```
fastExponent(x, n)
1  r = 1
2  p = x
3  e = n
4  while e > 0
5      if e mod 2 <> 0
6          r = r * p
7      p = p * p
8      e = e div 2
```

Т.к. в данном алгоритме используются операции целочисленного деления и нахождения остатка в отношении переменных $e = n$, то $n = e \in \mathbb{Z}$.

Т.к. при $e = n < 0$ алгоритм всегда получит $r = x^n = 1$, то подразумевается, что $e = n \geq 0$, т.е. $e = n \in \mathbb{N} \cup \{0\}$ (*)

1) При каждом выполнении тела цикла в первых 3 строках (в коде - строки с 5 по 7) выполняются инструкции, не изменяющие значение e, если его переменная e делится нацело на 2.

Т.к. $e \in \mathbb{N} \cup \{0\}$, то корректно рассмотреть представление e в двоичной системе счисления. Из (*) $\Rightarrow ((e > 0) \Leftrightarrow (e \neq 0))$.

На каждом шаге работы цикла тело цикла выполняется, если $e \neq 0$, т.е. в двоичном представлении есть хотя бы одна единица (хотя бы 1 разряд $\neq 0$)

Из (*) $\Rightarrow$ деление на 2 равносильно побитовому (поразрядному) сдвигу двоичной записи вправо на 1 бит (разряд)

Таким образом, если $c \neq 0$ изначально, т.е. $n \neq 0$, то тело цикла выполнится столько раз, сколько в числе n разрядов до последней слева единицы $+1$, иначе если $n = 0$, то тело цикла цикла выполнится 0 раз. Тогда количество раз, которое выполнится тело цикла, равно $f(n) = \begin{cases} \lfloor \log_2 n \rfloor + 1, & n \neq 0 \\ 0, & \text{иначе} \end{cases}$

Докажем что эту функцию можно задать в явном виде: $\lceil \log_2 (n+1) \rceil$.

□ Рассмотрим $n \in \mathbb{N} \cup \{0\}$

1) $n = 0$ $\begin{cases} f(n) = 0 \\ \lceil \log_2 (n+1) \rceil = \lceil \log_2 (1) \rceil = 0 \end{cases} \Rightarrow f(n) = \lceil \log_2 (n+1) \rceil$

2) $n \neq 0$ $(\Leftrightarrow n > 0)$

а) $\exists k \in \mathbb{N} \cup \{0\} \Rightarrow n = 2^k$ (т.е. n – степень двойки)

$$\begin{cases} f(n) = \lfloor \log_2(n) \rfloor + 1 = \lfloor k \rfloor + 1 = k+1 \\ \lceil \log_2(n+1) \rceil = \lceil \log_2(2^k + 1) \rceil = k+1 \end{cases} \Rightarrow f(n) = \lceil \log_2(n+1) \rceil$$

б) $\exists k \in \mathbb{N} \cup \{0\} \Rightarrow 2^k < n < 2^{k+1}$

(т.е. n – не степень двойки $\Rightarrow$ n зажата между двумя соседними степенями, т.к. $n > 0$)

$$\begin{cases} f(n) = \lfloor \log_2(n) \rfloor + 1 = k+1 \\ \lceil \log_2(n+1) \rceil = k+1 \end{cases} \Rightarrow f(n) = \lceil \log_2(n+1) \rceil$$

■

При каждом выполнении тела цикла r умножается сама на себя. Также, если $e \bmod 2 \neq 0$ ($\Leftrightarrow e \equiv 1 \pmod 2$), то r умножается на p (строка 6). Это равносильно тому, что последний бит (разряд) числа e в двоичной записи равен 1. Тогда, т.к. на каждой итерации цикла двоичная запись числа e сдвигается на 1 бит (разряд) вправо (при этом проверка последнего бита и возможное умножение происходят до сдвига), то

умножение x на p (строка 6) происходит столько раз, сколько единиц в двоичной записи числа n.

Введём функцию $popcnt : \mathbb{N} \cup \{0\} \to \mathbb{N} \cup \{0\}$, такую, что $popcnt : n \mapsto$ # единиц в двоичной записи числа n.

("кол-во")

Формально её можно определить так:

$$\forall n \in \mathbb{N} \cup \{0\} \Rightarrow popcnt(n) = \sum_{k=0}^{+\infty} \left(\left\lfloor \frac{n}{2^k} \right\rfloor \bmod 2 \right)$$

↑ побитовый сдвиг на k разрядов вправо двоичной записи n

↑ значение младшего разряда в двоичной записи n.

Т.к. $\forall k > \lfloor \log_2 n \rfloor \Rightarrow \left\lfloor \frac{n}{2^k} \right\rfloor = 0$, то

$$popcnt(n) = \sum_{k=0}^{+\infty} \left(\left\lfloor \frac{n}{2^k} \right\rfloor \bmod 2 \right) = \sum_{k=0}^{\lfloor \log_2 n \rfloor} \left(\left\lfloor \frac{n}{2^k} \right\rfloor \bmod 2 \right)$$

Итого при выполнении алгоритма будет выполнено $\lceil \log_2(n+1) \rceil + \sum_{k=0}^{\lfloor \log_2 n \rfloor} \left(\left\lfloor \frac{n}{2^k} \right\rfloor \bmod 2 \right)$ операций умножения.

При $n = 3$ при выполнении данного алгоритма будет выполнено

$\lceil \log_2(3+1) \rceil + \sum_{k=0}^{\lfloor \log_2 3 \rfloor} \left(\left\lfloor \frac{3}{2^k} \right\rfloor \bmod 2 \right) = 2 + 2 = 4$ операции умножения.

Наивный алгоритм возведет x в $n = 3$ за 2 операции умножения $x^n = x^3 = x \cdot x \cdot x$

$4 > 3 \Rightarrow$ данный алгоритм не всегда лучше наивного способа вычисления.

Ответ: $T(n) = \lceil \log_2(n+1) \rceil + \sum_{k=0}^{\lfloor \log_2 n \rfloor} \left(\left\lfloor \frac{n}{2^k} \right\rfloor \bmod 2 \right)$; данный алгоритм не всегда лучше наивного способа вычисления.

2. При инициализации (INIT)

$$r = 1, p = x, e = n \implies x^n = rp^e$$

При выходе из цикла (EXIT)

$$e = 0, r = x^n \implies x^n = rp^e$$

Пусть инвариант P цикла while: на каждой итерации $x^n = rp^e$

На этапе выхода (EXIT) и инициализации (INIT) выполнение P проверено.

При каждом выполнении тела цикла r умножается на p, если $e \equiv 1 (\mathrm{mod}\ 2)$, после этого p умножается на p, а e делится нацело пополам, т.е. выражение rp^e перейдет в $(rp^{(e \bmod 2)})(p^2)^{\lfloor \frac{e}{2} \rfloor}$

Докажем, что они равны:

Если $e \equiv 0 (\mathrm{mod}\ 2)$, то $(rp^{(e \bmod 2)})(p^2)^{\lfloor \frac{e}{2} \rfloor} =$

$= rp^0 * (p^2)^{\frac{e}{2}} = rp^e$

Иначе, если $e \equiv 1 (\mathrm{mod}\ 2)$, $(rp^{(e \bmod 2)})(p^2)^{\lfloor \frac{e}{2} \rfloor} =$

$= rp^1 * (p^2)^{\frac{e-1}{2}} = rp^e$

$\Rightarrow$ При каждом выполнении тела цикла (CONT) инвариант $P : x^n = rp^e$ выполняется.

Ответ: инвариант P: на каждой итерации $x^n = rp^e$