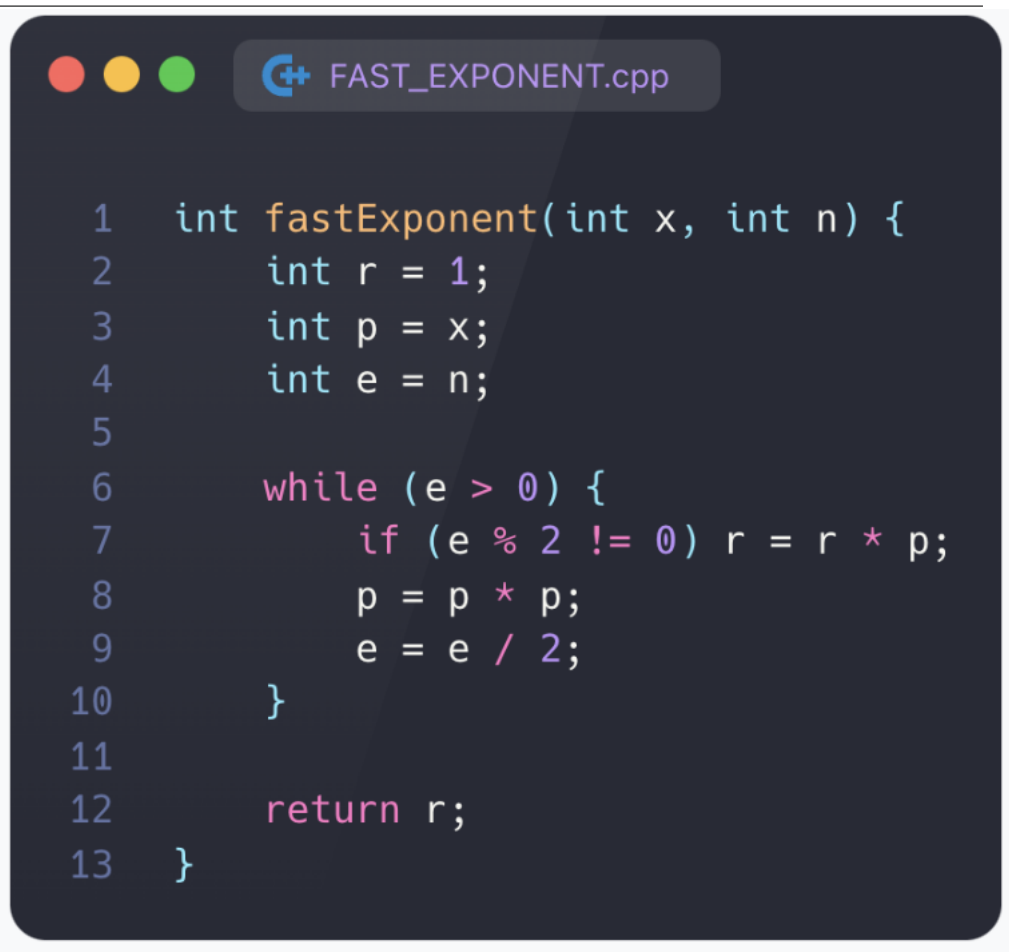


АиСД | SET-1 | А2

Потякин Арсений

TODO:

1. Какое точное количество операций умножения требуется выполнить, чтобы вычислить x^n с помощью алгоритма FAST EXPONENT? Всегда ли данный алгоритм лучше наивного способа вычисления?
2. Сформулируйте условие P, которое подходит в качестве инварианта цикла while. Представьте достаточное обоснование выбора инварианта. Выполните проверку выполнения найденного инварианта до входа в цикл (INIT), в каждой итерации цикла (MNT), а также при выходе из цикла (TRM).



```
1  int fastExponent(int x, int n) {
2      int r = 1;
3      int p = x;
4      int e = n;
5
6      while (e > 0) {
7          if (e % 2 != 0) r = r * p;
8          p = p * p;
9          e = e / 2;
10     }
11
12     return r;
13 }
```

Task 1: (Количество операций)

Умножение p на p выполняется $\lfloor \log_2 n \rfloor + 1$ раз, а умножение r на p происходит всякий раз, когда e нечетно. Количество таких умножений равно числу единичных битов в двоичной записи n , то есть весу Хэмминга. Таким образом, потребуется $\lfloor \log_2 n \rfloor + 1 + \text{hammingWeight}(n)$ операций

Алгоритм быстрой экспоненты работает за $T_{fast}(n) \approx 2 \log_2 n$, а наивный способ за $T_{naive}(n) = n - 1$. Найдем такое n , что $2 \log_2 n < n - 1 \implies 2 \log_2 n < n$ (для больших n) $\implies \log_2 n < \frac{n}{2}$. Из этого можно сделать вывод, что алгоритм быстрой сортировки более эффективен при $n \geq 5$

Task 2.1: (Инвариант и обоснование)

Предположим, что $x^n = r \cdot p^e$, тогда инвариантом цикла `while` можно представить $r \cdot p^e$: r - результат накопленных умножений, p - текущая степень основания, e - оставшаяся экспонента.

Обоснование: На каждой итерации цикла алгоритм уменьшает значение e , но сохраняет промежуточные результаты. \forall итерации справедливо, что текущее значение $r \cdot p^e$ равно исходному x^n , где e уменьшается с каждым шагом вдвое.

Task 2.2: (Проверка)

INIT: До входа в цикл справедливо, что $e = n, p = x, r = 1 \implies r \cdot p^e = 1 \cdot x^n = x^n$

MNT: На каждой итерации справедливо, что: если e нечетное, то $r = r \cdot p$, $p = p \cdot p$, $e = \frac{e}{2}$, при этом инвариант сохраняется, т.к. p^e обновляется согласно новому e ; r корректно обновляется умножением на p , которое учло нечетность экспоненты. $\implies r \cdot p^e = x^n$

TRM: После завершения цикла $e = 0$. Проверим инвариант. $r \cdot p^0 = r = x^n \implies r = x^n$, что и являлось целью алгоритма.

\implies инвариант выполняется