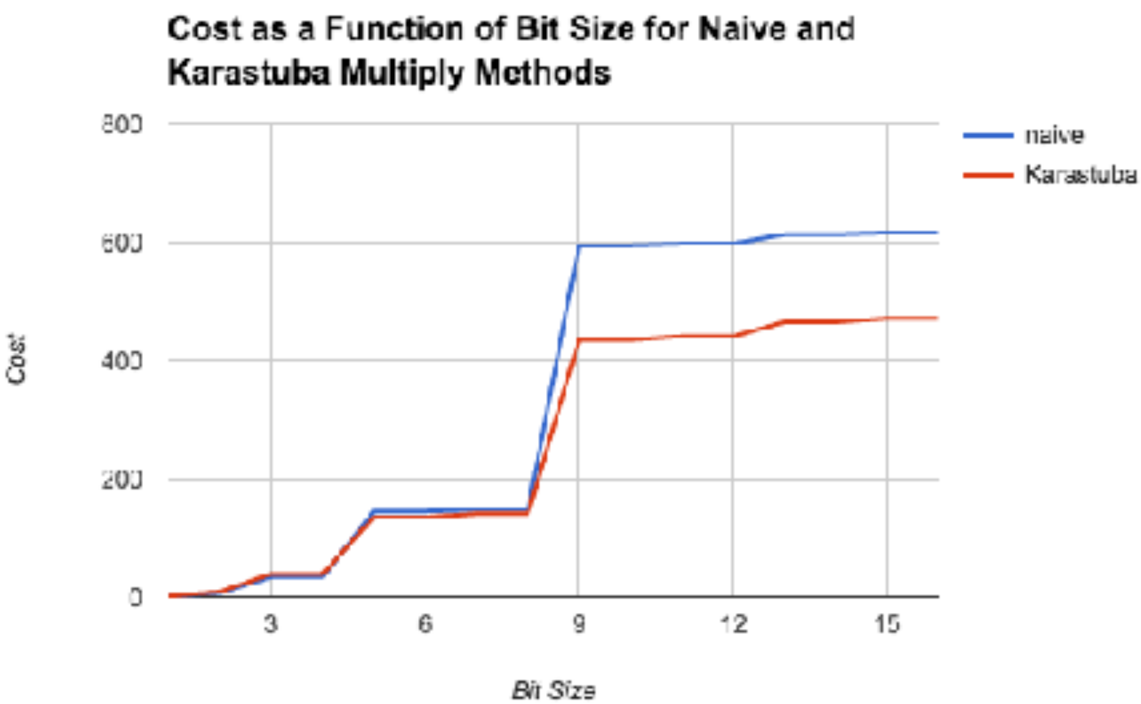

1. Based on results illustrated on the graph, the Karastuba multiply algorithm seems to be more efficient in terms of cost. The patterns of the two algorithms seem to be consistent with each other. The pattern seems to be that the cost remains the same for two bit sizes, and then there is an increase for the next two bit sizes.

2.

(a) T(n) = 25 T(n/5) + n

a = 25

b = 5

f(n) = n

log_b(a) = log_5(25) = 2

Using property 1:

if f(n) = O(n^(log_b(a - e))), for {e > 0}

then T(n) = Theta(n^(log_b(a)))

f(n) = O(n^(2 - e)), with {1 >= e > 0}

T(n) = Theta(n^2)

(b) $T(n) = 2\, T(n/3) + n \log(n)$

$a = 2$

$b = 3$

$f(n) = n \log(n)$

$\log_b(a) = \log_3(2)$

Using property 3:

if $f(n) = \Omega(n^{(\log_b(a + e))})$, for $\{e > 0\}$

and $f(n/b) \le cf(n)$, for $\{n0 > 0, n > n0, c < 1\}$

then $T(n) = \Omega(f(n))$

$f(n) = \Omega(\log_b(a) + e)$ for $\{2/3 \le c < 1\}$

$T(n) = \Omega(n \log(n))$

(c) $T(n) = T(3n/4) + 1$

$a = 1$

$b = 4/3$

$f(n) = 1$

$\log_b(a) = 1 \log(4/3)$

Using property 2:

if $f(n) = \Theta(n^{(\log_b(a))} \log^p(n))$

then $T(n) = heta(n^{(\log_b(a))} \log^{(p + 1)}(n))$

Let $p = 0$

$f(n) = \Theta(n^0 \log^0(n)) = \Theta(1)$

$T(n) = \Theta(\log(n))$

(d) $T(n) = 7\, T(n/3) + n^3$

$a = 7$

$b = 3$

$f(n) = n^3$

$\log_b(a) = \log_3(7)$

Using property 3:

if $f(n) = \Omega(n^{(\log_b(a + e))})$, for $\{e > 0\}$

and $f(n/b) \le cf(n)$, for $\{n0 > 0, n > n0, c < 1\}$

then $T(n) = \Omega(f(n))$

$f(n) = \Omega(n^{(\log_3(7) + e)})$ for $\{7(n/3)^3 \le cn^3\}$ and $\{0.26 \le c < 1\}$

$T(n) = \Omega(n^3)$

(e) $T(n) = T(n/2) + n(2 - \cos(n))$

$a = 1$

$b = 2$

f(n) = n(2 - cos(n))
log_b(a) = log_2(1) = 0

Using property 3:

if f(n) = Omega(n^(log_b(a + e))), for {e > 0}
and f(n/b) <= cf(n), for {n0 > 0, n > n0, c < 1}
then T(n) = Omega(f(n))

For this scenario, the master theorem DOES NOT APPLY

for n = 2pi * k (where k is odd and large)
for any n, c >= 3/2
Here, the regularity condition is violated

3.

Functions are defined as:
TA(n) = 7 TA(n/2) + n^2
TB(n) = a TB(n/4) + n^2

aA = 7
bA = 2
fA(n) = n^2
log_bA(aA) = log_2(7)

Using property 1:

if f(n) = O(n^(log_b(a - e))), for {e > 0}
then T(n) = Theta(n^(log_b(a)))

n^(log_b(a)) = n^(log_2(7))
f(n) = n^2

= O(n^(log_2(7) - e)) (e = 0.81 approx)
T(n) = Theta(n^2.81)

f(n) is the same for noth algorithms, so we try the first property for TB(n)
We get
log_4(a) = log_2(7)
a = 49

Therefore, B is asymptotically faster than A when a < 49
So the largest a would be **48**