CSE525 Monsoon 2020
Homework 1 Problem 1

Pramil Panjawani (19008)
Ashima Garg (19003)

(a) How many multiplications does this algorithm perform?

Solution: Total number of multiplications performed will be $n-1$.

Proof by induction:
Induction Hypothesis P(n): F(n)=n-1
Base case F(1)=0 (1!=1)
Hence P(1) is true

Induction Step
P(k): F(k)=k-1
Let P(K) be true
(n+1)!=(n+1)*n! P(k+1): F(k+1)=(k+1)*F(k) => 1+(k-1)
P(k+1):F(k+1)=k

Hence Proved

∎

(b) How many bits are required to write n! in binary? Express your answer in the form $\Theta(f(n))$, for some familiar function f (n). [Hint: $(n/2)^{n/2} < n! < n^n$.]

Solution:

Number of bits in n b(n) = $\lfloor log(n) \rfloor + 1$

Number of bits in n! b(n!) $= \lfloor log(n!) \rfloor + 1 \approx log(n!)$

(n/2)$^{n/2} < n! < n^n$

taking log on both side
(n/2)*log(n/2) < log(n!) < n*log(n)
(n/2)*(log(n)-log(2)) < log(n!) < n*log(n)
(n/2)*(log(n)-1) < log(n!) < n*log(n)

For large values of n
log(n)-1 $\approx log(n)$
$(n/2) * (log(n)) < log(n!) < n * log(n)$

Hence $\theta(f(n)) = nlog(n)$

■

(c) Your answer to (b) should convince you that the number of multiplications is not a good estimate of the actual running time of FACTORIAL. What is the running time of FACTORIAL. We can multiply any k-digit number and any l-digit number in $O(k.l)$ time using either the lattice algorithm or duplation and mediation. What is the running time of FACTORIAL if we use this multiplication algorithm as a subroutine?

Solution: Recurrence relation is given by:
T(n) = T(n-1) + logn*log(T(n-1))
T(n) = T(n-1) + logn*log([(n-1)log(n-1)]) . . . using the result from part b
T(n) $\approx T(n-1) + (logn)^2$
$T(n) \approx T(n-2) + log(n-1)^2 + (logn)^2$
$T(n) \approx T(n-3) + log(n-2)^2 + log(n-1)^2 + (logn)^2$
.
.
.
$T(n) \geq T(n-k) + \sum_{i=1}^{n}(log(n-i)^2)$
$BaseCase T(0) = O(1)$
$n = k$
$\sum_{i=1}^{n}(log(n-k)^2) \leq n(logn)^2$
$T(n) \geq n(logn)^2 + C$
$T(n) \geq O(n(logn)^2)$

■

Pramil Panjawani (19008)
Ashima Garg (19003)

(d) Recursive algorithm also computes the factorial function, but using a different grouping of the multiplications: What is the running time of FALLING(n, n) if we use grade-school multiplication? [Hint: As usual, ignore the floors and ceilings.]

Solution: Using grade-school multiplication, the time complexity of multiplying a k-digit number with l-digit number is $O(k.l)$.
Also, from part b, we know that the number of digits in $n!$ is $O(nlogn)$.

$$T(n,m) = T(n,\lfloor m/2\rfloor) + T(n-\lfloor m/2\rfloor,\lceil m/2\rceil) + log(T(n,m/2)).log(T(n-m/2,m/2))$$
$$T(n,n) = T(n,\lfloor n/2\rfloor) + T(n-\lfloor n/2\rfloor,\lceil n/2\rceil) + log(T(n,n/2)).log(T(n/2,n/2))$$
$$Takingupper\,bound,$$
$$T(n,n) \geq 2T(n,n/2) + (nlogn)*((n/2)*log(n/2))$$
$$T(n,n) \geq 2T(n,n/2) + (n^2*(logn)*(log(n/2)))$$
$$T(n,n) \geq 4T(n,n/4) + (n^2/2^0*2^1)*(logn)*(log(n/2)) + (n^2/(2^1*2^2))*(log(n/2^1)*log(n/2^2))$$
$$T(n,n) \geq (2^k)*T(n,n/2^k) + (n^2/2^0*2^1)*(logn)*(log(n/2))+$$
$$(n^2/(2^1*2^2))*(log(n/2^1)*log(n/2^2)) + \ldots + (n^2/2^k*2^{(k+1)})*log(n/2^k)*log(n/2^{(k+1)})$$
$$BaseCase:$$
$$T(n,0) = 1$$
$$T(n,1) = 1$$
$$k(levels) = logn$$
$$T(n,n) \geq n + \sum_{i=0}^{logn}((n/i)^2*(log(n/i)^2)$$
$$T(n,n) \geq n + n^2*\sum_{i=0}^{logn}((1/i)^2*(log(n/i)^2)$$
$$T(n,n) \geq n + n^2*\sum_{i=0}^{logn}((1/i)^2*(log(n)-log(i))^2)$$
$$T(n,n) \geq n + n^2*\sum_{i=0}^{logn}((1/i)^2*(log(n)^2+log(i)^2-2*log(n)*log(i))$$
$$\sum_{i=0}^{logn}((1/i)^2 = log(n)*(log(n)+1)*(2log(n)+1)/6 \approx (log(n))^3$$
$$T(n,n) \geq n + n^2*(3*log(n)^2/log(n)^3)$$
$$T(n,n) \geq n + (n^2/log(n))$$
$$T(n,n) \geq O(n^2/log(n))$$

■

Pramil Panjawani (19008)
Ashima Garg (19003)

(f) What are the running times of FACTORIAL(n) and FALLING(n, n) if we use the modified Karatsuba multiplication from part (e)?

Solution: 1. For FACTORIAL(n) and using the modified Karastuba Algorithm, Since, the time complexity of multiplying n-bit number and m-bit number when ($m \le n$) is, $O(m^{log3-1} * n) \dots$ (using the result from part e of the same question).
Recurrence relation is given by the following equation:

$$T(n) = T(n-1) + logn * log(T(n-1))$$
$$T(n) = T(n-1) + (logn)^{(log3-1)} * log(T(n-1))$$
$$T(n) = T(n-1) + (logn)^{(log3-1)} * log([(n-1)log(n-1)])$$
$$T(n) = T(n-1) + (logn)^{(log3-1)} * log(n-1) + (logn)^{(log3-1)} * log(log(n-1))$$
$$Since, log(n-1) \approx logn, loglogn << logn$$
$$Therefore,$$
$$T(n) = T(n-1) + (logn)^{log3}$$
$$T(n) = T(n-k) + (logn)^{log3} + (log(n-1))^{log3} + (log(n-2))^{log3} + \dots + (log(n-(k+1)))^{log3}$$
$$T(n) = (log1)^{log3} + (log2)^{log3} + (log3)^{log3} + \dots + (log(n-1))^{log3} + (logn)^{log3}$$
$$T(n) = O(n(logn)^{log3})$$

2. For FALLING(n, n) and using the modified Karastuba Algorithm, Since, the time complexity of multiplying n-bit number and m-bit number when ($m \le n$) is, $O(m^{log3-1} * n) \dots$ (using the result from part e of the same question).
Recurrence relation is given by the following equation:

$$T(n,n) = T(n,n/2) + T(n/2,n/2) + log(T(n,n/2)) * log(T(n/2,n/2))^{(log3-1)}$$
$$T(n,n) = 2T(n,n/2) + log(nlogn) * (log(n/2 * logn/2))^{(log3-1)}$$
$$T(n,n) \ge 2T(n,n/2) + log(nlogn)^{log3}$$
$$T(n,n) \ge 2^k * T(n,n/2^k) + log(nlogn)^{log3}$$
$$BaseCase : T(n,1) = 1$$
$$T(n,n) = logn * log(nlogn)^{log3}$$

Since, at each level maximum of $log(nlogn)^{log3}$ work is being done, and there are logn such levels. Here, approximation is taken for calculating the upper bound.

■