EC 504 – Fall 2021 – Homework Zero

Due Thursday, Sept 19 more or less at the beginning of class but I expect some will need help in class. Always unexpected things getting started. This is a zero grade practice problem to see if everyone can use the SCC and submit a source code to your file at `/projectnb/alg504/yourname/HW0` **Then Krishna and I will run a script to see if we can look at it. In the mean time you can start to read the CLRS Chapters 2, 2, 3 and 4 in our text.**

1 P^3: Practice Power Pragram

The exercise is to go to the GitHub and get the prototype code in the file `EC504_2021/HW0_codes`. The code is a main program that calls 3 function to compute the power x^N for large integer N. The frist function is the standard C routine. (It actually has to convert the integer power to a floating point. Argh so silly). The next is the slow one. So slow that I stop it before it bores you. The final one the fast multiple squaring routine – but I left out a line for you to complete. *That is the exercise – One line.*

```
double cPower(double x, long int N)
{
  return pow(x, (double)N);
}

double slowPower(double x, long int N)
{
  double pow = 1.0;
  int i;
  for( i = 0;  i < N && i < 1000000000; i++)
    {
  pow *= x;
     }
if(i < N)   cout <<"Slow Failed with iteration stop at i = " << i << endl;
  return pow ;
}

double  fastPower(double x, long int N)
{
  double square = x;
  double pow = 1.0;
  while(N > 0)
    {
      // cout<< " N%2 = "<<  N%2 << " N/2 = "<<  N << endl;
      if(N%2) pow = 1.0 ;
      N = N/2;
    }
  return pow;
```

```
}
```

The program is compiled automatically by putting `myPower.cpp` in a directory with `makefile` and typing `make -k` on the command line. Then it will run by typing `.\power`

Ok at this point you make a directory `/projectnb/alg504/yourname/HW0` and move it there. Do this right away and we can see if everything is set to go. Then you can fix up the fastPower and see how much faster it is.

You should try changing the power N for fun. Actually

```
N =  N0 - 50 +  rand()%100;
```

would give you a random size `N =  \in [N0 - 50, N0 +50]` centered at $N0$. So if your bored or really ambitious – not necessary – you could run a loop over may instances of each range and see how the algorithm on average scales. You might try to make a graph of the excution time vs the sized N and see how the time scale: between $\mathcal{O}(N)$ and $\mathcal{O}(logN)$ algorithms. BUT this is getting ahead of the course.

Have fun.

Rich