ProjectEuler Archive 19: Gregorian Calendar Calculations

YarrowForAlgernon

August 2025

Contents

1 Challenge Description

Challenge Name: *Counting Sundays*

Challenge Description:

> *You are given the following information, but you may prefer to do some research for yourself.*
>
> *1 Jan 1900 was a Monday. Thirty days has September, April, June and November. All the rest have thirty-one, Saving February alone, Which has twenty-eight, rain or shine. And on leap years, twenty-nine. A leap year occurs on any year evenly divisible by 4, but not on a century unless it is divisible by 400.*
>
> *How many Sundays fell on the first of the month during the twentieth century (1 Jan 1901 to 31 Dec 2000)?*

2 The Naive Solution

2.1 The First Thing That Comes To Mind

To solve the challenge we need to calculate the number of Sundays that fall on the first of a month over a period of 1200 months (1200 months in 100 years from January 1st 1901 - December 31st 2000).

We know from the challenge description that April, June, September and November have 30 days, and all the rest except February have 31. We can calculate what day the first of each month will be through modular arithmetic.

The first solution I thought of was to set a 'day counter' variable to 6, as the first Sunday of 1901 is the 6th of January, and continually increment it by 7, the number of days in a week, to keep it pointing to Sunday. Once the day counter exceeds the number of days in that month, we take it modulo the number of days in that month, the result of which will be the first Sunday of the next month and we repeat this process until we reach 31 December 2000. This means we will have to loop through all 5,200 weeks in the 100-year period, but provides a straightforward way to brute force the answer.

2.2 The Algorithm

This algorithm has a time complexity of O(n), and a space complexity of O(1).

Link to code.

3 Zellers Congruence

3.1 Introduction

In 1882 Julius Christian Johannes Zeller published an equation to calculate what day falls on any given date [1]. Using this method we can compute the beginning of each month and determine if it is a Sunday.

This allows us to loop month by month rather than week by week as we directly calculate the first of each month and check if it is a Sunday.

3.2 Different Expressions Of Zellers Congruence

Theorem 3.1 (The Original Equation). Here is the formula Zeller published for the Gregorian calendar in 1882:

$$h = q + \lfloor \frac{26(m+1)}{10} \rfloor + k + \lfloor \frac{k}{4} \rfloor - 2e \mod 7 \tag{3.1}$$

Where h is the day of the week ($0 =$ Tuesday), q is the day of the month, m is the number of the month in the year, J is the number of the century, K is the year of that century and e is $J/4$

Remark 3.1. January and February are computed as the 13th and 14th months of the previous year in Theorem 3.1, for example 01/01/1967 (DD/MM/YYYY) is interpreted as 01/13/1966.

Remark 3.2. Zeller notes the divisions in Theorem 3.1 were actually taking the quotient, which gives an integer result. This is why we add floor symbols around all divisions, as taking the quotient is equivalent to taking the floor of the fractions.

Zeller went on to continue his research into calculating dates and came up with the following formula:

Corollary 3.1.1 (Zellers Congruence). This is what is commonly referred to today as Zellers Congruence, which was taken from this paper [2].

$$h = q + \lfloor \frac{26(m+1)}{10} \rfloor + K + \lfloor \frac{K}{4} \rfloor + \lfloor \frac{J}{4} \rfloor - 2J \mod 7 \tag{3.2}$$

Where h is the day of the week (0 = Saturday), q is the day of the month, m is the number of the month in the year (3=March, 14 = February), J is the number of the century and K is the year within that century

Additionally, at the end of the same paper Zeller describes a useful substitution for K and J, which results in the formula we will be using:

Corollary 3.1.2 (Modified Zellers Congruence).

$$h = q + \lfloor \frac{26(m+1)}{10} \rfloor + Y + \lfloor \frac{Y}{4} \rfloor - \lfloor \frac{Y}{100} \rfloor + \lfloor \frac{Y}{400} \rfloor \mod 7 \tag{3.3}$$

where h is the day of the week (0 = Saturday), q the day of the month, m is the month (3 = March, 14 = February) and Y is the current year

Remark 3.3. There is actually an even more compact form of Zellers Congruence, which is described in this paper [3]. It also explains Corollary 3.1.2 in greater detail than in this section, so I recommend anyone interested in learning the meaning behind the terms of Zellers congruence to read this paper.

3.3 The Month By Month Algorithm

The time complexity of this algorithm is O(n).

Link to code.

4 14 Unique Gregorian Years

4.1 A Brief Summary Of The Gregorian Calendar

The Gregorian calendar was introduced as a replacement to the Julian calendar in 1582. This was done in order to consistently calculate when Easter occurred and more accurately simulate the rotation of the earth around the sun (or, as then believed, the suns' rotation around the earth).

The previous Julian calendar stated every four years was a leap year, which was on average 356.25 days a year. While close, the real number of days taken for the earth to complete one full orbit around the sun is closer to 365.2422 days, referred to as a tropical year.

The Gregorian calendar changed its calculations to reduce the number of leap years to better reflect this number, though it is still off from the tropical year by roughly 26 seconds every year [4].

4.2 Explanation Of 'Unique' Gregorian Years

Because the number of weeks in a non-leap year (AKA a 'common' year) is 52 weeks and one day, every year will start and end with the same day. This means what days of the week fall on the days of the year can be immediately known. Because a common year can only start with 7 different days, there can only be 7 unique common years. For example, 1902 and 1913 have the exact same arrangement of what days of the week fall on what days of the year, because they both start on a Wednesday and are therefore the same 'unique' common year.

Leap years have 52 weeks and 2 days with the addition of the 29th of February, which means instead of starting and ending on the same day, they end one day ahead of the day the year started with. For example, if a leap year begins with Tuesday instead of ending with Tuesday, as is the case in a common year, it will end on a Wednesday. Leap years also only have 7 'unique' years, as there are only 7 days of the week with which the year can start with.

All together, there are 14 unique Gregorian years and the only two variables needed to differentiate between unique years are the day the year starts on, and whether it is a leap year.

We can store the number of Sundays that fall on the first of a month in each 'unique' year and match them to each year from 1901 to 2000 depending on what day of the week they start with, and whether it is a leap year. This way, we can loop exactly n times by matching each year to its respective unique year and sum the number of Sundays that fall on the first of the those unique years months.

My initial thought was to use Zellers congruence to calculate the start of each year, but this is not actually necessary. These unique Gregorian years progress in a predictable pattern.

If it is a common year, the day the next year begins with is the following day the current year begins with, and two days over if the it is a leap year. For example, if 2002 (common year) begins with a Tuesday, 2003 will begin with a Wednesday. If 2004 (leap year) begins with Thursday, 2005 will begin with Saturday.

Given that we know that 1st January 1900 is a Monday, we can tell what day all subsequent years will start with using simple arithmetic, i.e. 1901 starts will with a Tuesday, 1902 with a Wednesday and so on.

This way, we no longer need to use Zellers congruence. Using a dictionary of what unique year corresponds to the number of Sundays that fall on the 1st of its months, the number of times we loop is the exact number of years to calculate.

4.3 The Unique Year Lookup Algorithm

The time complexity of this algorithm is O(n).

Link to code.

5 The 28 And 400-Year Solar Cycle

5.1 A Simple(ish) Sequence

There is a property of the sequence of unique years in the Julian calendar - they have a period of 28 years. This is not completely accurate in the Gregorian calendar, as years that are divisible by 100 but not 400 (1900, 1700, etc) disrupt this cycle. Despite these 'disruptive' years the exact sequence of unique years repeats every 400 years, hence the 400-year Gregorian solar cycle.

However, for our case we do not need to take this into consideration as 1901 to 2000 includes no 'disruptive' years. This allows for an unbroken 28-year cycle starting from January 1st 1901.

The number of Sundays that fall on the first of each month in the 28-year cycle from 1901 to 2000 is exactly 48 (calculated with the help of our algorithm in Section 4.3).

So if we take the quotient of 100 years (1901 to 2000) divided by the 28 years needed to complete a solar cycle ($\lfloor \frac{100}{28} \rfloor$) and multiply it by 48 we will get the number of Sundays that fall on the first of a month within 84 years from 1901 to 2000, with the remaining years not being enough to complete a full cycle.

Then we can calculate the remaining years within the last 28-year cycle using our lookup algorithm in Section 4.3. Note that the year we start with (1901) and the start of the remaining years we'll get after the last complete 28 year cycle (1985) will start on the same day, due to the fact that the sequence of unique years in the 28-year cycle repeats.

5.2 The 28-Year Solar Cycle Algorithm

The time complexity of this algorithm is O(1) as the amount of loops will never exceed 28, so the number of calculations has a very low upper limit. In this case the asymptotic time complexity is not a very useful way to describe the runtime of this algorithm.

Remark 5.1. because this calculates the 28-year cycle instead of the 400-year cycle, it will not work across 'disruptive' years (1900, 1700, 2100, etc) and is only applicable

to our challenge.

Link to code.

6 Bibliography

References

[1] C. Zeller, "Die grundaufgaben der kalenderrechnung auf neue und vereinfachte weise gelöst," German, *Württembergische Vierteljahrshefte für Landesgeschichte*, vol. 5, pp. 313–314, 1882. DOI: 10.53458/wvjh.v5i.14389.

[2] C. Zeller, "Kalender-formeln," German, *Acta Mathematica*, vol. 9, pp. 131–136, 1887. DOI: 10.1007/BF02406733.

[3] R. Sivaraman, "Determining day of given date mathematically," *Mathematics and Statostics*, vol. 8, no. 5, pp. 590–595, 2020. DOI: 10.13189/ms.2020.080514.

[4] J. C. Kolecki, D. Mazza, and T. M. Scott. "Calendar calculations." (2001), [Online]. Available: https://www.grc.nasa.gov/www/k-12/Numbers/Math/Mathematical_Thinking/calendar_calculations.htm. (accessed: 07.08.2025).