1995 Paper 13 Question 13

Complexity Theory

It turns out that the following program, when run using floating-point arithmetic that remains accurate to N decimal places, will compute and print the value of π correct to almost N places. The loop (which has as its main part a step which replaces the values in `a` and `b` by their arithmetic and geometric means, respectively) will be traversed about $\log(N)$ times.

```
a := 1;
b := 1/sqrt(2);
u := 1/4;
x := 1;
pn := 4;
do { p := pn;
     y := a; a := (a+b)/2; b := sqrt(y*b);
     u := u-x*(a-y)*(a-y);
     x := 2*x;
     pn := a**2/u; } while (pn<p);
print(p);
```

You are provided with procedures that can compute Fourier Transforms and their inverses with a transform on k points (using floating-point arithmetic), taking time proportional to $k \log k$. Explain how you could implement the high-precision arithmetic needed to make this program run fast. Do not discuss how the Fourier transform will be implemented — just how it is used, and assume that the floating-point accuracy achieved by the transform will be adequate for your purposes.

[14 marks]

Overall how long (as a function of N) would you expect the complete program to take to run? [6 marks]

You do not need to understand how or why this particular calculation arrives at a value for π, or why the loop is executed only $\log(N)$ times.