پاسخ سوال اول :

بله داریم شبه کد زیر:

FUNCTION Multiplication (x , y):

a) IF (x<10 or y<10) then Return x*y
b) END IF
c) SizeMax <- maximum length (x, y)
d) n <- ceil (SizeMax / 2)
e) a,b <- spilit (x) into two parts where each part length is n
f) c,d <- spilit (y) into two parts where each part length is n
g) ac <- Multiplication(a,c)
h) bd <- Multiplication(b,d)
i) adbc <- Multiplication(a+b,c+d) – ac –bd
j) returnvalue <- ac*(10^{2n})) + adbc*(10^{n}) + bd
k) Return returnvalue

شرح مختصری از شبه کد :

در شبه کد بالا تابع ما ابتدا دو عدد را میگیرید اگر یکی از دو عدد تک رقمی باشد در خط a جواب برگردانده میشود ولی اگر اینچنین نبود آنگاه در خط c میبینم اعداد چند رقمی هستند عددی که تعداد ارقام بیشتری دارد تعداد ارقامش را در SizeMax ذخیره میکنیم. حال در خط d این عدد را تقسیم میکنیم بر دو اگر عدد حاصل اعشاری باشد به سمت بالا گرد میکنیم و برابر n قرار میدهیم.

در خط e و f هردو عدد را به دو بخش تقسیم میکنیم هر بخش سایز nدارد به عنوان مثال اگر n =2 بود و عدد 2345 داریم:

a=23 و b = 45

همینطور c وd را به دست میاوریم

در خط های g,h,i تابع را دوباره بازخوانی میکنی و سپس در خط j نتایج را ادغام کرده و برمیگردانیم.

مشخص کردن بخش های تقسیم وغلبه:

تقسیم : در خطوط e تا i بخش تقسیم است که ابتدا اعداد را کوچک کردیم وسپس با فراخوانی تابع اعداد کوچک تر شده را به زیر مسعله کوچیکتر تبدیل کردیم که ممکن است این زیر مسعله به زیر مسعله کوچکتر تبدیل شود..

غلبه : در خط j نتایج حاصل از زیر مسعله ها را باهم ادغام میکنیم وباز میگردانیم.

پاسخ سوال دوم :

کوچکترین زیر مسعله در این روش حاصل ضرب دو عدد که یکی از دوعدد تک رقمی یا به عبارتی کوچکتر از ده میباشد.

پاسخ سوال سوم:

فایل پایتون که همرا با این PDF میباشد.

پاسخ سوال چهارم :

خب هر بار اجرای این الگوریتم 3 تا زیر مسعله با نصف شدن مقدار اصلی میباشد یا مسعله اول خودش کوچکترین مسعله است که انگاه پیچیدگی آن یک است پس داریم :

if n=1 $T(n) = 1$

if n >1 $T(n)=3T(n/2)$

$T(n/2)=3T(n/4)$

$T(n/4)=3T(n/8)$

$T(n/8)=3T(n/16)$

...

$T(n/k)=3T(n/2k)$

$T(n)= 3^kT(n/2^k)$

خب از انجا که هر بار n نصف میشود پس میتوان عمق را به صورت روبرو نوشت $k = \log_2 n$

پس داریم با توجه به نتیجه به دست اومده در بالا :

$n = 2^k \Rightarrow T(n)=3^kT(2^k/2^k)$, $3^{\log_2 n}=n^{\log_2 3} \Rightarrow T(n)= n^{\log_2 3} *T(1) \Rightarrow T(n)= n^{\log_2 3} = O(n^{1.58})$

پس میبینیم پیچیدگی زمانی کمتری را