جامعة القادسية

كليـــة التربيـــة

قسم الرياضيات


# خوارزميات إيجاد الاعداد الاولية

بحث مقدم من قبل الطالب

حامد حمزة عبد وناس

الى مجلس قسم الرياضيات/ كلية التربية/ جامعة القادسية

كجزء من متطلبات نيل شهادة البكالوريوس علوم في الرياضيات

بأشراف

**م.م. زينب فهد مهوس**

2019-2018


**(اللّهُ لاَ إِلَـهَ إِلاَّ هُوَ الْحَيُّ الْقَيُّومُ لاَ تَأْخُذُهُ سِنَةٌ وَلاَ نَوْمٌ لَّهُ مَا فِي السَّمَاوَاتِ وَمَا فِي الأَرْضِ مَن ذَا الَّذِي يَشْفَعُ عِنْدَهُ إِلاَّ بِإِذْنِهِ يَعْلَمُ مَا بَيْنَ أَيْدِيهِمْ وَمَا خَلْفَهُمْ وَلاَ يُحِيطُونَ بِشَيْءٍ مِّنْ عِلْمِهِ إِلاَّ بِمَا شَاء وَسِعَ كُرْسِيُّهُ السَّمَاوَاتِ وَالأَرْضَ وَلاَ يَؤُودُهُ حِفْظُهُمَا وَهُوَ الْعَلِيُّ الْعَظِيمُ)**

**سورة البقرة الآية﴿255﴾**

# الأهداء

الشكر لله ولرسوله صلى الله عليه وآله

ونحنُ نخطو خطواتنا الاخيره في الحياة الجامعية لابد لنا من وقفه نعود إلى اعوام مضت في رحاب جامعة القادسية او في المؤسسات التربوية مع اساتذتنا الافاضل الذي قدموا لنا الكثير باذلين قصار جهدهم لبناء جيل الغد لتحيا الامه من جديد

وقبل ان نمضي نقدم أسمى ايات الشكر والعرفان.. الى الذي حملوا اقدس رسالة في الحياة

الى الذي مهدوا طريق العلم والمعرفة... اساتذتنا الافاضل فلهم الشكر الموصول....

((كن عالماً.. فإن لم تستطع فكن متعلما.. فإن لم تستطع فأحب العلماء.. فإن لم تستطع فلا تبغضهم))

الشكر والتقدير إلى الذين كانوا عوناً وقدموا لنا اليسير في بحثنا هذا واسهموا في زرع التفاؤل ...وأيضاً اقدم خالص الشكر...

فلولا وجودهم لما احسسنا بمتعه الدراسة وحلاوة المنافسة الايجابية

فلهم كل الشكر والتقدير....

**الملخص:**

تناول البحث إضافة دراسة خوارزميات إيجاد الاعداد الأولية وإيجاد خوارزميات تحديد العدد اذا كان اولي ام لا، كذلك كتابة برامج لتلك الخوارزميات وبلغات مختلفة مثل C++ وبرنامج اكسل في قسم المطور VBA حيث تم كتابة برمجة خاصة في إيجاد الاعداد الأولية المحصورة في فترة معينة تبداء من 2 الى حد معين يحدده المستخدم ولاعداد كبيرة واستخدامنا لبرنامج اكسل جاء من احتياجنا الى خلايا كثيرة لتمثيل الاعداد بها وثم فحصها وهذا ما يوفره برنامج اكسل.

## المحتويات

University of AL-Qadisiyah

Education College

Department of Mathematics

# Algorithms for finding prime numbers

A paper

Submitted to the council of mathematic dept.-College of education

University of Al- Qadisiyah As a Partial Fulfillment of the Requirements for the Degree Of Bachelor of Science in Mathematics

By

**Hamed Hamza Abd Wanas**

Supervised By

**Assist. Lecture Zainab Fahad Mhawes**

2018-2019

**Abstract:**

In this paper we write algorithms for different programing language such as C ++ and Excel in the VBA, where special programming was written to find initial numbers confined to a certain period starting from 2 to a certain limit defined by the user and a large number . Our use of the Excel program came from the need to many cells to represent the numbers and then checked and this is provided by Excel.

# الفصل الأول

# الفصل الثاني

# المصادر

## المـقدمـــــة

لقد عرّف العلماء العدد الأولي بأنه أي عدد أكبر من الواحد و عوامله الأولية الموجبة هي الواحد و العدد نفسه ، و عكس هذا هو العدد المركب ( Composite) و هو العدد الذي يمكن تحليله إلى عوامل أصغر منه ، فعلى سبيل المثال ، العدد 10 يمكن تجزئته إلى : $2 \times 5$ و بالتالي هو عدد مركب و ليس أولي ، و لكن العدد 7 لا يمكن تجزئته و بالتالي هو عدد أولي ، و أول ستة أعداد أولية هي : 2 ، 3 ، 5 ، 7 ، 11 ، 13.
يعتبر الإغريق هم أول من درس الأعداد الأولية و خصائصها ، حيث كان رياضيو مدرسة فيثاغورس ( 500 ق.م إلى 300 ق.م ) مهتمين بالأعداد و خصائصها السحرية و المنطق العددي ، فقد فهموا فكرة الأولية ، و كانت الأعداد التامة ( Pefect ) ، و الأعداد المتحابة ( Amicable ) موضع اهتمامهم .

و تعتبر مبرهنة فيرمات الصغيرة هي الأساس لكثير من النتائج في نظرية الأعداد ، و كذلك هي الأساس لعدة طرق لمعرفة الأعداد الأولية و التي ما زالت تستخدم حتى الآن في الحواسيب الإلكترونية .
و قد وافق فيرمات في ما توصل إليه مع رياضيي عصره ، و بالخصوص مع مونك مارين ميرسين ( Mersenne )ففي أحد رسائله إلى ميرسين تحدث فيرمات عن حدسه في أن العدد يكون أوليا دائما عندما يكون n من قوى العدد 2 ، مثل ( 1 ، 2 ، 4، 8 ، 16 ، ..... ) و قد تحقق من ذلك بالنسبة للأعداد , $n = 1$ )
( 16 , 8 , 4 , 2، و أوضح بأنه إذا كانت n ليس من قوى 2 فالنتيجة خاطئة .
و الأعداد من هذه الصورة سميت بأعداد فيرمات ، و قد كان فيرمات مخطئا في حدسه هذا و لم يتم إثبات ذلك إلا بعد أكثر من 100 سنة و ذلك عندما أثبت أويلر أن العدد :
4294967297 =يقبل القسمة على 641 و بالتالي فهو ليس أوليا .
أما بالنسبة للأعداد من الصورة فقد استدعت انتباه الرياضيين لسهولة إثبات أنه إذا لم يكن n عددا أوليا ، فيجب أن يكون العدد مركبا ، و قد سميت هذه الأعداد بأعداد ميرسين .

تكون هذا البحث من فصلين حيث تناول الفصل الأول مقدمة حول الاعداد الأولية اما الفصل الثاني فقد ناقش خوارزميات الاعداد الأولية وبرمجة هذه الخوارزمية باستخدام لغة البر مجة C++ وبرنامج اكسل عن طريق ميزة المطور في VBA.

## الفصل الاول

### O-الكبيرة و o-الصغيرة:

### 1.1 تعريف:

نقول ان $f$ O-الكبيرة للدالة $g$ اذا كان:

$$\lim_{N\to\infty} \frac{f(N)}{g(N)} = 1$$

كذلك نقول ان $f$ o-الصغيرة للدالة $g$ اذا كان:

$$\lim_{N\to\infty} \frac{f(N)}{g(N)} = 0$$

### 1.2 كيف يتم توزيع الأعداد الأولية:

في عام 1737 ، حقق L. Euler دليلًا جديدًا على أن هناك أعدادًا لا حصر لها من الأعداد الأولية:لقد أظهر أن مجموع التبادلات بين الأعداد الأولية هو مجموع متباعد ، وبالتالي يجب أن تحتوي على العديد من الحدود الغير منتهية.

### 1.3 تعريف:

$$\pi(x) = \#\{p: p \le x\,, p \text{ عدد اولي}\}$$

### 1.4 مثال:

$$\pi(5) = 3\,, \pi(2) = 1, \pi(10) = 4$$

### 1.5 مبرهنة (جبى-شيف):

يوجد هناك اعداد موجبة $A,B$ بحيث انه لكل $x \ge 3$ يكون:

$$\frac{Ax}{\ln x} \le \pi(x) \le \frac{Bx}{\ln x}$$

## 1.6 مبرهنة (هدمار):

عندما $x \to \infty$ تتحقق المعادلة:

$$\pi(x) \cong \frac{x}{\ln x}$$

## 1.7 مبرهنة (ليجندر):

$$\pi(x) \cong \frac{x}{\ln x - 1}$$

| $x$ | $\pi(x)$ |
|---|---|
| $10^2$ | 25 |
| $10^3$ | 168 |
| $10^4$ | 1229 |
| $10^6$ | 78498 |
| $10^8$ | 5761455 |
| $10^{12}$ | 37607912018 |
| $10^{16}$ | 279238341033925 |
| $10^{17}$ | 2623557157654233 |
| $10^{18}$ | 24739954287740860 |
| $10^{19}$ | 234057667276344607 |
| $10^{20}$ | 2220819602560918840 |
| $10^{21}$ | 21127269486018731928 |
| $10^{22}$ | 201467286689315906290 |
| $4 \cdot 10^{22}$ | 783964159847056303858 |

## 1.8 الاعداد الأولية المشهورة:

لقد أشرنا إلى أن تعريف الأعداد الأولية بسيط للغاية ، حتى الآن الأسئلة المتعلقة بالأعداد الأولية يمكن أن تكون صعبة للغاية. في هذا القسم نعرض مختلف المشاكل التي ناقشها التاريخ .

### 1.8.1 الأعداد الأولية المزدوجة:

النظر في حالة الأعداد الأولية المزدوجة ، وهذا يعني اثنين من الأعداد الأولية التي تختلف عن بعضها ب 2. من السهل العثور على هذه الأزواج ، مثل 11 أو 13 أو 197 ، 199. لكن الأزواج الكبيرة ليس سهل جدا، ولكن لا يزال من الممكن ، للعثور على أزواج كبيرة نسبيا ، أكبر النتائج الحديثة :

$835335 \cdot 239014 \pm 1$ ،

وجدت في عام 1998 من قبل R. Ballinger و Y. Gallot ،

الزوج

$361700055 \cdot 239020 \pm 1$ ،

وجدت في عام 1999 من قبل H. Lifchitz

$2409110779845 \cdot 260000 \pm 1$ ،

تأليف H. Wassing و A. J´arai و K.-H. Indlekofer ،

$665551035 \cdot 280025 \pm 1$ ،

بواسطة P. Carmody

$154798125 \cdot 2169690 \pm 1$ ،

ذكرت في عام 2004 من قبل .D. Papp

**1.8.2 اعداد مرسين:**

هناك عدة شروط لايجاد صيغ اعداد مرسين ومسألة مرسين هي مشكلة قديمة تمدد لعدة قرون .

**1.8.3 مبرهنة:**

اذا كان $M_q = 2^q - 1$ عدد اولي فان $q$ عدد اولي.

| | | | |
|---|---|---|---|
| $2^2-1$ | $2^3-1$ | $2^5-1$ | $2^7-1$ |
| $2^{13}-1$ | $2^{17}-1$ | $2^{19}-1$ | $2^{31}-1$ |
| $2^{61}-1$ | $2^{89}-1$ | $2^{107}-1$ | $2^{127}-1$ |
| $2^{521}-1$ | $2^{607}-1$ | $2^{1279}-1$ | $2^{2203}-1$ |
| $2^{2281}-1$ | $2^{3217}-1$ | $2^{4253}-1$ | $2^{4423}-1$ |
| $2^{9869}-1$ | $2^{9941}-1$ | $2^{11213}-1$ | $2^{19937}-1$ |
| $2^{21701}-1$ | $2^{23209}-1$ | $2^{44497}-1$ | $2^{86243}-1$ |
| $2^{110503}-1$ | $2^{132049}-1$ | $2^{216091}-1$ | $2^{756839}-1$ |
| $2^{859433}-1$ | $2^{1257787}-1$ | $2^{1398269}-1$ | $2^{2976221}-1$ |
| $2^{3021377}-1$ | $2^{6972593}-1$ | $2^{13466917}-1$ | $2^{20996011}-1$ |
| $2^{24036583}-1$ | $2^{25964951}-1$ | | |

### 1.8.4 مبرهنة (اقليد-اويلر):

أي عدد زوجي $n$ هو تام (يساوي مجموع قواسمه الأولية) اذا وفقط اذا كان بالصيغة الاتية:

$$n = 2^{q-1}M_q$$

بحيث $M_q = 2^q - 1$ عدد اولي.

### 1.8.5 اعداد فيرمات:

من الاعداد المشهورة هي اعداد فيرمات $F_n = 2^{2^n} + 1$ مثل اعداد ميرسن ، التي ظهرت في عام 1637 حيث ادعى فيرمات ان هذه الاعداد هي اعداد أولية والتي كان بها فيرمات خاطئ وربما خاطئ جداً حيث كانت هناك اعداد تمتلك عوامل قسمة مثل $F_5$ .

### 1.8.6 مبرهنة:

اذا كان $p = 2^m + 1$ عدد اولي فردي فان $m$ هي قوى للعدد 2.

**1.8.7 مبرهنة :**

لكل $n \geq 2$ فان أي عامل اولي $p$ لعدد فيرمات $F_n = 2^{2^n} + 1$ يجب ان يحقق المعادلة الاتية:

$$p = 1 \bmod 2^{n+2}$$

**1.8.8 مبرهنة(ويلسن-لاكرانج):**

ليكن $p$ عدد صحيح اكبر من واحد، فان $p$ عدد اولي اذا وفقط اذا كان:

$$(p-1)! = -1 \bmod p$$

**1.9 تخمينات الاعداد الأولية:**

1-يوجد عدد لانهائي من الاعداد الأولية على الشكل $n^2+1$ وهذا التخمين لم يثبت لحد الان.

2-تخمين جولدباخ عام 1742م:

يمكن التعبير عن أي عدد صحيح زوجي اكبر من 2 كمجموع عددين اولين.

3-يوجد عدد لانهائي من الاعداد الأولية على الشكل $p+2$ حيث $p$ عدد اوليوهي الاعداد المزدوجة.

4- تخمين لاكرانج عام 1775م: اذا كان $n$ عدد صحيح فردي اكبر من 5 فان $n=p_1+2p_2$ عددين اولين.

**1.10 تعريف:**

الترتيب الطبيعي للاعداد الأولية هو $p_1=2,p_2=3,\ldots,p_n,\ldots$ ويسمى $p_n$ العدد الاولي النوني في الترتيب الطبيعي.

**1.11 مبرهنة:**

لكل $n\in N$ فان المتراجحة الاتية متحققة:

$$p_n \leq 2^{2^{n-1}}$$

**1.12 نتيجة:**

اذا كان $n \geq 1$ عدداً صحيحاً فيوجد على الأقل $n+1$ من الاعداد الأولية كل منها اقل من $2^{2^n}$

**البرهان:**

بما ان كلا من $p_1,p_2,\ldots,p_{n+1}$ اقل من $2^{2^n}$ اذن وحسب النظرية أعلاه فانه يوجد على الأقل $n+1$ من الاعداد الأولية كل منها اقل من $2^{2^n}$.

**1.13 مبرهنة:**

1-كل عدد صحيح اكبر من الواحد يقبل القسمة على عدد اولي.

2-مجموعة الاعداد الأولية لانهائية.

**البرهان:**

1-نفرض ان :

$$A = \{n \in N:\ n \text{ اكبر من الواحد ولايقبل القسمة على عدد اولي}\} \neq \phi$$

اذا المجموعة $A$ تحوي عنصر اصغر وليكن $m$ . اذا $m$ اكبر من الواحد ولا يقبل القسمة على عدد اولي فاذا كان $m$ عددا اولي فان $m/m$ وهذا تناقض. اما اذا كان $m$ غير اولي فان $r$ يقبل القسمة على عدد اولي وليكن $p$ وعليه فان $p/m$ وهذا تناقض ، اذا $A = \phi$>

2-نفرض ان مجموعة الاعداد الأولية مجموعة منتهية وان عناصرها هي $p_1,p_2,\ldots,p_n$ وليكن $n=1+(p_1p_2\ldots p_n)$ اذا $n>1$ وعليه فان $n$ يقبل القسمة على عدد اولي مثل $p$ وحسب (1) من المبرهنة

. فاذا كان $p=p_i$ لبعض قيم $1 \le i \le n$ فان $p\backslash n$ و $p\backslash(p_1p_2\dots p_n)$ وعليه فان $p\backslash 1$ أي ان $p=1$ وهذا تناقض لان p عدد اولي. اذن $p \ne p_i$ لكل $i=1,2,\dots,n$ وعليه فان مجموعة الاعداد الأولية مموعة لانهائية.

## 1.14 نتيجة:

كل عدد غير اولي n يمتلك قاسم اولي p بحيث $p \le \sqrt{n}$

**البرهان:**

بما ان n عدد غير اولي اذاً $n=ab$ ، $1 < a \le b$ وعليه فان $a^2 \le n$ وبالتالي فان

$a \le \sqrt{n}$ اذا يوجد عدد اولي p بحيث $p\backslash a$ لكن $a\backslash n$ اذاً $p\backslash n$ و $p \le a$ وعليه $p \le \sqrt{n}$ .

## 1.15 دوال تعريف الاعداد الأولية:

للحصول على الاعداد الأولية من الطبيعي ان نحتاج الى تعريف دوال f المعرفة على كل الاعداد الطبيعية $n \ge 1$ التي تحسب بعض او كل الاعداد الأولية. هذه الدوال يمكن ان تعتمد على شروط معينة وسوف نناقش بعض هذه الشروط:

1- $f(n) = p_n, \quad n \ge 1$

2- $f(n)$ عدد اولي دائماً, اذا كان $n \ne m \rightarrow f(n) \ne f(m)$

3- مجموعة الاعداد الاولة هي مجموعة الاعداد الموجبة التي تنتج من الدالة f

### 1.15.1 الدوالة التي تحقق الشرط (1):

في هذه الحالة يظهر لدينا سؤالين هما :

- هل هناك صيغة للعدد الاولي من الرتبة n.
- -هل هناك صيغة للعدد الاولي بحدود الاعداد التي قبله

بالنسبة للسؤال الأول يعتمد على إيجاد الحد النوني للاعداد الأولية وهذا الحد يرتبط ارتباط وثيق بعدد الاعداد الأولية لذلك فهو يرتبط بالدالة $\pi(x)$ لذللك حاول العلماء إيجاد صيغة عامة للدالة $\pi(x)$ مثل صيغة ويلانس عام 1964 وهي :

$$\pi(n) = -1 + \sum_{j=1}^{n} F(j)$$

$$F(j) = \left[ cos^2 \pi \frac{(j-1)! + 1}{j} \right],$$

$$F(j) = 1 \ \ if\ j\ prime\ , F(j) = 0\ otherwise\ , F(1) = 1$$

## 1.16 مبرهنة البواقي الصينية:

لنأخذ نظام البواقي الاتية:

$$x = a_1 \ mod\ n_1$$

$$x = a_2 \ mod\ n_2$$

$$\vdots$$

$$x = a_k \ mod\ n_k$$

بحيث $n_1, n_2, \dots, n_k$ اعداد أولية ثنائية ، فان هناك حل واحد فقط x في الفترة $[1, n_1 n_2 \dots n_k - 1]$ ، كذلك فان مجموعة كل الحلول هي مجموعة كل الاعداد الصحيحة y بحيث ان:

$$y = x \ mod\ n_1 n_2 \dots n_k$$

## 1.17 نموذج الحل للنظام:

$$x = a_1 \, mod \, n_1$$

$$x = a_2 \, mod \, n_2$$

$$\vdots$$

$$x = a_k \, mod \, n_k$$

هو:

$$N = n_1 . n_2 \ldots . n_k$$

$$N_i = N/n_i$$

$$N_i' = N_i^{-1} mod \, n_i$$

$$x = \left( \sum_{i=1}^{k} a_i N_i \, N_i' \right) mod \, N$$

**1.18 مثال:**

حل النظام الاتي:

$$x = 1\ mod\ 3$$

$$x = 1\ mod\ 4$$

$$x = 1\ mod\ 5$$

$$x = 0\ mod\ 7$$

$$N = 3.4.5.7 = 420$$

$$x = 1\ mod\ 3 \rightarrow N_1 = 140\ \rightarrow N_1' = N_1^{-1}\ mod\ 3 = 2$$

$$x = 1\ mod\ 4 \rightarrow N_2 = 105\ \rightarrow N_2' = N_2^{-1}\ mod\ 4 = 1$$

$$x = 1\ mod\ 5 \rightarrow N_3 = 84\ \rightarrow N_3' = N_3^{-1}\ mod\ 5 = 4$$

$$x = 0\ mod\ 7 \rightarrow N_4 = 60\ \rightarrow N_4' = N_4^{-1}\ mod\ 7 = 2$$

$$x = 1.140.2 + 1.105.1 + 1.84.4 + 0.60.2$$

$$= 280 + 105 + 336 = 721\ mod\ 420 = 301$$

## الفصل الثاني:

## 2. خوارزميات إيجاد الاعداد الأولية:

### 2.1 اختبار لوكاس للاولية:

اذا كان p عدد أكبر من واحد يكون أولي إذا وفقط إذا كانت المقسومات الوحيدة لـ p هي 1 و p. الأعداد الأولية القليلة الأولى هي 2 ، 3 ، 5 ، 7 ، 11 ، 13 ، ...

اختبار Lucas هو اختبار أولي لعدد طبيعي n ، يمكنه اختبار الاولية من أي نوع.

يتبع من نظرية Fermat Little Theorem: إذا كانت p أولية و a عددًا صحيحًا ، فإن

$$a^p = a \bmod p$$

**Lucas' Test** : A positive number **n** is prime if there exists an integer **a** (1 < a < n) such that :

$$a^{n-1} \equiv 1 \pmod{n}$$

And for every prime factor q of (n-1),

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

```
lucasTest(n):
If n is even
    return composite
Else
   Find all prime factors of n-1
   for i=2 to n-1
      pick 'a' randomly in range [2, n-1]
      if a^(n-1) % n not equal 1:
          return composite
      else
          // for all q, prime factors of (n-1)
          if a^(n-1)/q % n not equal 1
             return prime
   Return probably prime
```

```cpp
// CPP Program for Lucas Primality Test
#include <bits/stdc++.h>
using namespace std;

// function to generate prime factors of n
void primeFactors(int n, vector<int>& factors)
{
    // if 2 is a factor
    if (n % 2 == 0)
        factors.push_back(2);
    while (n % 2 == 0)
        n = n / 2;

    // if prime > 2 is factor
    for (int i = 3; i <= sqrt(n); i += 2) {
        if (n % i == 0)
            factors.push_back(i);
        while (n % i == 0)
            n = n / i;
    }
    if (n > 2)
      factors.push_back(n);
}
```

```
// this function produces power modulo
// some number. It can be optimized to
// using
int power(int n, int r, int q)
{
    int total = n;
    for (int i = 1; i < r; i++)
        total = (total * n) % q;
    return total;
}

string lucasTest(int n)
{
    // Base cases
    if (n == 1)
        return "neither prime nor composite";
    if (n == 2)
        return "prime";
    if (n % 2 == 0)
        return "composite1";



// Generating and storing factors
// of n-1
vector<int> factors;
prime_factors(n - 1, factors);

// Array for random generator. This array
// is to ensure one number is generated
// only once
int random[n - 3];
for (int i = 0; i < n - 2; i++)
    random[i] = i + 2;

// shuffle random array to produce randomness
shuffle(random, random + n - 3,
          default_random_engine(time(0)));

// Now one by one perform Lucas Primality
// Test on random numbers generated.
for (int i = 0; i < n - 2; i++) {
    int a = random[i];
    if (power(a, n - 1, n) != 1)
        return "composite";
```

```
            // this is to check if every factor
            // of n-1 satisfy the condition
            bool flag = true;
            for (int k = 0; k < factors.size(); k++) {
                // if a^((n-1)/q) equal 1
                if (power(a, (n - 1) / factors[k], n) == 1) {
                    flag = false;
                    break;
                }
            }

            // if all condition satisfy
            if (flag)
                return "prime";
        }
        return "probably composite";
}

// Driver code

     int main()
     {
         cout << 7 << " is " << lucasTest(7) << endl;
         cout << 9 << " is " << lucasTest(9) << endl;
         cout << 37 << " is " << lucasTest(37) << endl;
         return 0;
     }
```

## 2.2 AKS اختبار الاولية

هناك العديد من اختبارات الاولية المتاحة للتحقق مما إذا كان الرقم أوليًا أم لا مثل نظرية فيرمات ، واختبار ميلر رابين الاولية ، وغير ذلك الكثير. ولكن المشكلة مع كل منهم هو أنهم جميعا احتمالية. لذلك ، هنا يأتي طريقة اختبار AKS الاولية (اختبار الاولية لـ Agrawal-Kayal-Saxena)وهو صحيح بشكل قاطع لأي رقم عام.

**ميزات اختبار البدائية :AKS**

1. يمكن استخدام خوارزمية AKS للتحقق من أولوية أي رقم عام معين.

2. يمكن التعبير عن الحد الأقصى لوقت تشغيل الخوارزمية على شكل كثير الحدود على عدد الأرقام في الرقم المستهدف.

3. الخوارزمية مضمونة للتمييز بشكل قاطع ما إذا كان الرقم المستهدف أولي أو مركب.

4. صحة AKS ليست مشروطة بأي فرضية فرعية غير مثبتة.

تعتمد هذه الخوارزمية على المتطابقة الاتية:

$$(x + a)^n = x^n + a \ mod \ n$$

وبذلك يمكن كتابة المتطابقة السابقة كالاتي:

$$(x + a)^n - (x^n + a) = (x^r - 1)g + f$$

بحيث ان f,g متعددات الحدود.

Checking for n = 3 :

(x-1)^3 – (x^3 – 1)
= (x^3 – 3x^2 + 3x – 1) – (x^3 – 1)
= -3x^2 + 3x

```cpp
// C++ code to check if number is prime. This
// program demonstrates concept behind AKS
// algorithm and doesn't implement the actual
// algorithm (This works only till n = 64)
#include <bits/stdc++.h>
using namespace std;

// array used to store coefficients .
long long c[100];

// function to calculate the coefficients
// of (x - 1)^n - (x^n - 1) with the help
// of Pascal's triangle .
void coef(int n)
{
    c[0] = 1;
    for (int i = 0; i < n; c[0] = -c[0], i++) {
        c[1 + i] = 1;

        for (int j = i; j > 0; j--)
            c[j] = c[j - 1] - c[j];
    }
}



    // function to check whether
    // the number is prime or not
    bool isPrime(int n)
    {
        // Calculating all the coefficients by
        // the function coef and storing all
        // the coefficients in c array .
        coef(n);

        // subtracting c[n] and adding c[0] by 1
        // as ( x - 1 )^n - ( x^n - 1), here we
        // are subtracting c[n] by 1 and adding
        // 1 in expression.
        c[0]++, c[n]--;

        // checking all the coefficients whether
        // they are divisible by n or not.
        // if n is not prime, then loop breaks
        // and (i > 0).
        int i = n;
        while (i-- && c[i] % n == 0)
            ;
```

```
    // Return true if all coefficients are
    // divisible by n.
    return i < 0;
}

// driver program
int main()
{
    int n = 37;
    if (isPrime(n))
        cout << "Prime" << endl;
    else
        cout << "Not Prime" << endl;
    return 0;
}
```

## 2.3 خوارزمية إيجاد اعداد ميرسن:

```
// Program to generate mersenne primes
#include<bits/stdc++.h>
using namespace std;

// Generate all prime numbers less than n.
void SieveOfEratosthenes(int n, bool prime[])
{
    // Initialize all entries of boolean array
    // as true. A value in prime[i] will finally
    // be false if i is Not a prime, else true
    // bool prime[n+1];
    for (int i=0; i<=n; i++)
        prime[i] = true;

    for (int p=2; p*p<=n; p++)
    {
        // If prime[p] is not changed, then it
        // is a prime
        if (prime[p] == true)
        {
            // Update all multiples of p
            for (int i=p*2; i<=n; i += p)
                prime[i] = false;
        }
    }
}
```

```
// Function to generate mersenne primes less
// than or equal to n
void mersennePrimes(int n)
{
    // Create a boolean array "prime[0..n]"
    bool prime[n+1];

    // Generating primes using Sieve
    SieveOfEratosthenes(n,prime);

    // Generate all numbers of the form 2^k - 1
    // and smaller than or equal to n.
    for (int k=2; ((1<<k)-1) <= n; k++)
    {
        long long num = (1<<k) - 1;

        // Checking whether number is prime and is
        // one less then the power of 2
        if (prime[num])
            cout << num << " ";
    }
}
```

```
// Driven program
int main()
{
    int n = 31;
    cout << "Mersenne prime numbers smaller "
         << "than or equal to " << n << endl;
    mersennePrimes(n);
    return 0;
}
```

## 2.4 إيجاد الاعداد الأولية باستخدام برنامج اكسل:

```
Dim prime As Double
Dim prime2 As Double
Dim startx As Double
Dim starty As Double
Dim toplim As Double
Dim r2 As Double
Dim rcol As Integer
Dim gcol As Integer
Dim bcol As Integer
Sub start()

toplim = Application.InputBox("Please enter the highest number")

r = 1
r2 = 1
c = 2
prime = 1

Do Until numbs > Sqr(toplim)
```

```
If Cells(r, c).Value = "" Then
prime = c + ((r - 1) * 10)
Cells(r, c).Value = prime
Cells(r, c).Font.ColorIndex = 3
Cells(r, c).Font.Bold = True
Cells(r2, 13).Value = prime

rcol = Int((255 - 0 + 1) * Rnd + 0)
gcol = Int((255 - 0 + 1) * Rnd + 0)
bcol = Int((255 - 0 + 1) * Rnd + 0)

Cells(r, c).Interior.color = RGB(rcol, gcol, bcol)
Cells(r2, 13).Interior.color = RGB(rcol, gcol, bcol)

r2 = r2 + 1

Call primer

Else:
c = c + 1
If c > 10 Then
```

```
                c = 1
            r = r + 1
                End If
            End If
numbs = c + ((r - 1) * 10)
            Loop


            Call fill


            End Sub
            Sub primer()


prime2 = prime ^ 2


primep = Int(prime2 / 10)


startx = 1 + primep


starty = prime2 - (primep * 10)
Cells(startx, starty).Value = prime2
```

```
Cells(startx, starty).Interior.color = RGB(rcol, gcol, bcol)

Call composite

End Sub

Sub composite()

num = prime + prime2

x = startx

y = starty

Do Until num > toplim

y = num Mod 10

x = Int(num / 10)

If num Mod 10 = 0 Then x = x - 1

If y = 0 Then y = 10

Cells(1 + x, y).Value = num
```

```
Cells(1 + x, y).Interior.color = RGB(rcol, gcol, bcol)

num = num + prime

Loop

End Sub
Sub fill()

r1 = 1
c1 = 2
Do Until c1 + ((r1 - 1) * 10) > toplim

If Cells(r1, c1).Value = "" Then
Cells(r1, c1).Value = c1 + ((r1 - 1) * 10)
Cells(r1, c1).Interior.ColorIndex = 5
Cells(r1, c1).Font.ColorIndex = 3
Cells(r1, c1).Font.Bold = True
Cells(r2, 13).Value = Cells(r1, c1).Value
r2 = r2 + 1
End If
c1 = c1 + 1
```

```
If c1 > 10 Then
c1 = 1
r1 = r1 + 1
End If
Loop

End Sub

Sub cleaner()
r1 = 1
c1 = 2
Do Until c1 + ((r1 - 1) * 10) > 1000000

If Cells(r1, c1).Font.ColorIndex = 3 Then

Cells(r1, c1).Interior.ColorIndex = 5

Else:
Cells(r1, c1).Value = ""
Cells(r1, c1).Interior.ColorIndex = 0
End If
c1 = c1 + 1
```

```
If c1 > 10 Then
c1 = 1
r1 = r1 + 1
End If
Loop


End Sub

Sub tellall()

ccc = Cells(1, 3).Interior.ColorIndex
MsgBox ccc

End Sub
Sub clean2()
r1 = 1
c1 = 2
Do Until c1 + ((r1 - 1) * 10) > 1000

If Cells(r1, c1).Interior.ColorIndex = 53 Then
```

```
Cells(r1, c1).Interior.ColorIndex = 36

Else:
Cells(r1, c1).Value = ""
Cells(r1, c1).Interior.ColorIndex = 0
End If
c1 = c1 + 1
If c1 > 10 Then
c1 = 1
r1 = r1 + 1
End If
Loop
End Sub
Sub composite2()

prime = 83
num = prime
toplim2 = 10000
x = 0
y = num

Do Until num > toplim2
```

```
y = num Mod 10

x = Int(num / 10)
If num Mod 10 = 0 Then x = x - 1

If y = 0 Then y = 10

Cells(1 + x, y).Value = num
Cells(1 + x, y).Interior.ColorIndex = 4

num = num + prime

Loop

End Sub
```

## المصادر


1-Richard Crandall Carl Pomerance, Prime numbers,springer, USA, 2000.

2-2007 د.فالح بن عمران بن محمد، مقدمة في نظرية الاعداد، مكة المكرمة-جامعة ام القرى،

3-.2017 د. أسامة الكامل، مايكرو سوفت اكسل، مصر، القاهرة،

4-.2015 د. انس الشيخ الخفاجي، البرمجة والحاسب، سوريا، حلب،