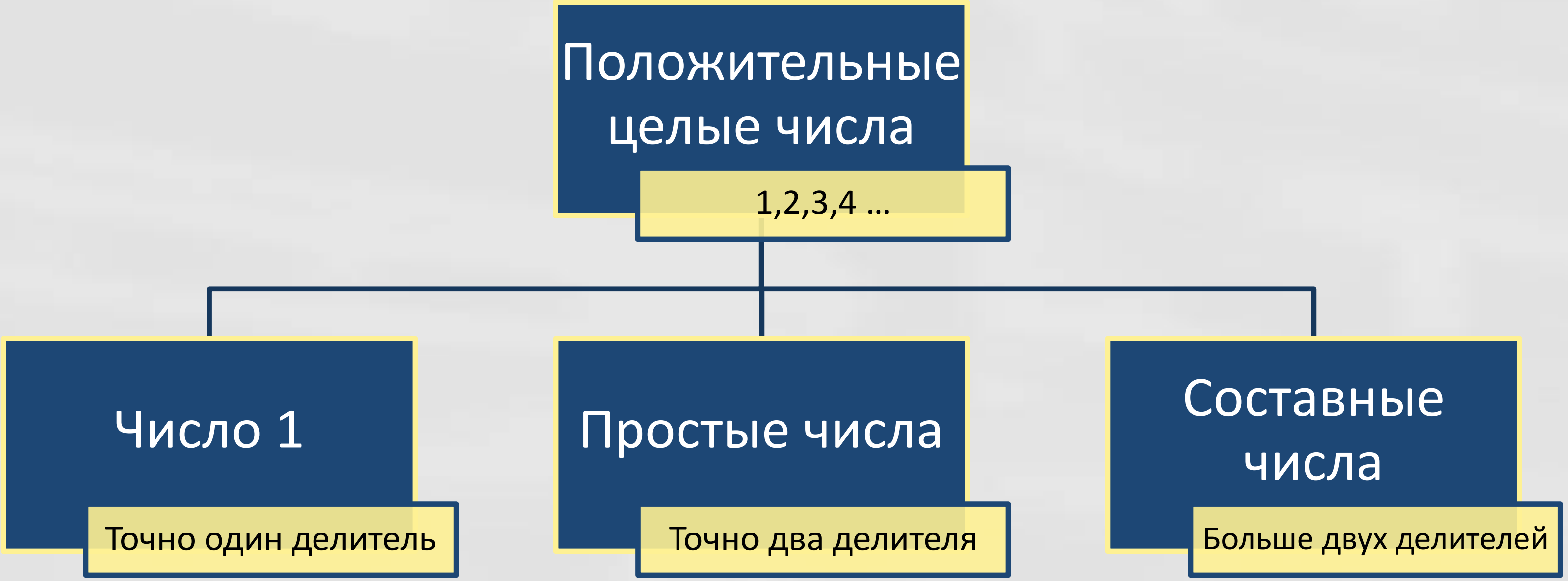
Определения



- Положительное целое число является простым *(prime)* числом тогда и только тогда, когда оно точно делимо без остатка на два целых числа — на 1 и на само себя
- Составное число — это положительное целое число больше с чем двумя делителями
- Два положительных целых числа a и b являются взаимно простыми *(coprime),* если $НОД\ (a, b) = 1$
- Если p — простое число, тогда все числа от 1 до $p–1$ являются взаимно простыми к p

Количество простых чисел

- Количество простых чисел бесконечно. Доказательство:
  - Пусть *p* –наибольшее простое. Вычислим $P = 2 \times 3 \times 5 \times \ldots \times p$.
  - Пусть *(P+1)* имеет простой делитель *q>1*. Если $q \leq p$, то *q* делит и *P*
  - Тогда *q* делит и разность *(P+1)-P=1*. Получается, что *q=1*, что противоречит условию *q>1.* Поэтому простое число $q > p$.
- Количество $\pi(n)$ простых чисел, меньших *n*
  - Нижний предел обнаружил Лагранж $[n/(\ln n)] < \pi(n)$
  - Верхний предел обнаружил Гаусс $\pi(n)$<[n/(ln n- 1,08366)]

Полезные свойства простых чисел

- Если число x не делится ни на одно из простых чисел, лежащих в диапазоне от 2 до x-1, то x-простое число
- Если число x не делится ни на одно из простых чисел, лежащих в диапазоне от 2 до p, где $p = [\sqrt{x}]$, то x-простое число:
  - Доказательство: Пусть $x = a \times b, a > p, b > p$, $p = [\sqrt{x}]$
    Тогда $a \times b \geq (p + 1) \times (p + 1) > x$ имеем противоречие

Малая теорема Ферма

- Первая версия
  - Если p — простое число и a — целое число, такое, что p не является делителем a, тогда $a^{p-1} \equiv 1 \; mod \; p$
    - Следствие $a^{-1} \equiv a^{p-2} mod \; p$
- Вторая версия
  - Если p - простое число и a - целое число, то $a^{p} \equiv a \; mod \; p$
- Приложения:
  - Возведение в степень: $3^{100} mod \; 97 = 81$ $\quad 99^{73} \; mod \; 73 = 26$
  - Мультипликативная инверсия: $5^{-1} mod \; 11 = 9$

Функция Эйлера

- Функция $\varphi(n)$ вычисляет количество целых чисел меньших, чем *n*, и взаимно простых с *n*. Пример $\varphi(10)=4$, {1,3,7,9}
- Свойства функции:
  - $\varphi(1) = 0$
  - $\varphi(p) = p - 1$, если p – простое
  - $\varphi(m \times n) = \varphi(m) \times \varphi(n)$, если n и m – взаимно простые
  - $\varphi(p^e) = p^e - p^{e-1}$, если p- простое
- Сложность нахождения $\varphi(n)$ зависит от сложности нахождения разложения *n* на множители

Теорема Эйлера

- Первая версия (подобна первой версии малой теоремы Ферма)
  - Если a и n – взаимно простые, то $a^{\varphi(n)} \equiv 1 \; mod \; n$
    - Следствие: $a^{-1} mod \; n = a^{\varphi(n)-1} mod \; n$
- Вторая версия (подобна второй версии малой теоремы Ферма)
  - Если $n = p \times q, a < n, \quad k$ – целое число, то $a^{k \times \varphi(n)+1} \equiv a \; mod \; n$
- Приложения:
  - Возведение в степень: $6^{24} mod \; 35 =$ **1** $\quad 20^{62} mod \; 77 =$ **15**
  - Мультипликативная инверсия: $7^{-1} mod \; 15 =$ **13**

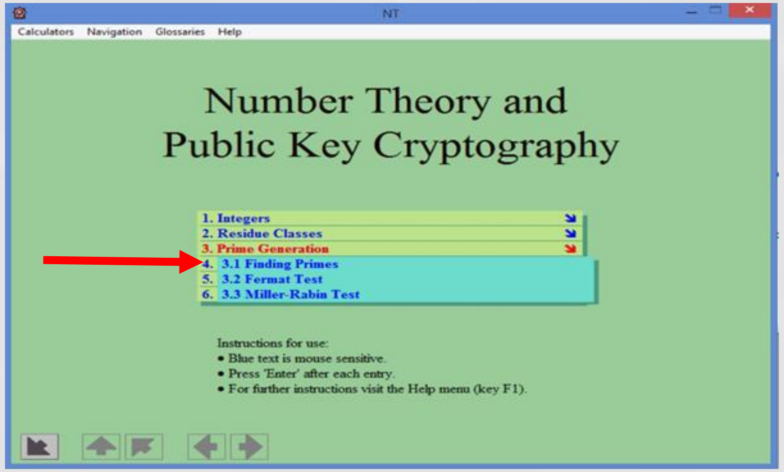
Фильтры и генераторы простых целых чисел

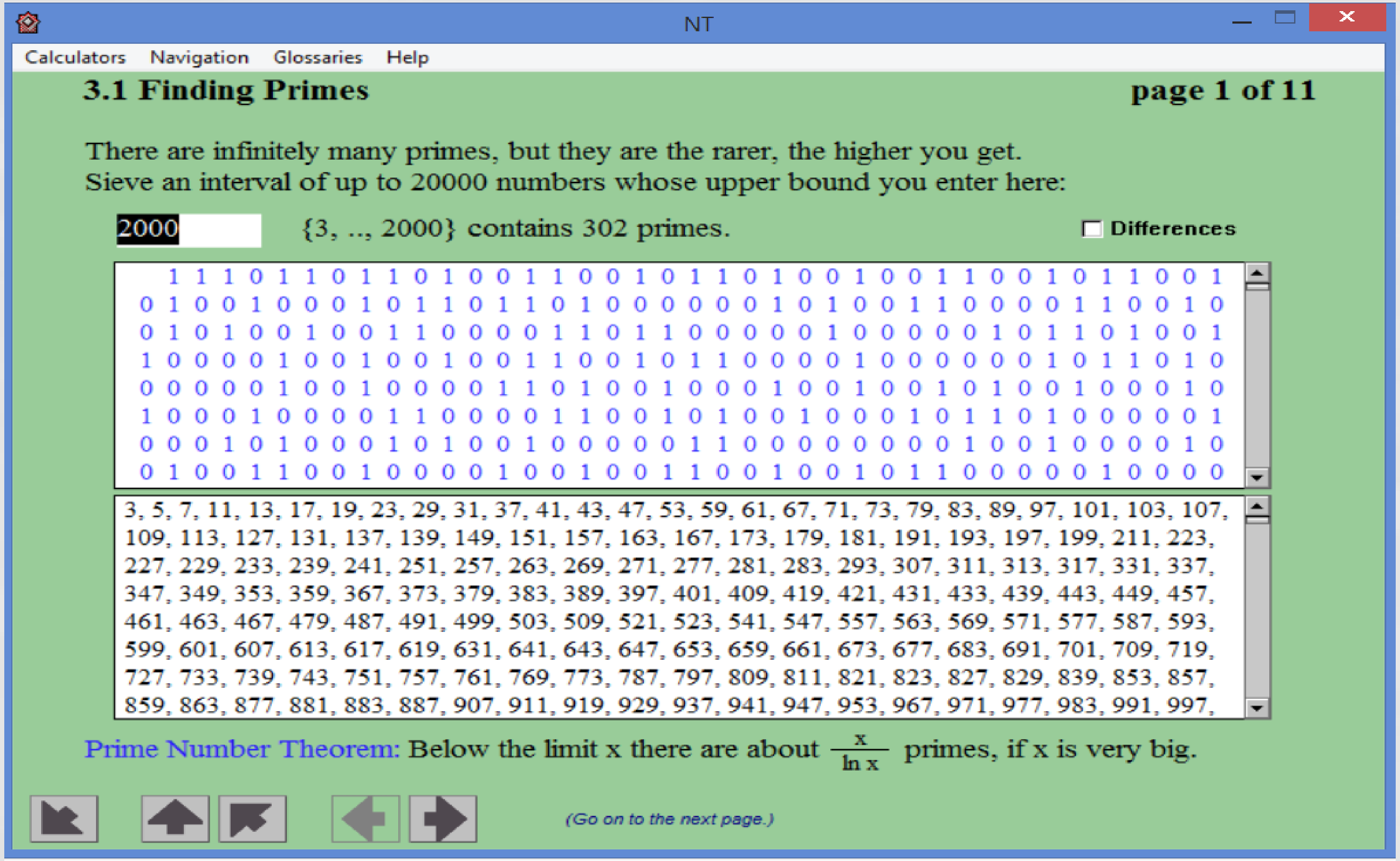
Фильтрация простых чисел меньше заданного *n*

- Решето Эратосфена
  - Пусть *n=100*. Найдем все простые числа меньше $[\sqrt{100}]$. Это числа *2,3,5,7*.
  - Впишем все числа от *2* до *100* в таблицу и будем последовательно вычеркивать числа делящиеся на *2*, затем на 3,5,7 (кроме самих этих чисел
  - Оставшиеся числа – простые (в выделенных клетках)

	2	**3**	4	**5**	6	**7**	8	9	10
11	12	**13**	14	15	16	**17**	18	**19**	20
21	22	**23**	24	25	26	27	28	**29**	30
31	32	33	34	35	36	**37**	38	39	40
41	42	**43**	44	45	46	**47**	48	49	50
51	52	**53**	54	55	56	57	58	**59**	60
61	62	63	64	65	66	**67**	68	69	70
71	72	**73**	74	75	76	77	78	**79**	80
81	82	**83**	84	85	86	87	88	**89**	90
91	92	93	94	95	96	**97**	98	99	100

Фильтр в CrypTool





Пример реализации фильтра:

Вход: натуральное число n

Пусть A — булевый массив, индексируемый числами от 2 до n, изначально заполненный значениями **true**.

для $i := 2, 3, 4, \ldots,$ **пока** $i^2 \leq n$:
  если $A[i]$ = **true**:
    для $j := i^2, i^2 + i, i^2 + 2i, \ldots,$ **пока** $j \leq n$:
      $A[j]$:= **false**

Выход: числа i, для которых $A[i]$ = **true**.

- Сложность $O(n * \ ln(ln\, n))$
- Можно сократить вдвое число операций, если оперировать только нечётными числами и
- Можно существенно сэкономить потребление памяти, храня n переменных булевского типа не как n байт, а как n бит
- Детали: https://habrahabr.ru/post/91112/

Попытки генерация простых целых чисел

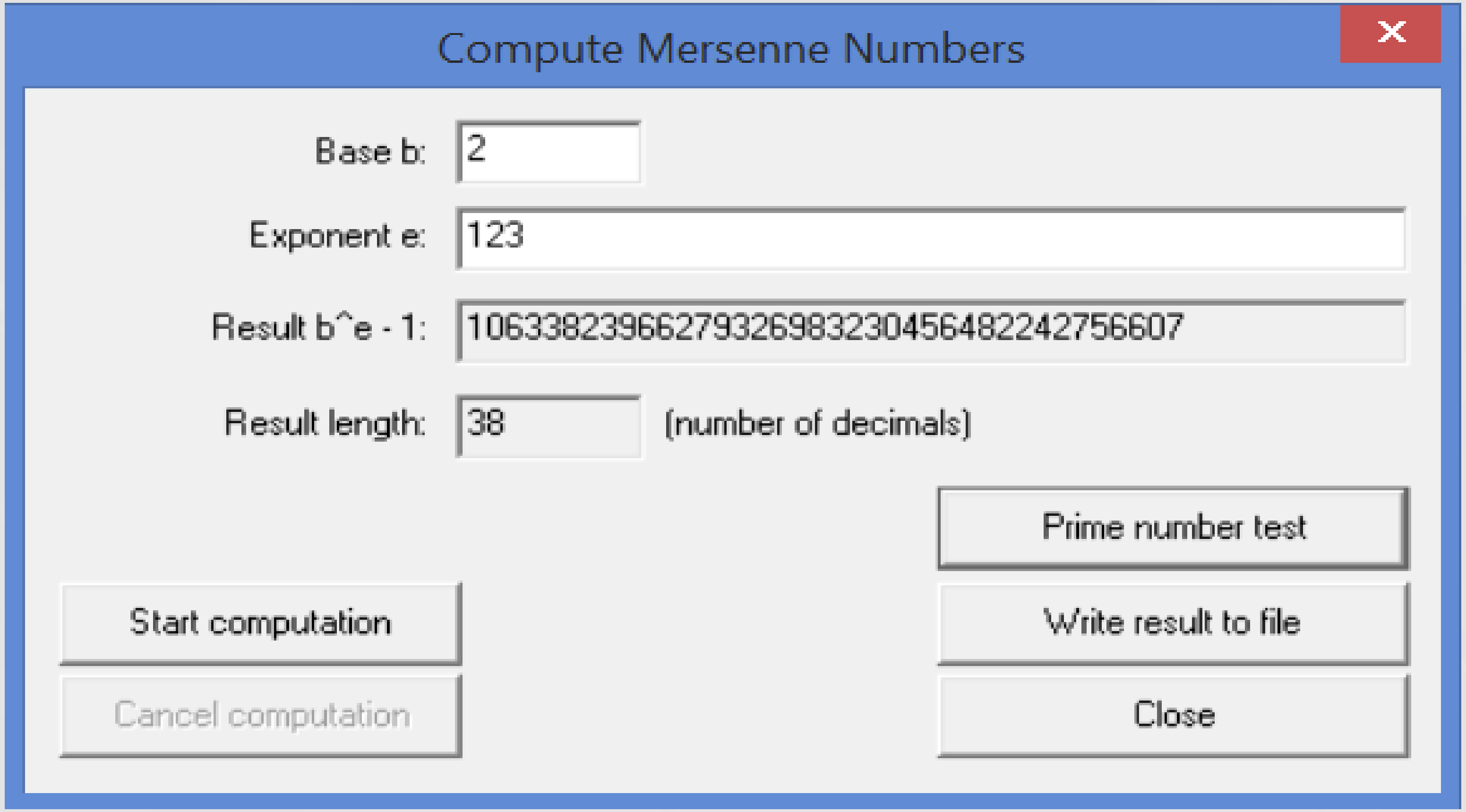
- **Простые числа Ферма**
  - Предполагал, что формула для генерации простого числа имеет вид $F_n = 2^{2^n} + 1$
  - Подобные числа представляются битовой строкой вида   100000…001
  - Доказано, что если $2^k + 1$ простое число, то k является степенью 2
  - Доказано, что многие номера после F_4 являются составными. Например,

    $F_5 = 4294967297 = 641 \times 6700417$

Попытки генерация простых целых чисел

- **Простые числа Мерсенна**
  - Предполагал, что для простого числа p может вычислено другое простое M_pчисло (номер Мерсенна) по формуле $M_p = 2^p - 1$
  - Номер Мерсенна представляется битовой строкой вида 111111…111
  - Доказано, что если $M_p = 2^p - 1$ простое число, то p тоже простое число
  - Обратное утверждение неверно, например, $M_{11} = 2^{11} - 1 = 2047 = 23 \times 89$, т.е. не все числа полученные по этой формуле, являются простыми
  - История поиска: https://habr.com/ru/post/409503/#:~:text=Простое%20число%20Мерсенна%20—%20это%20простое,известно%2050%20простых%20чисел%20Мерсенна.
  - 7 декабря 2018 года было открыто наибольшее известное простое число, которое равняется $2^{82\,589\,933} - 1$ и содержит 24 862 048 десятичных цифр.

Генератор в Cryptool



Тестирование на простоту

Вероятностные тесты

- Вероятностный алгоритм правильно выявляет простое число в большинстве (но не во всех) случаях. Вероятность ошибки (назвать составное простым) настолько маленькая, что это почти гарантирует, что алгоритм вырабатывает правильный ответ

Детерминированные тесты

- Детерминированный алгоритм принимает целое число и выдает на выходе признак: это число — простое число или составное. Детерминированный алгоритм всегда дает правильный ответ.

Вероятностные тесты на простоту

Идея вероятностных тестов на простоту

- В основе вероятностного алгоритма лежит математически доказанное свойство простого числа
- Проверяется выполнение этого свойства , как необходимого условия «простоты» числа: ***ЕСЛИ*** *число простое* ***ТО*** *свойство*
- Если проверяемое целое число фактически является простым число, алгоритм объявит его простым
- Если проверяемое целое число фактически является составным , алгоритм объявляет его составным с вероятностью $1 - \varepsilon$, но может объявить простым числом с ε вероятностью.
- Вероятность ошибки может быть улучшена, если проверять необходимое условие несколько раз с различными параметрами или с использованием различных методов

Вероятностный тест испытания кв. корнем

- Необходимое условие простоты числа: Если n-простое число то уравнение $x^2 \equiv 1 \bmod n$ имеет только два корня *1 и (n-1)*
- Если n составное число, то кроме указанных значений могут быть и еще другие
  - Пример: Каковы корни $x^2 \equiv 1 \bmod n$ при *n=7*: **{1,6}**
  - Пример: Каковы корни $x^2 \equiv 1 \bmod n$ при *n=8*: **{1,3,5,7}**
  - Пример: Каковы корни $x^2 \equiv 1 \bmod n$ при *n=22*: **{1,21}**
- Когда дано число *n*, то все числа, меньшие, чем *n* (кроме чисел *1* и *n–1*), должны быть возведены в квадрат по модулю , чтобы гарантировать, что ни одно из них не дает решения равного 1

Вероятностный тест Ферма

- Необходимое условие простоты числа: если p-простое число, то $a^{p-1} \equiv 1 \; mod \; p$
- Если p – составное, то вышеуказанный тест может быть пройден с ошибкой: 2^{340}=$1 \; mod \; 341,$ но $341 = 31 \times 11$
- Вероятность может быть улучшена, если проверка делается с несколькими числами $a_i, i = 1{,}100$.
- Сложность разрядной операции теста Ферма равна сложности алгоритма быстрого возведения в степень $O(p_b)$

Вероятностный тест Миллера-Рабина

- Является комбинацией тестов Ферма и квадратного корня
- Изначально алгоритм был разработан Гари Миллером в 1976 году, а Майкл Рабин модифицировал его в 1980 году
- В основе используется доказанное утверждение:

Пусть $p > 2$ — простое число. Представим число $p - 1$ в виде $p - 1 = 2^s d$, где d — нечётно. Тогда для любого a из $\mathbb{Z}_p$ выполняется одно из условий:

1. $a^d \equiv 1 \pmod p$
2. $\exists r, 0 \le r \le s-1 : \ a^{2^r d} \equiv -1 \pmod p$

- Если это утверждение выполняется для некоторых чисел a и p, то число a называют свидетелем простоты числа p, а само число p — вероятно простым

Псевдокод теста Миллера-Рабина

Вход: $p > 2$, нечётное натуральное число, которое необходимо проверить на простоту; k — количество раундов.

Выход: ***составное***, означает, что p является составным числом;

вероятно простое, означает, что p с высокой вероятностью простое

Представить $p - 1$ в виде $2^s \cdot d$, где d нечётно

ЦИКЛ А: повторить k раз:

Выбрать случайное целое число a в отрезке $[2, p - 2]$

$x \leftarrow a^d \bmod p$

ЕСЛИ $x = 1$ или $x = p - 1$, то перейти на следующую итерацию цикла А

ЦИКЛ В: повторить $s - 1$ раз

$x \leftarrow x^2 \bmod p$

если $x = 1$, то вернуть ***составное***

если $x = p - 1$, то перейти на следующую итерацию цикла А

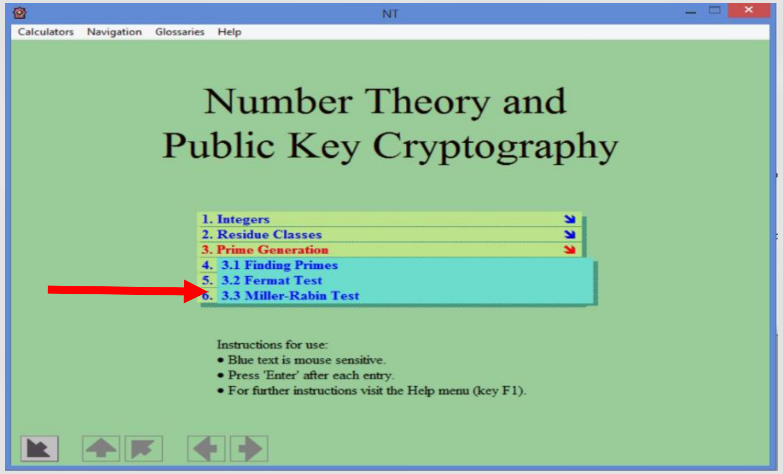
КОНЕЦ ЦИКЛА В

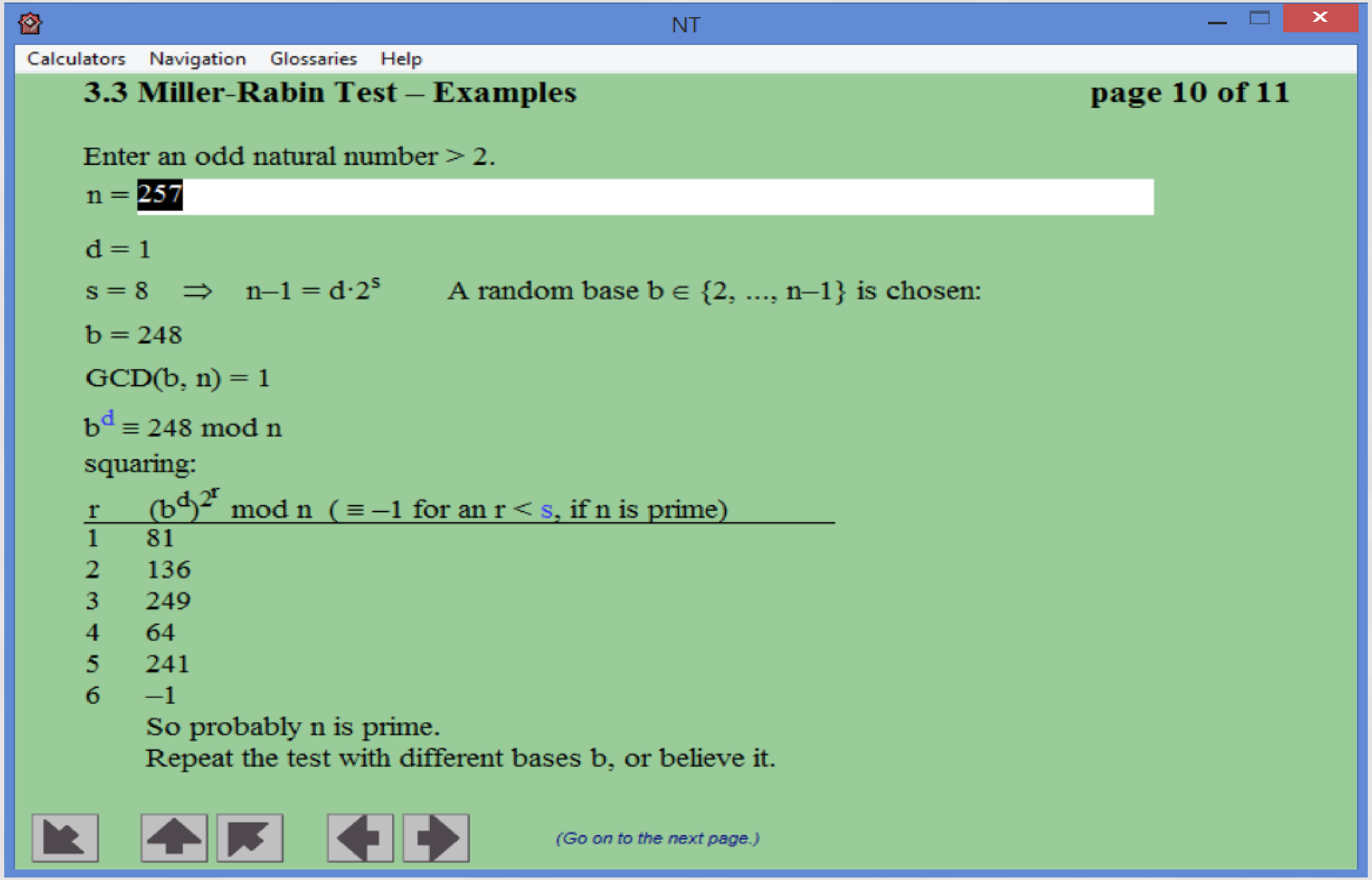
вернуть ***составное***

КОНЕЦ ЦИКЛА А

вернуть ***вероятно простое***

Тест Миллера-Рабина в CrypTool





Свойства теста

- Идея теста заключается в том, чтобы проверять для случайно выбранных чисел $a<p$, являются ли они свидетелями простоты числа p.
- Если на определённом шаге алгоритма было проверено k чисел, и все они оказались свидетелями простоты, то вероятность того, что число p составное не более $(1/4)^k$ (это доказано)
- Время работы алгоритма полиномиально $O(k \times {log_2}^2 n)$

Рекомендованные тесты

- Сегодня один из самых популярных тестов простоты чисел — комбинация теории делимости и теста Миллера-Рабина. При этом рекомендуются следующее шаги:
  - Выбрать нечетное целое число
  - Сделать некоторые тривиальные испытания теории делимости на некоторых известных простых числах, таких как 3, 5, 7, 11, 13. Если они не являются делителями выбранного числа, перейти к следующему шагу, иначе выбрать другое нечетное число.
  - Выбрать набор оснований для теста. Большое множество оснований предпочтительно.
  - Сделать тест Миллера-Рабина на каждом из оснований. Если одно из них не проходит, выбрать другое нечетное число. Если тесты прошли для всех оснований, объявите выбранное число, как сильное псевдопростое число.

Детерминированные тесты на простоту

Идея детерминированных тестов на простоту

- В основе детерминированного алгоритма тоже лежит математически доказанное свойство простого числа
- Проверяется выполнение этого свойства , как достаточного условия «простоты» числа: ***ЕСЛИ*** *свойство* ***ТО*** *число простое*
- Если проверяемое целое число фактически является простым число, алгоритм объявит его простым
- Если проверяемое целое число фактически является составным , алгоритм объявляет его составным

Детерминированный алгоритм пробного деления

- Используем в качестве делителей все числа, меньшие, чем $\sqrt{n}$.
- Если любое из этих чисел делит n, тогда n — составное
- Алгоритм может быть улучшен, если проверять только нечетные номера и использовать таблицу простых чисел от 2 до $\sqrt{n}$
- Сложность разрядной операции алгоритма $2^{n_b/2}$, где n_b число битов в n

```
Тест на делимость (n)

{
  r ← 2
  while (r < √n)
    {
    if (r|n) return "a composite"// составное
    r ← r + 1
   }
  return "a prime"//простое
}
```

Алгоритм на основе подбора разложения

- <u>Теорема:</u> Пусть *p* – целое нечетное число, *p>1*. Число *p* является простым, если существует $b \leq p-1,$ такое, что :
  - $b^{p-1} = 1\, mod\, p$
  - $b^{p-1/m_i} \neq 1 mod\, p$ для каждого простого делителя m_i числа *p-1*
- <u>Алгоритм:</u>
  - Случайным образом выбираются простые числа $\{m_1, m_2, \ldots m_q\}$
  - Вычисляем $p = 1 + 2\prod_{i=1}^{q} m_i$
  - Выбирается $b \leq p-1$ и проверяются условия Теоремы. Если есть такое b, то p найдено, если нет, то выбирается новые $\{m_1, m_2, \ldots m_q\}$
- <u>Свойства решения:</u>
  - Длина p примерно в q раз больше средней длины $\{m_1, m_2, \ldots m_q\}$
  - Мы заранее знаем разложение *p-1*

Алгоритм из стандарта ГОСТ Р 34.10-94

- Теорема: Пусть $p = qN + 1$, где q >2 простое число, N – четное число и $p < (2q+1)^{2}$. Число p является простым, если $2^{qN} = 1 \, mod \, p$ и $2^{N} \neq 1 \, mod \, p$
- Алгоритм формирования числа *p* длины t$\geq$ 17 бит:
  - Построим убывающий набор натуральных чисел $t_0, t_1, \ldots, t_s$, в котором $t_0 = t$ и $t_s < 17$ бит, таким образом, что $t_i = [\frac{t_{i-1}}{2}]$
  - Последовательно генерируются простые числа $p_s, p_{s-1}, \ldots, p_0$: $p_{i-1} = p_i N + 1$
    - p_s формируется случайным выбором числа размером t_s бит и проверки на простоту методом пробного деления
    - N – случайное целое чётное число, такое, что размер числа $p_{i-1} = p_i N + 1$ равен значению t_{i-1}
  - Число p_{i-1} считается полученным, если $2^{p_i N} = 1 \, mod \, p_{i-1}$ и $2^{N} \neq 1 \, mod \, p_{i-1}$
  - Если одно из условий не выполнено, то N увеличивается на 2 и вычисляется новое p_{i-1} и так до получения простого числа p_{i-1}

Детерминированный AKS-алгоритм

- В 2002 г. индийские ученые Агравал, Каял и Саксена (**A**grawal, **K**ayal и **S**axena) объявили, что они нашли алгоритм для испытания простоты чисел с полиномиальной сложностью времени разрядных операций
- В основе этого теста на простоту лежит доказанное тождество:
  - Пусть $\text{НОД}(a,n) \equiv 1$. Число n является простым тогда, и только тогда, когда $(x-a)^n \equiv (x^n - a) \bmod n$

Теорема ASK

- Пусть $n \geq 2; n$ – целое; q, r – простые числа, причем :

а) $\forall m \in \{1, 2, \ldots, r\} : \text{НОД}(m, n) = 1$

б) $q \mid (r-1)$

в) $q \geqslant 4\sqrt{r}\log n$, где $log \equiv log_2$

г) $n^{(r-1)/q} \not\equiv 1 \pmod{r}$

д) $\forall a \in \{1, \ldots, \lfloor 2\sqrt{r}\log n \rfloor + 1\} : (x-a)^n \equiv (x^n - a) \pmod{n, x^r - 1}$

Тогда n – простое число

Псевдокод алгоритма ASK

```
r=2;
while(r<n){
    if(НОД(r,n)≠1) output составное;
    if(r - простое число, r > 2){
        q = наибольший простой делитель у (r − 1);
        if((q ⩾ 4√r log n) and ( n^((r−1)/q) ≢ 1 (mod r))) break;
    }
    r ← r + 1;
}
for a = 1 to (⌊2√r log n⌋ + 1)
    if((x − a)^n ≢ (x^n − a) (mod x^r − 1, n) ) output составное;
if(n = a^b;  a,b - целые; a,b ⩾ 2) output составное;
    else output простое;
```

- Сложность алгоритма $O(log_2{}^{19}n)$
- Это первый детерминированный алгоритм, работающий за полиномиальное время и имеющий изящное доказательство
- Алгоритм может оказаться практически применимым, если будет доказана правильность его более эффективного варианта