FFT 对某种特定 DP 的优化

李凌霄，王迪

成都七中
csimstu@gmail.com, zcwwzdjn@hotmail.com

July 4, 2012

第一部分:多项式与 FFT

多项式

一个多项式 (polynomial)$A(x)$ 是一个函数[1]:

$$A(x) = \sum_{j=0}^{n-1} a_j x^j$$

[1]假定多项式长度相等,均为 n,且 $n = 2^t$

多项式

一个多项式 (polynomial)$A(x)$ 是一个函数[1]:

$$A(x) = \sum_{j=0}^{n-1} a_j x^j$$

多项式的加法、乘法:

$$C(x) = A(x) + B(x) \to c_j = a_j + b_j$$

$$C(x) = A(x)B(x) \to c_j = \sum_{k=0}^{j} a_k b_{j-k}$$

[1] 假定多项式长度相等,均为 n,且 $n = 2^t$

多项式的表示

多项式的表示

- 系数表示法 (Coffecient representation)

$$A(x) = \sum_{j=0}^{n-1} a_j x^j$$

多项式的表示

- 系数表示法 (Coffecient representation)

$$A(x) = \sum_{j=0}^{n-1} a_j x^j$$

- 点值表示法 (Point-value representation)
  用 n 个二元组来表示一个多项式:

$$\{(x_0, y_0), (x_1, y_1), (x_2, y_2), \ldots, (x_{n-1}, y_{n-1})\}$$

满足对所有的 k，x_k 互不相等且 $y_k = A(x_k)$。

多项式的表示

- 系数表示法 (Coffecient representation)

$$A(x) = \sum_{j=0}^{n-1} a_j x^j$$

- 点值表示法 (Point-value representation)
  用 n 个二元组来表示一个多项式:

$$\{(x_0, y_0), (x_1, y_1), (x_2, y_2), \ldots, (x_{n-1}, y_{n-1})\}$$

满足对所有的 k，x_k 互不相等且 $y_k = A(x_k)$。 可以证明，点值表示法可以对应一个唯一的系数表示法。

对比两种表示方法

对比两种表示方法

- 求值

$$O(n) < O(?)$$

对比两种表示方法

- 求值

$$O(n) < O(?)$$

- 多项式加法

$$O(n) = O(n)$$

对比两种表示方法

- 求值

$$O(n) < O(?)$$

- 多项式加法

$$O(n) = O(n)$$

- 多项式乘法

$$O(n^2) > O(n)$$

对比两种表示方法

- 求值

$$O(n) < O(?)$$

- 多项式加法

$$O(n) = O(n)$$

- 多项式乘法

$$O(n^2) > O(n)$$

- 多项式自乘 k 次

$$O(kn^2) \gg O(n\log k)$$

对比两种表示方法

- 求值

$$O(n) < O(?)$$

- 多项式加法

$$O(n) = O(n)$$

- 多项式乘法

$$O(n^2) > O(n)$$

- 多项式自乘 k 次

$$O(kn^2) \gg O(n\log k)$$

点值表示法是多项式运算的核心。

两种表示方法相互转化

两种表示方法相互转化

- 系数 to 点值

两种表示方法相互转化

- 系数 to 点值
  任取 n 个不同的数作为 x，$O(n^2)$

两种表示方法相互转化

- 系数 to 点值

  任取 n 个不同的数作为 x，$O(n^2)$
- 点值 to 系数

两种表示方法相互转化

- 系数 to 点值

  任取 n 个不同的数作为 x，$O(n^2)$
- 点值 to 系数

  解方程 $O(n^3)$

两种表示方法相互转化

- 系数 to 点值

  任取 n 个不同的数作为 x，$O(n^2)$
- 点值 to 系数

  解方程 $O(n^3)$
  Lagrange’s formula $O(n^2)$

两种表示方法相互转化

- 系数 to 点值

  任取 n 个不同的数作为 x，$O(n^2)$
- 点值 to 系数

  解方程 $O(n^3)$
  Lagrange's formula $O(n^2)$

毫无疑问，以上方法都太慢了！

引入 FFT

快速傅里叶变化 (Fast Fourier Transforms) 是一种能在 $O(n \log n)$ 时间内实现两种转化的技术。

快速傅里叶变化 (Fast Fourier Transforms) 是一种能在 $O(n\log n)$ 时间内实现两种转化的技术。
我们注意到，$\{x\}$ 的选值是可以人为操控的，还有很大的利用余地。

引入 FFT

快速傅里叶变化 (Fast Fourier Transforms) 是一种能在 $O(n\log n)$ 时间内实现两种转化的技术。
我们注意到，$\{x\}$ 的选值是可以人为操控的，还有很大的利用余地。一种比较好的方法是选用 $\omega^n = 1$ 的 n 个复数根。

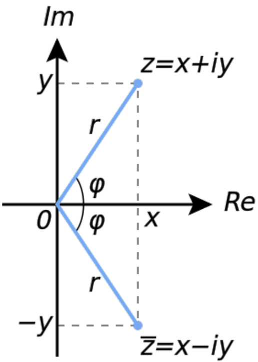
复数基础

复数基础

复单位：i,定义 $i^2 = -1$。

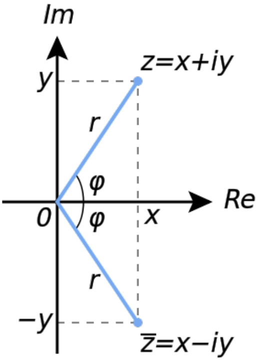
复数基础

复单位：i，定义 $i^2 = -1$。
复平面：



复数基础

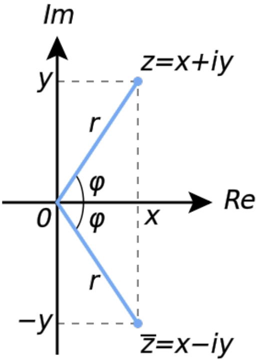
复单位：i，定义 $i^2 = -1$。
复平面：



复数的四则运算：直接推。

复数基础

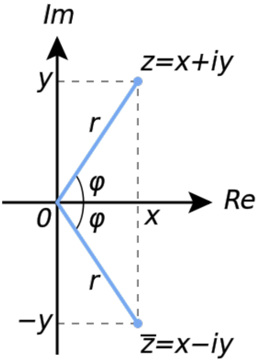
复单位：i,定义 $i^2 = -1$。
复平面：



复数的四则运算:直接推。
如何实现？

复数基础

复单位：i，定义 $i^2 = -1$。
复平面：



复数的四则运算：直接推。
如何实现？
重载运算 (封装)。

复数基础

欧拉公式:

$$e^{i\theta} = \cos\theta + i\sin\theta$$

单位根：

$$\omega_n = e^{2\pi i/n}$$

复数基础

欧拉公式:

$$e^{i\theta} = \cos\theta + i\sin\theta$$

单位根：

$$\omega_n = e^{2\pi i/n}$$

$\omega^n = 1$ 的 n 个复数根可以表示为

$$\omega_n^0, \omega_n^1, \ldots, \omega_n^{n-1}$$

复数基础

欧拉公式:

$$e^{i\theta} = \cos\theta + i\sin\theta$$

单位根：

$$\omega_n = e^{2\pi i/n}$$

$\omega^n = 1$ 的 n 个复数根可以表示为

$$\omega_n^0, \omega_n^1, \ldots, \omega_n^{n-1}$$

优秀性质：

复数基础

欧拉公式:

$$e^{i\theta} = \cos\theta + i\sin\theta$$

单位根：

$$\omega_n = e^{2\pi i/n}$$

$\omega^n = 1$ 的 n 个复数根可以表示为

$$\omega_n^0, \omega_n^1, \ldots, \omega_n^{n-1}$$

优秀性质：

- 周期性：$\omega_n^{j+kn} = \omega_n^j$

复数基础

欧拉公式:

$$e^{i\theta} = \cos\theta + i\sin\theta$$

单位根：

$$\omega_n = e^{2\pi i/n}$$

$\omega^n = 1$ 的 n 个复数根可以表示为

$$\omega_n^0, \omega_n^1, \ldots, \omega_n^{n-1}$$

优秀性质：

- 周期性：$\omega_n^{j+kn} = \omega_n^j$
- 消除性：$\omega_{dn}^{dk} = \omega_n^k$

复数基础

欧拉公式:

$$e^{i\theta} = \cos\theta + i\sin\theta$$

单位根：

$$\omega_n = e^{2\pi i/n}$$

$\omega^n = 1$ 的 n 个复数根可以表示为

$$\omega_n^0, \omega_n^1, \ldots, \omega_n^{n-1}$$

优秀性质：

- 周期性：$\omega_n^{j+kn} = \omega_n^j$
- 消除性：$\omega_{dn}^{dk} = \omega_n^k$
- 和为零：$\sum_{j=0}^{n-1} (\omega_n^k)^j = 0$

It's show time

约定系数 to 点值称为 DFT,点值 to 系数称为 IDFT。

It's show time

约定系数 to 点值称为 DFT，点值 to 系数称为 IDFT。 对每个 k，我们实际上是要计算 $y_k = A(\omega_n^k) = \sum_{j=0}^{n-1} a_j \omega_n^{kj}$。

It's show time

约定系数 to 点值称为 DFT，点值 to 系数称为 IDFT。 对每个 k，我们实际上是要计算 $y_k = A(\omega_n^k) = \sum_{j=0}^{n-1} a_j \omega_n^{kj}$。
将多项式 A 按奇偶分解：

It's show time

约定系数 to 点值称为 DFT，点值 to 系数称为 IDFT。 对每个 k，我们实际上是要计算 $y_k = A(\omega_n^k) = \sum_{j=0}^{n-1} a_j \omega_n^{kj}$。
将多项式 A 按奇偶分解：

$$A^{[0]}(x) = a_0 + a_2 x + a_4 x^2 + \ldots + a_{n-2} x^{n/2-1}$$

It's show time

约定系数 to 点值称为 DFT，点值 to 系数称为 IDFT。 对每个 k，我们实际上是要计算 $y_k = A(\omega_n^k) = \sum_{j=0}^{n-1} a_j \omega_n^{kj}$。
将多项式 A 按奇偶分解：

$$A^{[0]}(x) = a_0 + a_2 x + a_4 x^2 + \ldots + a_{n-2} x^{n/2-1}$$

$$A^{[1]}(x) = a_1 + a_3 x + a_5 x^2 + \ldots + a_{n-1} x^{n/2-1}$$

It’s show time

约定系数 to 点值称为 DFT，点值 to 系数称为 IDFT。 对每个 k，我们实际上是要计算 $y_k = A(\omega_n^k) = \sum_{j=0}^{n-1} a_j \omega_n^{kj}$。

将多项式 A 按奇偶分解：

$$A^{[0]}(x) = a_0 + a_2 x + a_4 x^2 + \ldots + a_{n-2} x^{n/2-1}$$

$$A^{[1]}(x) = a_1 + a_3 x + a_5 x^2 + \ldots + a_{n-1} x^{n/2-1}$$

于是：

$$A(x) = A^{[0]}(x^2) + xA^{[1]}(x^2)$$

。

约定系数 to 点值称为 DFT，点值 to 系数称为 IDFT。 对每个 k，我们实际上是要计算 $y_k = A(\omega_n^k) = \sum_{j=0}^{n-1} a_j \omega_n^{kj}$。
将多项式 A 按奇偶分解：

$$A^{[0]}(x) = a_0 + a_2 x + a_4 x^2 + \ldots + a_{n-2} x^{n/2-1}$$

$$A^{[1]}(x) = a_1 + a_3 x + a_5 x^2 + \ldots + a_{n-1} x^{n/2-1}$$

于是：

$$A(x) = A^{[0]}(x^2) + x A^{[1]}(x^2)$$

。 将 $x = \omega_n^k$ 代入：

$$A(\omega_n^k) = A^{[0]}(\omega_n^{2k}) + \omega_n^k A^{[1]}(\omega_n^{2k})$$

。

It's show time

约定系数 to 点值称为 DFT，点值 to 系数称为 IDFT。 对每个 k，我们实际上是要计算 $y_k = A(\omega_n^k) = \sum_{j=0}^{n-1} a_j \omega_n^{kj}$。
将多项式 A 按奇偶分解：

$$A^{[0]}(x) = a_0 + a_2 x + a_4 x^2 + \ldots + a_{n-2} x^{n/2-1}$$

$$A^{[1]}(x) = a_1 + a_3 x + a_5 x^2 + \ldots + a_{n-1} x^{n/2-1}$$

于是：

$$A(x) = A^{[0]}(x^2) + x A^{[1]}(x^2)$$

。 将 $x = \omega_n^k$ 代入:

$$A(\omega_n^k) = A^{[0]}(\omega_n^{2k}) + \omega_n^k A^{[1]}(\omega_n^{2k})$$

。 由消除性，可得：

$$A(\omega_n^k) = A^{[0]}(\omega_{n/2}^k) + \omega_n^k A^{[1]}(\omega_{n/2}^k)$$

。

It's show time

约定系数 to 点值称为 DFT，点值 to 系数称为 IDFT。 对每个 k，我们实际上是要计算 $y_k = A(\omega_n^k) = \sum_{j=0}^{n-1} a_j \omega_n^{kj}$。
将多项式 A 按奇偶分解：

$$A^{[0]}(x) = a_0 + a_2 x + a_4 x^2 + \ldots + a_{n-2} x^{n/2-1}$$

$$A^{[1]}(x) = a_1 + a_3 x + a_5 x^2 + \ldots + a_{n-1} x^{n/2-1}$$

于是：

$$A(x) = A^{[0]}(x^2) + x A^{[1]}(x^2)$$

。 将 $x = \omega_n^k$ 代入:

$$A(\omega_n^k) = A^{[0]}(\omega_n^{2k}) + \omega_n^k A^{[1]}(\omega_n^{2k})$$

。 由消除性，可得：

$$A(\omega_n^k) = A^{[0]}(\omega_{n/2}^k) + \omega_n^k A^{[1]}(\omega_{n/2}^k)$$

。 注意到多项式 $A^{[0]}$ 与 $A^{[1]}$ 都只有 $n/2$ 项，若 $k \geq n/2$ 有 $\omega_{n/2}^k = \omega_{n/2}^{k-n/2}$。

It's show time

约定系数 to 点值称为 DFT，点值 to 系数称为 IDFT。 对每个 k，我们实际上是要计算 $y_k = A(\omega_n^k) = \sum_{j=0}^{n-1} a_j \omega_n^{kj}$。
将多项式 A 按奇偶分解：

$$A^{[0]}(x) = a_0 + a_2 x + a_4 x^2 + \ldots + a_{n-2} x^{n/2-1}$$

$$A^{[1]}(x) = a_1 + a_3 x + a_5 x^2 + \ldots + a_{n-1} x^{n/2-1}$$

于是：

$$A(x) = A^{[0]}(x^2) + xA^{[1]}(x^2)$$

。 将 $x = \omega_n^k$ 代入:

$$A(\omega_n^k) = A^{[0]}(\omega_n^{2k}) + \omega_n^k A^{[1]}(\omega_n^{2k})$$

。 由消除性，可得：

$$A(\omega_n^k) = A^{[0]}(\omega_{n/2}^k) + \omega_n^k A^{[1]}(\omega_{n/2}^k)$$

。 注意到多项式 $A^{[0]}$ 与 $A^{[1]}$ 都只有 $n/2$ 项，若 $k \geq n/2$ 有 $\omega_{n/2}^k = \omega_{n/2}^{k-n/2}$。 问题被完美的转化为了子问题！系数到点值的转化就能在 $O(n \log n)$ 时间内进行了。

逆转化

逆转化

一个结论(证明略去)：

$$a_j = \frac{1}{n}\sum_{k=0}^{n-1} y_k \omega_n^{-kj}$$

逆转化

一个结论（证明略去）：

$$a_j = \frac{1}{n}\sum_{k=0}^{n-1} y_k \omega_n^{-kj}$$

跟刚才的形式如出一辙。我们只需要将 ω_n 替换为 ω_n^{-1}，最后全部除以 n 就解决了。

基本的 FFT 只需 10 行:

```
void fft(int delta, int step, int size, int flag) {
    if ( size == 1 ) { res[delta] = coef[delta]; return; }
    fft(delta, step * 2, size / 2, flag);
    fft(delta + step, step * 2, size / 2, flag);
   Complex acc(1, 0), pri(cos(flag * 2 * PI / size), sin(flag * 2 * PI / size));
    for ( int i = 0; i < size / 2; i ++, acc = acc * pri )
        tmp[delta + i * step] = res[delta + i * step * 2] + acc * res[delta +
step + i * step * 2];
    for ( int i = size / 2; i < size; i ++, acc = acc * pri )
        tmp[delta + i * step] = res[delta + (i - size / 2) * step * 2] + acc *
res[delta + step + (i - size / 2) * step * 2];
    for ( int i = 0; i < size; i ++ ) res[delta + i * step] = tmp[delta + i *
step];
}
```

第二部分:例题 *6

eg0. 高精度乘法

Example

给两个 50000 位左右的正整数, 算它们的乘积。

eg0. 高精度乘法

Example

给两个 50000 位左右的正整数, 算它们的乘积。

Solution

把正整数表示成多项式 $f(x)$, 使得代入 $x = 10$ 得到原来的数, 于是原来整数的每一位都是多项式中它对应的那一项的系数,然后利用 *FFT* 做多项式乘法即可。
大家可以到 *HDU1402* 验证自己的正确性。

eg0. 高精度乘法

Example

给两个 50000 位左右的正整数, 算它们的乘积。

Solution

把正整数表示成多项式 $f(x)$, 使得代入 $x = 10$ 得到原来的数, 于是原来整数的每一位都是多项式中它对应的那一项的系数,然后利用 *FFT* 做多项式乘法即可。
大家可以到 *HDU1402* 验证自己的正确性。

ext: 可以考虑压位来加速。

eg0. 高精度乘法

Example

给两个 50000 位左右的正整数, 算它们的乘积。

Solution

把正整数表示成多项式 $f(x)$, 使得代入 $x = 10$ 得到原来的数, 于是原来整数的每一位都是多项式中它对应的那一项的系数, 然后利用 *FFT* 做多项式乘法即可。
大家可以到 *HDU1402* 验证自己的正确性。

ext: 可以考虑压位来加速。
extext:FFT 后得到的数组中实数的范围会不会超过 int?

eg0. 高精度乘法

Example

给两个 50000 位左右的正整数, 算它们的乘积。

Solution

把正整数表示成多项式 $f(x)$, 使得代入 $x = 10$ 得到原来的数, 于是原来整数的每一位都是多项式中它对应的那一项的系数,然后利用 *FFT* 做多项式乘法即可。

大家可以到 *HDU1402* 验证自己的正确性。

ext: 可以考虑压位来加速。
extext:FFT 后得到的数组中实数的范围会不会超过 int?
extextext: 有没有办法可以在一定程度上消除精度误差?

eg1.SPOJ TSUM

Example

给定 40000 个绝对值不超过 20000 且互不相同的整数, 考虑下标三元组 (a, b, c), 其中 a, b, c 互不相同, 其权值为对应 3 个数的和, 对每个可能的权值统计无序三元组 (即 $(a, b, c) = (b, a, c) = \ldots$)。

Hint: 若 a, b, c 有序且不必互不相同, 有没有办法?

Example

定义一个算法,$f[i+1][(j+k)\%n] = \sum f[i][j] \times g[k]\%(n+1)$, 保证 $n+1$ 为质数, 且 n 最大的质因子不超过 7。给出 $n, c, f[0]$ 数组和 g 数组, 计算 $f[c]$ 数组, 其中 $n \le 5 \times 10^5, c \le 10^9$。

Hint: 先不考虑答案要模 $n+1$, 如何计算 $f[i+1][(j+k)\%n] = \sum f[i][j] \times g[k]$?

eg2. CTSC2010 性能优化

ext: 注意到这题的 n 不是 2 的幂,但满足最大的质因子不超过 7,怎么破?

eg2. CTSC2010 性能优化

ext: 注意到这题的 n 不是 2 的幂,但满足最大的质因子不超过 7,怎么破?
extext: 若前两个问题都解决了, 可以考虑做今年集训队答辩中伍一鸣的 Binomial。

eg3. TopCoder SRM 518 DIV1 Nim

Example

两人玩 Nim 游戏, 桌上有不超过 10^9 堆石子, 每堆石子的个数是质数且不超过 5×10^4。问有多少种情况使得后手必胜?

eg3. TopCoder SRM 518 DIV1 Nim

Example

两人玩 Nim 游戏, 桌上有不超过 10^9 堆石子, 每堆石子的个数是质数且不超过 5×10^4。问有多少种情况使得后手必胜?

Hint: 能不能写出 DP 的方程?

eg3. TopCoder SRM 518 DIV1 Nim

Example

两人玩 Nim 游戏, 桌上有不超过 10^9 堆石子, 每堆石子的个数是质数且不超过 5×10^4。问有多少种情况使得后手必胜?

Hint: 能不能写出 DP 的方程?
我们注意到这和多项式乘法很相似, 只不过下标变化从 $j+k$ 变成了 $j \oplus k$, 于是我们考虑转化。

Example

两人玩 Nim 游戏, 桌上有不超过 10^9 堆石子, 每堆石子的个数是质数且不超过 5×10^4。问有多少种情况使得后手必胜?

Hint: 能不能写出 DP 的方程?
我们注意到这和多项式乘法很相似, 只不过下标变化从 $j+k$ 变成了 $j \oplus k$, 于是我们考虑转化。
我们用 $\star$ 符号来表示两个数组的某种运算,$\star$ 的两个参数是数组,返回的也是数组。上式可以描述为 $f[i+1] = f[i] \star g$。
定义:
用大写字母表示数组, 用小写字母表示数字,$[\ldots]$ 表示数组, 比如 $A = [a, b]$。
数组的含义:下标从 0 开始,$A[x]$ 表示 xor 起来为 x 的方案数。
$(A \times B)[i] = A[i] \times B[i]$
$(A + B)[i] = A[i] + B[i]$

eg3. TopCoder SRM 518 DIV1 Nim

考虑多项式乘法,$\star$ 的含义就是对两个数组做多项式的乘法,在过程中我们有 $DFT(A \star B) = DFT(A) \times DFT(B)$, $IDFT(DFT(A)) = A$。

eg3. TopCoder SRM 518 DIV1 Nim

考虑多项式乘法,$\star$ 的含义就是对两个数组做多项式的乘法,在过程中我们有 $DFT(A \star B) = DFT(A) \times DFT(B)$, $IDFT(DFT(A)) = A$。
考虑现在的问题, 我们能否找到一种转换方式使得 $tf(A \star B) = tf(A) \times tf(B), utf(tf(A)) = A$?
猜测 tf 可能与 utf 类似, 我们重点考虑 tf。

考虑多项式乘法,$\star$ 的含义就是对两个数组做多项式的乘法,在过程中我们有
$DFT(A \star B) = DFT(A) \times DFT(B)$, $IDFT(DFT(A)) = A$。
考虑现在的问题, 我们能否找到一种转换方式使得
$tf(A \star B) = tf(A) \times tf(B), utf(tf(A)) = A$?
猜测 tf 可能与 utf 类似, 我们重点考虑 tf。
显然 $tf([a]) = [a]$。 $tf([a, b])$ 是什么呢?

eg3. TopCoder SRM 518 DIV1 Nim

Hint: 考虑 $[a,b] \star [c,d]$。
Hint: 猜测 $tf([a,b])$ 的两项都是关于 a 和 b 的线性函数。

eg3. TopCoder SRM 518 DIV1 Nim

Hint: 考虑 $[a, b] \star [c, d]$。
Hint: 猜测 $tf([a, b])$ 的两项都是关于 a 和 b 的线性函数。
其实 $tf([a, b]) = [a + b, a - b]$。

eg3. TopCoder SRM 518 DIV1 Nim

Hint: 考虑 $[a, b] \star [c, d]$。
Hint: 猜测 $tf([a, b])$ 的两项都是关于 a 和 b 的线性函数。
其实 $tf([a, b]) = [a + b, a - b]$。
进一步猜测:$tf([A, B]) = [tf(A) + tf(B), tf(A) - tf(B)]$,
证明?

eg3. TopCoder SRM 518 DIV1 Nim

Hint: 考虑 $[a, b] \star [c, d]$。
Hint: 猜测 $tf([a, b])$ 的两项都是关于 a 和 b 的线性函数。
其实 $tf([a, b]) = [a + b, a - b]$。
进一步猜测:$tf([A, B]) = [tf(A) + tf(B), tf(A) - tf(B)]$,
证明?
ext: 这题是异或、逻辑与、逻辑或等运算是不是也可以做呢?

eg4. TopCoder TCO12 Round2A EvenPaths

Example

给一个点数不超过 50 的 DAG, 有一些点是空地有一些是障碍, 还有一些是未探明的区域。设未探明的区域有 k 个,k 不会超过 32, 那么 DAG 就有 2^k 种可能。我们考虑这 2^k 种情况中, 有多少使得从 0 到 1 的路径数是偶数?

eg4. TopCoder TCO12 Round2A EvenPaths

Example

给一个点数不超过 50 的 DAG, 有一些点是空地有一些是障碍, 还有一些是未探明的区域。设未探明的区域有 k 个,k 不会超过 32, 那么 DAG 就有 2^k 种可能。我们考虑这 2^k 种情况中, 有多少使得从 0 到 1 的路径数是偶数?

Hint: 对一个确定的图, 如何计算 0 到 1 的路径数的奇偶性?

eg4. TopCoder TCO12 Round2A EvenPaths

Example

给一个点数不超过 50 的 DAG, 有一些点是空地有一些是障碍, 还有一些是未探明的区域。设未探明的区域有 k 个,k 不会超过 32, 那么 DAG 就有 2^k 种可能。我们考虑这 2^k 种情况中, 有多少使得从 0 到 1 的路径数是偶数?

Hint: 对一个确定的图, 如何计算 0 到 1 的路径数的奇偶性?
Hint: 枚举 2^{32} 次方是不可能的, 能否通过折半, 枚举 2^{16} 再将两边合并?

eg4. TopCoder TCO12 Round2A EvenPaths

Example

给一个点数不超过 50 的 DAG, 有一些点是空地有一些是障碍, 还有一些是未探明的区域。设未探明的区域有 k 个,k 不会超过 32, 那么 DAG 就有 2^k 种可能。我们考虑这 2^k 种情况中, 有多少使得从 0 到 1 的路径数是偶数?

Hint: 对一个确定的图, 如何计算 0 到 1 的路径数的奇偶性?
Hint: 枚举 2^{32} 次方是不可能的, 能否通过折半, 枚举 2^{16} 再将两边合并? 对图进行拓扑排序, 把 32 个点分为两个集合 A 和 B,A 包含前 16 个点,B 包含后 16 个点。
考虑 S 为 B 集合加上 1 号点。
这样有很好的一个性质, 就是从 0 走到 1, 必定经过 S 集合中至少 1 个点。
于是我们可以通过考虑从 0 到 1 碰到 S 中第一个点来对所有的路径进行计数。

Example

给一个点数不超过 50 的 DAG, 有一些点是空地有一些是障碍, 还有一些是未探明的区域。设未探明的区域有 k 个,k 不会超过 32, 那么 DAG 就有 2^k 种可能。我们考虑这 2^k 种情况中, 有多少使得从 0 到 1 的路径数是偶数?

Hint: 对一个确定的图, 如何计算 0 到 1 的路径数的奇偶性?
Hint: 枚举 2^{32} 次方是不可能的, 能否通过折半, 枚举 2^{16} 再将两边合并? 对图进行拓扑排序, 把 32 个点分为两个集合 A 和 B,A 包含前 16 个点,B 包含后 16 个点。
考虑 S 为 B 集合加上 1 号点。
这样有很好的一个性质, 就是从 0 走到 1, 必定经过 S 集合中至少 1 个点。
于是我们可以通过考虑从 0 到 1 碰到 S 中第一个点来对所有的路径进行计数。
注意到奇偶是一个 01 变量, 能不能通过状态压缩来进行动态规划?
考虑枚举 B 集合后,S 集合中的每点走到 1 号点路径数的奇偶性用一个长度为 17 的二进制数来表示。

eg4. TopCoder TCO12 Round2A EvenPaths

Example

给一个点数不超过 50 的 DAG, 有一些点是空地有一些是障碍, 还有一些是未探明的区域。设未探明的区域有 k 个,k 不会超过 32, 那么 DAG 就有 2^k 种可能。我们考虑这 2^k 种情况中, 有多少使得从 0 到 1 的路径数是偶数?

Hint: 对一个确定的图, 如何计算 0 到 1 的路径数的奇偶性?
Hint: 枚举 2^{32} 次方是不可能的, 能否通过折半, 枚举 2^{16} 再将两边合并? 对图进行拓扑排序, 把 32 个点分为两个集合 A 和 B,A 包含前 16 个点,B 包含后 16 个点。
考虑 S 为 B 集合加上 1 号点。
这样有很好的一个性质, 就是从 0 走到 1, 必定经过 S 集合中至少 1 个点。
于是我们可以通过考虑从 0 到 1 碰到 S 中第一个点来对所有的路径进行计数。
注意到奇偶是一个 01 变量, 能不能通过状态压缩来进行动态规划?
考虑枚举 B 集合后,S 集合中的每点走到 1 号点路径数的奇偶性用一个长度为 17 的二进制数来表示。
用 f[mask] 来表示使 mask 这种情况发生的 B 集合状态数。

Summary

以上几道题向我们展示了 FFT 可以优化某种特定的 DP，它们都具有以下共性：

Summary

以上几道题向我们展示了 FFT 可以优化某种特定的 DP,它们都具有以下共性:

1. 状态本质为一维。

Summary

以上几道题向我们展示了 FFT 可以优化某种特定的 DP,它们都具有以下共性:

1. 状态本质为一维。
2. 具有 $C[i \star j] = A[i] \times B[j]$ 的形式。 $\star$ 可以为加法、带模加法、按位异或、按位与,等等。

Summary

以上几道题向我们展示了 FFT 可以优化某种特定的 DP，它们都具有以下共性：

1. 状态本质为一维。
2. 具有 $C[i \star j] = A[i] \times B[j]$ 的形式。 $\star$ 可以为加法、带模加法、按位异或、按位与，等等。
3. 优化的过程都是转化成类点值表示法**解耦合**，然后每对点值**独立运算**，再转化回去。

Summary

以上几道题向我们展示了 FFT 可以优化某种特定的 DP，它们都具有以下共性：

1. 状态本质为一维。
2. 具有 $C[i\star j]=A[i]\times B[j]$ 的形式。 $\star$ 可以为加法、带模加法、按位异或、按位与，等等。
3. 优化的过程都是转化成类点值表示法**解耦合**，然后每对点值**独立运算**，再转化回去。
4. 转化与逆转化都借用了分而治之的思想。

More Exercises

CDOJ1714 Graph Game，CDOJ 是 UESTC 的 OJ。

CDOJ1714 Graph Game

Example

A 与 B 在一个无向图上做一个游戏。A 与 B 轮流操作,A 先手。每次删掉一条边,满足该边连接的两个点度和为偶数。谁不能找到这样合法的边谁就输。
现在,他们得到了一个包含 n 个点 m 条边的无向图。A 想知道,有多少子图使得他能获胜?(两人都以最优策略删边)
$2 \le n \le 40, 1 \le m \le 400$

CDOJ1714 Graph Game

Example

A 与 B 在一个无向图上做一个游戏。A 与 B 轮流操作,A 先手。每次删掉一条边,满足该边连接的两个点度和为偶数。谁不能找到这样合法的边谁就输。
现在,他们得到了一个包含 n 个点 m 条边的无向图。A 想知道,有多少子图使得他能获胜?(两人都以最优策略删边)
$2 \leq n \leq 40, 1 \leq m \leq 400$

先手必胜的条件?

CDOJ1714 Graph Game

Example

A 与 B 在一个无向图上做一个游戏。A 与 B 轮流操作,A 先手。每次删掉一条边,满足该边连接的两个点度和为偶数。谁不能找到这样合法的边谁就输。
现在,他们得到了一个包含 n 个点 m 条边的无向图。A 想知道,有多少子图使得他能获胜?(两人都以最优策略删边)
$2 \leq n \leq 40, 1 \leq m \leq 400$

先手必胜的条件? 如何利用 FFT 优化?