به جای عبارت expression باید a باشد. طبق قضیه اساسی حساب هر عدد را می‌توان به صورت نمایش از مجموع توان‌های عدد ۲ نوشت. در این الگوریتم نیز ما در واقع توان را با محاسبه نمایش باینری آن می‌سازیم، عمل‌های ضرب صرفا برای ساختن توان هستند. برای مثال: x^{11} و x^{30}:

$(30)_2 = 11110$ $\quad (11)_2 = 1011$

هنگامی که تقسیم بر 2 می‌کنیم اگر باقیمانده‌ها را از راست به چپ بنویسیم نمایش باینری عدد می‌شود.

$30 \mid 2$
$30 \quad 15 \mid 2$
$0 \quad 14 \quad 7 \mid 2$
$\quad 1 \quad 6 \quad 3 \mid 2$
$\quad \quad 1 \quad 2 \mid 2$
$\quad \quad \quad 1$

$\leftarrow 11110$

در الگوریتم نیز به این صورت است که در هر بار که حلقه اجرا می‌شود توان ۲ برابر می‌شود و اگر بیت i ام (از راست) که بیانگر حلقه i ام است ۱ باشد، یعنی در تقسیم فرد بوده است و در این صورت آن مقدار توان قبلی در a ذخیره می‌شود در غیر این صورت توان ۲ برابر می‌شود و ۱ بیت بعدی را رویم.

هنگامی که i مرحله اجرا می‌شود توان را به صورت $2^{i-1}d_i + \cdots + 2^1 d_2 + 2^0 d_1$ ساخته‌ایم که نمایش باینری توان است. چون تعداد بیت‌های k برابر $\log k$ است بنابراین $\Theta(\log k)$ زمان می‌برد که عدد x^k را بسازیم.

در هر مرحله حلقه هم حداکثر ۳ عملیات ضرب و تقسیم انجام می‌شود که $\Theta(3\log k)$ می‌شود $\Theta(\log k)$.

- Initialization: قبل از شروع حلقه هنوز عملیاتی انجام نداده‌ایم، $y=x$ ، $i=k$ ، $a=1$ است که مقدار 2^0 را نشان می‌دهد.
- Maintenance: در مرحله i ام که حلقه اجرا می‌شود یک نمایش باینری به صورت $2^{i-1}d_i + 2^{i-2}d_{i-1} + \cdots + 2^0 d_1$ از راست حساب شده که d_i تا d_1 بیت اول عدد k از راست در نمایش باینری هستند چون در هر مرحله که k را تقسیم بر ۲ می‌کنیم یک بیت از سمت راست آن را حساب کرده‌ایم. در این مرحله 2^{i-1} در توان y ذخیره شده و در a مقدار $x^{2^{i-1}d_i + \cdots + 2^0 d_1}$ ذخیره شده است.
- Termination: در این مرحله با توجه به اینکه i مرحله از بیت‌های سمت راست توان محاسبه شده است و این بار که تقسیم می‌کنیم بیت $i+1$ ام مشخص می‌شود و اگر ۱ باشد توان a را با توان y جمع می‌کنیم تا مرحله‌ای که از صفر شود در این مرحله همه‌ی بیت‌های k محاسبه شده و در a ذخیره شده است و در آخر a را به عنوان expression باز می‌گردانیم.