MATH 800 Assignment 3, Fall 2023

Instructor: Michael Monagan

Please hand in the assignment by 11pm Monday October 23rd.
Late Penalty -20% off for up to 24 hours late. Zero after that.
For Maple problems, please submit a printout of a Maple worksheet containing Maple code and the execution of examples.

Question 1 : Recurrences (10 marks)

(a) Solve the following recurrences using the rsolve command in Maple. The first is for the number of multiplications in the Berkowitz algorithm. The second is for the number of multiplications in the optimized FFT where the powers of ω have been pre-computed.

(1) $M(n) = M(n-1) + r + (r-1)(r^2+r) + \sum_{k=1}^{r} \sum_{j=0}^{r-k} 1$ where $r = n-1$ and with initial value $M(1) = 0$.

(2) $T(n) = 2T(n/2) + \frac{1}{2}n$ with initial value $T(1) = 0$.

(b) A first implementation of the FFT might allocate space for two intermediate arrays B and C, each of size $n/2$, to store $[a_0, a_2, a_4, \ldots, a_{n-2}]$ and $[a_1, a_3, a_5, \ldots, a_{n-1}]$ respectively. Thus we need space for n elements of the field F in total. But the FFT is recursive so the recursive calls will allocate more arrays. How many arrays are allocated in total? How much space is allocated in total?

Let $A(n)$ be the number of temporary arrays allocated. Let $S(n)$ be the number of elements of storage allocated for all the temporary arrays. Give recurrence relations for $A(n)$ and $S(n)$ and initial values for $A(1)$ and $B(1)$. Solve the recurrences by hand.

Question 2: The Fourier Transform (10 marks)

(a) (6 marks) Let $n = 2m$ and let ω be a primitive n'th root of unity. To apply the FFT recursively, we use the fact that ω^2 is a primitive m'th root of unity. Prove this.

Also, for $p = 97 = 3 \times 2^5 + 1$, find a primitve 8'th root of unity in $\mathbb{Z}_p$. You may use Maple's numtheory[primroot] command to find a primitive element α for $\mathbb{Z}_p$.

(b) (4 marks) Let ω be a primitive n'th root of unity. What is the Fourier Transform for the polynomials (i) $a(x) = x^n + c$ where c is a constant, and (ii) $a(x) = 1 + x + x^2 + \ldots + x^{n-1}$, i.e., what are the vectors $[a(1), a(\omega), \ldots, a(\omega^{n-1})]$? No justification needed.

Question 3 Coding the FFT and Fast Multiplication (15 marks)

(a) (10 marks)
Program the FFT in Maple as a recursive procedure. Your Maple procedure should take as input (n, A, p, ω, T) and any other values that you need.

Here n is a power of 2, A is an array of size n indexed from 0 for storing the input coefficients $a_0, a_1, \ldots, a_{n-1}$, p is a prime, ω is a primitive n'th root of unity in $\mathbb{Z}_p$ and T is a temporary array of size n indexed from 0.

Your procedure may not allocate any temporary arrays. Use T as needed. If you want to precompute an array $W = [1, \omega, \omega^2, \ldots, \omega^{n/2-1}]$ of the powers of ω to save $\frac{1}{2}$ the multiplications you may do so for 3 bonus marks.

Test your procedure on the following input. Let $A = [1, 2, 3, 4, 3, 2, 1, 0]$, $p = 97$ and w be the primitive 8'th root of unity. To check that your output B is correct, verify that $FFT(n, B, p, \omega^{-1}, T) = nA \bmod p$.

(c) (5 marks)
Let $a(x) = -x^3 + 3x + 1$ and $b(x) = 2x^4 - 3x^3 - 2x^2 + x + 1$ be polynomials in $\mathbb{Z}_{97}[x]$. Use your FFT code from part (a) to calculate the product of $c(x) = a(x)b(x)$.

Question 4 : Fast Division (10 marks)

Consider computing the quotient of $a \div b$ in $F[x]$. To use the fast method we need to compute f^{-1} to $O(x^n)$ where $n = \deg a - \deg b + 1$ and $f = b^r$. Write a Maple procedure `FastNewton(f,x,n,p)` that computes f^{-1} to $O(x^n)$ for $F = \mathbb{Z}_p$ using a Newton iteration.

Use `Expand(...) mod p;` for the polynomial multiplications so you get Maple's fast multiplication. To make the Newton iteration efficient when n is not a power of 2, compute $y = f^{-1}$ recursively to order $O(x^{\lceil n/2 \rceil})$. To truncate a polynomial b modulo x^n you could use `rem(b,x^n,x)`. Use `convert(taylor(b,x,n),polynom)` instead which is more efficient.

Test your algorithm on the following problem in $\mathbb{Z}_p[x]$.

```
> p := 11;
> f := 3+x+4*x^3+x^5;
> FastNewton(f,x,6,p);
```

$$10\,x^5 + 7\,x^4 + 10\,x^3 + 9\,x^2 + 6\,x + 4$$

Question 5 Applying the FFT (10 marks)

Let $a(x) = \sum_{i=0}^{n-1} a_i x^i$. Let $F_\omega : F^n \to F^n$ denote the Fourier transform, that is

$$F_\omega([a_0, a_1, a_2, \ldots, a_{n-1}) = [a(1), a(\omega), a(\omega^2), \ldots, a(\omega^{n-1})].$$

Let F_ω^{-1} denote the inverse transform, i.e.,

$$F_\omega^{-1}(b) = \frac{1}{n} F_{\omega^{-1}}(b).$$

(a) (5 marks)

Suppose we have three polynomials $a, b, c \in F[x]$ of degree d and we want to compute $g(x) = 3ab + 5ac + 7bc$. If use use the FFT to multiply the three products ab, ac and bc, separately, how many calls to the FFT algorithm will we make?

Reorganize the computation to optimize the number of calls to the FFT. Describe your algorithm in terms of vector operations in F^n and calls to F_ω and F_ω^{-1}. How many calls to the FFT does your optimized algorithm make?

(b) (5 marks) In the product tree algorithm we need to multiply monic polynomials a and b of degree $d = 2^i$ for some i. Let $c = ab = x^{2d} + \Delta(x)$ where $\deg(\Delta) < 2d$. Using our FFT multiplication algorithm we need $n > \deg(c) = 2d$ so we would need FFTs of size $n = 4d$. Let's try do the multiplication using FFTs of size $2d$ which will save a factor of two in time and space. Let $n = 2d$ and let ω be a primitive n'th root of unity. What is $F_\omega(c)$? Explain how can you modify the FFT multiplication algorithm so that you can recover $c(x)$ from $F_\omega(c)$ efficiently.