```

代码实现

因为数据点比较弱，所以用递归版本就能过，但建议大家多练习使用非递归版本，避免爆栈或爆空间。

```
#include<bits/stdc++.h>
using namespace std;
const double PI = acos(-1);
int n, p;
complex<double> c[65540], ans = {0, 0};
void DFT(complex<double>*a, int n, int op) {
    if (!n) return;
    complex<double>a0[n], a1[n];
    for (int i = 0; i < n; i++)
        a0[i] = a[i << 1], a1[i] = a[i << 1 | 1];
    DFT(a0, n >> 1, op);
    DFT(a1, n >> 1, op);
    complex<double> wn(cos(PI / n), sin(PI / n) * op), w(1, 0);
    for (int i = 0; i < n; i++, w *= wn)
        a[i] = a0[i] + w * a1[i], a[i + n] = a0[i] - w * a1[i];
```

```
}
int main() {
    ios::sync_with_stdio(false);
    cin.tie(0);
    cin >> n;
    p = 1 << n;
    for (int i = 0; i < p; i++)
        cin >> c[i];
    FFT(c, p >> 1, 1);
    for (int i = 0; i < p; i++)
        ans += cos(i) * c[i];
    printf("%.2lf %.2lf", ans.real(), ans.imag());
    return 0;
}
```