۱- بله می‌توان با استفاده از روش تقسیم و غلبه الگوریتم بهینه تری را برای ضرب دو عدد صحیح پیاده سازی کرد. این الگوریتم با نام ضرب کرات (Karatsuba) شناخته می‌شود و با استفاده از روش تقسیم و غلبه زمان اجرای الگوریتم از $O(n^2)$ به $O(n^{\log_2 3})$ کاهش می‌یابد.

فرض می‌کنیم دو عدد صحیح a و b به شکل زیر باشند:

$$a = a_1 \times B^{\frac{n}{2}} + a_0$$

$$b = b_1 \times B^{\frac{n}{2}} + b_0$$

که B مبنای عدد و n تعداد ارقام آن می‌باشد.

سپس می‌توانیم ضرب a و b را به شکل زیر تقسیم کنیم:

$$a \times b = \left(a_1 \times B^{\frac{n}{2}} + a_0\right) \times \left(b_1 \times B^{\frac{n}{2}} + b_0\right) = a_1 b_1 B^n + (a_1 b_0 + a_0 b_1) \times B^{\frac{n}{2}} + a_0 b_0$$

که در اینجا مراحل تقسیم و ضرب را انجام داده‌ایم، اما به دلیل اینکه ضرب اعداد n رقمی به طور مستقیم سنگین است از روش تقسیم و غلبه استفاده می‌کنیم.

رابطه بالا را می‌توان به شکل زیر بازنویسی کرد که به ما اجازه می‌دهد به جای انجام سه ضرب با n رقم، چهار ضرب با $\frac{n}{2}$ رقم انجام دهیم.

$$a \times b = a_1 b_1 B^n + \left((a_1 + a_0) \times (b_1 + b_0) - a_1 b_1 - a_0 b_0\right) B^{\frac{n}{2}} + a_0 b_0$$

برای پیاده سازی این الگوریتم می‌توانیم از روش بازگشتی استفاده کنیم. در هر مرحله دو عدد a و b را به دو نیمه تقسیم کرده و ضرب هر دو نیمه را با روش بازگشتی انجام می‌دهیم.

شبه کد این الگوریتم به صورت زیر خواهد بود: (ورودی ها در مبنای B هستند.)

Algorithm

Input a, b

Output Product of a, b

begin

if n=1; return ab

a1=a>>n/2

a0=a mod $2^{\frac{n}{2}}$

b1=b>>n/2

b0=b mod $2^{\frac{n}{2}}$

a2=a1+a0

b2=b1+b0

R1=a1b1

R2=a0b0

R3=a2b2

return R1$\times B^n + (R3 - R1 - R2) \times B^{\frac{n}{2}} + R2$

در این جا هر عدد به دو عدد $\frac{n}{2}$ رقمی تقسیم شده و برای هر مرحله این روش به صورت بازگشتی انجام می‌شود تا به ضرب دو عدد یک رقمی برسیم.

۲- کوچکترین زیرمسئله در این روش ضرب دو رقمی است. به عنوان مثال در ضرب دو عدد n رقمی، هر عدد را می‌توان به دو عدد $\frac{n}{2}$ رقمی تقسیم کرد و این کار تا رسیدن به ضرب دو عدد یک رقمی ادامه داد. این کار را با الگوریتم تقسیم و غلبه می‌توان انجام داد که همان طور که بیان شد پیچیدگی زمانی بهتری را خواهد داشت.

۴- الگوریتم ضرب طولانی دو عدد در هم که در پرسش ارائه شده است، به صورت خطی با رشته هر دو عدد کار می‌کند و در هر مرحله یک ضرب را انجام می‌دهد. این امر باعث می‌شود پیچیدگی زمانی این الگوریتم به طول رشته هر دو عدد وابسته باشد. برای محاسبه پیچیدگی می‌توانیم از نماد O برای نشان دادن حد بالای تعداد عملیات مورد نیاز در الگوریتم استفاده کنیم. با فرض این که دو عدد دارای n رقم باشند، در هر مرحله n ضرب و یک جمع انجام می‌شود. بدین شکل تعداد کل ضرب ها n^2 و تعداد کل جمع ها برابر n خواهد بود و در نتیجه پیچیدگی الگوریتم ضرب طولانی برابر $O(n^2)$ خواهد بود.

پیچیدگی الگوریتم به میزان منابع مورد نیاز برای اجرای آن الگوریتم وابسته است. این منابع می‌تواند شامل زمان و حافظه مورد نیاز برای اجرای الگوریتم باشد. برای مثال پیچیدگی زمانی الگوریتم، تعداد مراحل و عملیات های مورد نیاز برای اجرای الگوریتم است. برای الگوریتم ضرب طولانی دو عدد همان طور که گفته شد پیچیدگی زمانی‌ای متناسب با مربع تعداد ارقام اعداد ورودی دارد و برای ضرب اعداد با تعداد ارقام بسیار بالا مثلا بیش از هزار رقم، بسیار کند خواهد بود. برای مقایسه الگوریتم کرات دارای پیچیدگی زمانی $O(n^{\log_2 3})$ است که برابر $O(n^{1.585})$ خواهد بود که در ضرب اعداد با تعداد ارقام بسیار زیاد بسیار بهینه تر از روش ضرب طولانی خواهد بود. در نهایت برای مقایسه پیچیدگی الگوریتم ها می‌توان به موارد دیگری نظیر پیچیدگی حافظه یا تعداد دستورات مورد نیاز در اجرای الگوریتم ها نیز توجه کرد.