NCERT Archive

SOLUTIONS BOOK

CLASS 9

Mathematics · Ganita Manjari

CHAPTER 11

The World of Algorithms

Complete Exercise Solutions

Exercise Set 11.1 | Exercise Set 11.2 | Exercise Set 11.3 | End-of-Chapter Exercises | In-Text Questions

Contents

Class 9 Mathematics · Ganita Manjari · Chapter 11 — The World of Algorithms

11.1 **Exercise Set 11.1** — 5 questions · Q1 Q2 Q3 Q4 Q5 — 3

11.2 **Exercise Set 11.2** — 3 questions · Q1 Q2 Q3 — 7

11.3 **Exercise Set 11.3** — 2 questions · Q1 Q2 — 10

EOC **End-of-Chapter Exercises** — 5 questions · Q1 Q2 Q3 Q4 Q5 — 12

IT **In-Text Questions** — 4 questions · T1 T2 T3 T4 — 18

Every question is followed by its answer. Question numbers above are clickable and the PDF bookmarks panel lets you jump to any exercise set or question.

NCERT Archive

11.1

CLASS 9 MATHS · GANITA MANJARI · CHAPTER 11

Exercise Set 11.1

Solutions · 5 questions

QUESTION 1

Add two 4-digit numbers using the steps we have written down. Make sure you follow the steps precisely; do not perform any action that is not explicitly mentioned. Are you able to obtain the correct result?

ANSWER

Take the 4-digit numbers 3847 and 5692. Write them one below the other with the digits aligned from right to left (Step 1), then follow Steps 2–5 exactly.

Units: $7 + 2 = 9$, less than 10 → write 9, carry = 0
Tens: $4 + 9 + 0 = 13$, more than 10 → write 3, carry = 1
Hundreds: $8 + 6 + 1 = 15$, more than 10 → write 5, carry = 1
Thousands: $3 + 5 + 1 = 9$, less than 10 → write 9, carry = 0

No digits are left on the left (Step 4) and carry = 0, so Step 5 adds nothing. The bottom row reads **9539**. Check: 3847 + 5692 = 9539.

A second example in which Step 5 is needed: 6758 + 5491.

Units: $8 + 1 = 9$ → write 9, carry = 0
Tens: $5 + 9 + 0 = 14$ → write 4, carry = 1
Hundreds: $7 + 4 + 1 = 12$ → write 2, carry = 1
Thousands: $6 + 5 + 1 = 12$ → write 2, carry = 1
Step 5: carry = 1 → write 1 to the left → **12249**

Check: 6758 + 5491 = 12249.

Remark. Steps 2 and 3 only mention column sums “less than 10” and “more than 10”. A column sum of exactly 10 (for example 4 + 6) is not mentioned; we read it as “10 or more”, as in the introduction of the chapter (“equal to or more than 10”). With this reading the steps always work.

Yes, the steps give the correct result: 3847 + 5692 = 9539 and 6758 + 5491 = 12249.

NCERT Archive

QUESTION 2

What happens if you add a 5-digit number to a 3-digit number. Do our steps handle this situation correctly?

ANSWER

Take 23764 + 589. Step 1 aligns the digits from the right and we proceed column by column.

Units: $4 + 9 = 13 \rightarrow$ write 3, carry = 1
Tens: $6 + 8 + 1 = 15 \rightarrow$ write 5, carry = 1
Hundreds: $7 + 5 + 1 = 13 \rightarrow$ write 3, carry = 1

Now only 23764 has digits left (3 in the thousands column and 2 in the ten-thousands column). Step 3 says to “add the two digits and the current value of carry”, but each of these columns has only one digit, and the steps do not say what to do when one number runs out of digits. So, as written, the steps do not tell us how to finish.

The steps work if the missing digits of the shorter number are treated as 0, i.e. 589 is written as 00589:

Thousands: $3 + 0 + 1 = 4 \rightarrow$ write 4, carry = 0
Ten-thousands: $2 + 0 + 0 = 2 \rightarrow$ write 2, carry = 0

The result is **24353**. Check: 23764 + 589 = 24353. If we had just copied the leftover digits and ignored the carry, we would get 23353, which is wrong.

The steps as written do not fully handle this case. They work correctly once the shorter number is padded with leading zeros (or a rule is added: if only one number has a digit in a column, add that digit and the carry).

QUESTION 3

Why is it important to align the columns from right to left?

ANSWER

Each column stands for a place value: units, tens, hundreds, … The algorithm adds each group separately and carries a newly formed ten to the next group on the left. Only digits of the same place value can be added together. Aligning from the right puts the units of both numbers in one column, the tens in the next column, and so on.

NCERT Archive

Example: 473 + 95.

Aligned from the right (473 + 095): 3 + 5 = 8, 7 + 9 = 16, 4 + 0 + 1 = 5 → **568** (correct)
Aligned from the left, the hundreds digit 4 would be added to the tens digit 9, 7 to the units digit 5, and so on → 1423 (wrong)

Aligning from the right makes digits of equal place value fall in the same column. Otherwise we would add, for example, hundreds to tens and get a wrong sum.

QUESTION 4

In Step 3, why cannot the value of carry be more than 1?

ANSWER

Every digit is at most 9. In Step 2 the largest possible column sum is 9 + 9 = 18, which is less than 20, so the first carry is 0 or 1.

Now suppose the carry coming into a column is 0 or 1. In Step 3 the largest possible column sum is 9 + 9 + 1 = 19. The carry to the next column is the number of tens in the column sum, and since 19 is less than 20, this number is at most 1. So the new carry is again 0 or 1.

Repeating this argument column after column shows that the carry is always 0 or 1.

A column sum is at most 9 + 9 + 1 = 19, which is less than 20, so the carry cannot be more than 1.

QUESTION 5

What happens if we do not include the fifth step in the algorithm above? Give examples where the algorithm will work correctly and where it will fail to work.

ANSWER

Step 5 writes the final carry to the left of the bottom row. Without it, a carry of 1 coming out of the leftmost column is lost.

✓ **Where the algorithm works correctly**

When the final carry is 0, nothing is lost. For example, 214 + 325: 4 + 5 = 9, 1 + 2 = 3, 2 + 3 = 5, giving 539, which is correct. Similarly 3847 + 5692 = 9539 is obtained correctly without Step 5.

NCERT Archive

✗ Where the algorithm fails

When the leftmost column gives a carry of 1, the answer is incomplete. For example, 473 + 695: units 3 + 5 = 8; tens 7 + 9 = 16 → write 6, carry 1; hundreds 4 + 6 + 1 = 11 → write 1, carry 1. Without Step 5 we stop at 168, but the correct answer is 1168.

Another example: 85 + 27 gives 12 without Step 5, whereas the correct sum is 112.

> **Without Step 5 the algorithm works only when the final carry is 0; whenever the sum has one more digit than the numbers added, the leading 1 is lost.**

NCERT Archive

11.2

CLASS 9 MATHS · GANITA MANJARI · CHAPTER 11

Exercise Set 11.2

Solutions · 3 questions

QUESTION 1

Suppose List 1 and List 2 are two lists of numbers in increasing order.

(i) Write an algorithm to find elements in List 1 that are not present in List 2.

(ii) Write an algorithm to find elements in List 2 that are not present in List 1.

ANSWER

(i) Elements of List 1 that are not in List 2

Algorithm to find the elements of List 1 not present in List 2

1. Start with an empty list only-in-list-1.
2. For each number x in List 1:
   – if x does not appear in List 2, add x to only-in-list-1.
3. Report only-in-list-1.

Example: List 1 = [1, 3, 5, 15, 25, 75, 125, 375] (the divisors of 375) and List 2 = [1, 3, 5, 11, 15, 25, 33, 55, 75, 165, 275, 825] (the divisors of 825). The numbers 1, 3, 5, 15, 25, 75 of List 1 appear in List 2, while 125 and 375 do not. So only-in-list-1 = [125, 375].

Since the lists are in increasing order, the output is also in increasing order. The order can also be used to avoid searching List 2 from the start each time: compare the current entries of the two lists; if they are equal, move ahead in both lists; if the List 1 entry is smaller, it is not in List 2, so add it and move ahead in List 1; otherwise move ahead in List 2.

(ii) Elements of List 2 that are not in List 1

Algorithm to find the elements of List 2 not present in List 1

1. Start with an empty list only-in-list-2.
2. For each number y in List 2:
   – if y does not appear in List 1, add y to only-in-list-2.
3. Report only-in-list-2.

NCERT Archive

For the same example, 11, 33, 55, 165, 275, 825 of List 2 are not in List 1, so only-in-list-2 = [11, 33, 55, 165, 275, 825].

Part (ii) is part (i) with the roles of List 1 and List 2 interchanged.

QUESTION 2

Describe an algorithm to compute the least common multiple (lcm) of two numbers.

ANSWER

A Algorithm from the definition

The lcm of m and n is the smallest number that is a multiple of both. One common multiple is m × n, so the lcm lies between 1 and m × n. We check the numbers in increasing order and stop at the first one that is a multiple of both.

Algorithm to find lcm(m, n)

1. For each number j in the sequence 1, 2, 3, …, m × n:
   – if m divides j and n divides j, report j as lcm(m, n) and stop.

Example: lcm(12, 18). None of 1, 2, …, 35 is divisible by both 12 and 18, but 36 is (36 = 12 × 3 = 18 × 2). So lcm(12, 18) = 36.

B A faster way using the gcd

If g = gcd(m, n), write m = ga and n = gb, where a and b have no common factor. Then lcm(m, n) = gab = (m × n) ÷ g. So we can first compute the gcd with the algorithm of Section 11.2.3 (or the improved Euclid's algorithm).

Algorithm to find lcm(m, n) using gcd

1. Compute g = gcd(m, n).
2. Report (m × n) ÷ g as lcm(m, n).

Example: gcd(375, 825) = 75 and 375 × 825 = 309375, so lcm(375, 825) = 309375 ÷ 75 = 4125. Check: 4125 = 375 × 11 = 825 × 5.

lcm(m, n) is the first j in 1, 2, …, m × n divisible by both m and n; equivalently, lcm(m, n) = (m × n) ÷ gcd(m, n).

NCERT Archive

QUESTION 3 *

Divisors occur in pairs. For instance, the divisors of 18 are (1, 18), (2, 9) and (3, 6).

(i) If we write out divisors in pairs, how many numbers do we have to examine between 1 and *n* to find all the divisors of *n*?

(ii) If we list out the divisors in pairs, will our gcd algorithm still work in the manner we have described?

ANSWER

(i) How many numbers must be examined?

If d divides n, then n ÷ d also divides n, so every divisor d has a partner n ÷ d. In each pair (d, n ÷ d) one number is at most √n, because if both were greater than √n their product would be greater than n. Hence it is enough to examine the numbers 1, 2, 3, ... up to √n; every divisor d found gives the pair (d, n ÷ d). If d = n ÷ d (when n is a perfect square), the number is written only once.

Example: for n = 18, √18 ≈ 4.2, so we examine only 1, 2, 3, 4. We get (1, 18), (2, 9), (3, 6), while 4 is not a divisor. For n = 375, √375 ≈ 19.4, so we examine only 19 numbers instead of 375: the pairs are (1, 375), (3, 125), (5, 75), (15, 25).

We only have to examine the numbers from 1 up to √n, instead of all the numbers up to n.

(ii) Does the gcd algorithm still work?

Not exactly as described. When divisors are written in pairs they come out as (1, 18), (2, 9), (3, 6): the first numbers of the pairs increase but their partners decrease, so the list [1, 18, 2, 9, 3, 6] is not in increasing order. Step 4 of the gcd algorithm only checks whether x appears in the other list, so it still works. But Step 5 reports the *rightmost* element of common-divisors as the gcd, and that is correct only when the list is in increasing order.

Example: gcd(12, 18). In pairs, divisors-of-12 = [1, 12, 2, 6, 3, 4] and divisors-of-18 = [1, 18, 2, 9, 3, 6]. Then common-divisors = [1, 2, 6, 3], whose rightmost element is 3, but gcd(12, 18) = 6.

The algorithm works only if the pairs are first arranged in increasing order (the small divisors, followed by their partners in reverse order), or if Step 5 reports the largest element of common-divisors instead of the rightmost one.

NCERT Archive

11.3

CLASS 9 MATHS · GANITA MANJARI · CHAPTER 11

Exercise Set 11.3

Solutions · 2 questions

QUESTION 1

How would our original algorithm change if we computed the divisors of *n* by examining the numbers from 1 to *n* in reverse order, from *n* down to 1?

ANSWER

The divisors-of-n algorithm itself is unchanged except that j runs through n, n − 1, …, 2, 1. Every divisor is still found, but now the new divisor is added at the end of a list that is built from the largest divisor down, so the list is in *decreasing* order.

For n = 18: 18 divides 18 → [18]; 17, …, 10 do not divide 18; 9 divides → [18, 9];
8, 7 do not; 6 divides → [18, 9, 6]; 5, 4 do not; 3 divides → [18, 9, 6, 3];
2 divides → [18, 9, 6, 3, 2]; 1 divides → [18, 9, 6, 3, 2, 1]

The amount of work is the same (n numbers are examined). But the gcd algorithm of Section 11.2.3 depends on the order: common-divisors is now also in decreasing order, so the gcd is the *leftmost* element, not the rightmost. (Step 5 would otherwise report 1.) In fact, the first x in divisors-of-m that also appears in divisors-of-n is already the gcd.

Example: divisors-of-375 = [375, 125, 75, 25, 15, 5, 3, 1] and divisors-of-825 = [825, 275, 165, 75, 55, 33, 25, 15, 11, 5, 3, 1]. Then common-divisors = [75, 25, 15, 5, 3, 1], and the leftmost element 75 is gcd(375, 825).

The list of divisors is produced in decreasing order, and Step 5 of the gcd algorithm must report the leftmost element of common-divisors.

NCERT Archive

QUESTION 2

What about the last algorithm described above? What happens when we look at common divisors starting from min(*m*, *n*) and work backwards to 1?

ANSWER

The last algorithm keeps most-recent-common-divisor and updates it every time a k divides both m and n. If k goes from min(m, n) down to 1, the *most recent* common divisor is always the smallest one found so far, so the update is wrong.

Example: m = 6, n = 12, min(6, 12) = 6.
k = 6 divides both → most-recent-common-divisor = 6; k = 5, 4 do not divide 6;
k = 3 divides both → 3; k = 2 divides both → 2; k = 1 divides both → 1.
The algorithm ends with 1, but gcd(6, 12) = 6.

Working backwards, the *first* k that divides both m and n is already the greatest common divisor, so no updating is needed and we can stop immediately:

1. For each k from min(m, n) down to 1:
   – if k divides both m and n, report k as gcd(m, n) and stop.

For m = 6, n = 12 this reports 6 at the very first check. The algorithm always stops, because 1 divides every number.

Backwards, the algorithm as written would always end with 1; it must instead report the first k found (and stop), which is then the gcd.

NCERT Archive

EOC

CLASS 9 MATHS · GANITA MANJARI · CHAPTER 11

End-of-Chapter Exercises

Solutions · 5 questions

QUESTION 1

Compute the following using the improved version of Euclid’s algorithm.

(i) gcd(375, 825)

(ii) gcd(51000, 81000)

(iii) gcd(1789287, 237656)

(iv) gcd(2587392, 157656)

ANSWER

Improved Euclid’s algorithm: (1) if m < n, reverse the numbers; (2) if n = 0, the answer is m; (3) otherwise reduce to gcd(n, m mod n).

(i) **gcd(375, 825)**

Step 1: 375 < 825, so reverse the numbers: gcd(825, 375)

825 = 2 × 375 + 75 → gcd(825, 375) reduces to gcd(375, 75)

375 = 5 × 75 + 0 → gcd(375, 75) reduces to gcd(75, 0)

Now n = 0, so Step 2 reports the answer m = 75.

gcd(375, 825) = 75

NCERT Archive

(ii) **gcd(51000, 81000)**

Step 1: 51000 < 81000, so reverse the numbers: gcd(81000, 51000)
81000 = 1 × 51000 + 30000 → gcd(81000, 51000) reduces to gcd(51000, 30000)
51000 = 1 × 30000 + 21000 → gcd(51000, 30000) reduces to gcd(30000, 21000)
30000 = 1 × 21000 + 9000 → gcd(30000, 21000) reduces to gcd(21000, 9000)
21000 = 2 × 9000 + 3000 → gcd(21000, 9000) reduces to gcd(9000, 3000)
9000 = 3 × 3000 + 0 → gcd(9000, 3000) reduces to gcd(3000, 0)
Now n = 0, so Step 2 reports the answer m = 3000.

gcd(51000, 81000) = 3000

(iii) **gcd(1789287, 237656)**

1789287 = 7 × 237656 + 125695 → gcd(1789287, 237656) reduces to gcd(237656, 125695)
237656 = 1 × 125695 + 111961 → gcd(237656, 125695) reduces to gcd(125695, 111961)
125695 = 1 × 111961 + 13734 → gcd(125695, 111961) reduces to gcd(111961, 13734)
111961 = 8 × 13734 + 2089 → gcd(111961, 13734) reduces to gcd(13734, 2089)
13734 = 6 × 2089 + 1200 → gcd(13734, 2089) reduces to gcd(2089, 1200)
2089 = 1 × 1200 + 889 → gcd(2089, 1200) reduces to gcd(1200, 889)
1200 = 1 × 889 + 311 → gcd(1200, 889) reduces to gcd(889, 311)
889 = 2 × 311 + 267 → gcd(889, 311) reduces to gcd(311, 267)
311 = 1 × 267 + 44 → gcd(311, 267) reduces to gcd(267, 44)
267 = 6 × 44 + 3 → gcd(267, 44) reduces to gcd(44, 3)
44 = 14 × 3 + 2 → gcd(44, 3) reduces to gcd(3, 2)
3 = 1 × 2 + 1 → gcd(3, 2) reduces to gcd(2, 1)
2 = 2 × 1 + 0 → gcd(2, 1) reduces to gcd(1, 0)
Now n = 0, so Step 2 reports the answer m = 1.

gcd(1789287, 237656) = 1

NCERT Archive

(iv) **gcd(2587392, 157656)**

2587392 = 16 × 157656 + 64896 → gcd(2587392, 157656) reduces to gcd(157656, 64896)
157656 = 2 × 64896 + 27864 → gcd(157656, 64896) reduces to gcd(64896, 27864)
64896 = 2 × 27864 + 9168 → gcd(64896, 27864) reduces to gcd(27864, 9168)
27864 = 3 × 9168 + 360 → gcd(27864, 9168) reduces to gcd(9168, 360)
9168 = 25 × 360 + 168 → gcd(9168, 360) reduces to gcd(360, 168)
360 = 2 × 168 + 24 → gcd(360, 168) reduces to gcd(168, 24)
168 = 7 × 24 + 0 → gcd(168, 24) reduces to gcd(24, 0)
Now n = 0, so Step 2 reports the answer m = 24.

gcd(2587392, 157656) = 24

QUESTION 2

Assume that *m* ≥ *n*. Verify that *d* divides *m* and *n* if and only if *d* divides both *n* and *m* mod *n*.

ANSWER

Let q be the quotient and r = m mod n the remainder when m is divided by n, so

m = q × n + r, where 0 ≤ r < n, so r = m − q × n

⇒ **If d divides m and n, then d divides n and r**

Write m = ad and n = bd. Then r = m − qn = ad − qbd = (a − qb)d, so d divides r. Also d divides n, by assumption.

⇐ **If d divides n and r, then d divides m and n**

Write n = xd and r = yd. Then m = qn + r = qxd + yd = (qx + y)d, so d divides m. Also d divides n, by assumption.

Hence d divides both m and n if and only if d divides both n and m mod n. So gcd(m, n) = gcd(n, m mod n), which justifies the reduction step.

NCERT Archive

QUESTION 3

Write an algorithm prime(*n*) to check if *n* is prime.

Hint: A prime number *p* has exactly two distinct factors, 1 and *p*. Can you make use of divisors(*n*) to write out prime(*n*)?

ANSWER

A number n is prime exactly when its list of divisors has two entries, 1 and n.

Algorithm prime(n)

1. Compute the list divisors(n) using the algorithm of Section 11.2.1.
2. If the list divisors(n) has exactly two elements, report “n is prime”.
3. Otherwise, report “n is not prime”.

Examples: divisors(7) = [1, 7] has two elements, so 7 is prime. divisors(18) = [1, 2, 3, 6, 9, 18] has six elements, so 18 is not prime. divisors(1) = [1] has only one element, so 1 is not prime.

prime(n) reports “prime” exactly when divisors(n) has exactly two elements.

QUESTION 4 *

Write an algorithm primedivisors(*n*) to compute the list of divisors of *n* that are prime numbers.

Hint: Compute divisors(*n*) and then filter out the primes in this list.

ANSWER

Algorithm primedivisors(n)

1. Compute the list divisors(n).
2. Start with an empty list prime-divisors.
3. For each number d in divisors(n):
   – if prime(d) reports “d is prime”, add d to the list prime-divisors.
4. Report prime-divisors.

Example: n = 60. divisors(60) = [1, 2, 3, 4, 5, 6, 10, 12, 15, 20, 30, 60]. Of these, only 2, 3 and 5 are prime (1 is not prime), so primedivisors(60) = [2, 3, 5]. For n = 18, divisors(18) = [1, 2, 3, 6, 9, 18] and primedivisors(18) = [2, 3].

primedivisors(n) is obtained by keeping only those divisors d of n for which prime(d) is true; the list stays in increasing order.

NCERT Archive

QUESTION 5 *

We can also find the gcd of two numbers by computing prime factorisation of both the numbers. Try to write an algorithm to compute the prime factorisation of a number.

(i) The prime factorisation of 180 is $2^2 \times 3^2 \times 5^1$. How would you represent this?

(ii) How would you compare the prime factorisations of two numbers?

ANSWER

A Algorithm for the prime factorisation of n

Algorithm to find the prime factorisation of n

1. Start with an empty list factors, and let remaining = n.
2. Let p = 2.
3. While remaining is greater than 1:
   - if p divides remaining, add p to the list factors and replace remaining by remaining ÷ p;
   - otherwise, increase p by 1.
4. Report the list factors.

Only primes can be added: before p is tried, all smaller numbers have been divided out as often as possible, so a composite p cannot divide what is left. Example: n = 180. p = 2: 180 → 90 → 45, factors = [2, 2]. p = 2 does not divide 45; p = 3: 45 → 15 → 5, factors = [2, 2, 3, 3]; p = 3 and p = 4 do not divide 5; p = 5: 5 → 1, factors = [2, 2, 3, 3, 5].

(i) Representing 180 = $2^2 \times 3^2 \times 5^1$

Write each prime together with its exponent, as a list of pairs in increasing order of the primes:

180 = [(2, 2), (3, 2), (5, 1)]

This is obtained from the list [2, 2, 3, 3, 5] by counting how many times each prime occurs. (Equivalently, keep two lists: primes [2, 3, 5] and exponents [2, 2, 1].)

(ii) Comparing two prime factorisations

Keep both lists of pairs in increasing order of the primes. For each prime that appears in both lists, take the *smaller* of the two exponents; a prime that appears in only one list is left out (its exponent in the other number is 0). The gcd is the product of these prime powers.

NCERT Archive

Example: 375 = 3 × 5³ = [(3, 1), (5, 3)] and 825 = 3 × 5² × 11 = [(3, 1), (5, 2), (11, 1)]. The primes 3 and 5 are common, with smaller exponents 1 and 2; 11 occurs only in 825.

$$\gcd(375, 825) = 3^1 \times 5^2 = 3 \times 25 = 75$$

The gcd is the product of the common primes, each raised to the smaller of its two exponents. This agrees with gcd(375, 825) = 75 found earlier.

NCERT Archive

IT

CLASS 9 MATHS · GANITA MANJARI · CHAPTER 11

In-Text Questions

Solutions · 4 questions

THINK AND REFLECT 11.1 · Q1

See if you can complete the argument about grouping by units, tens, hundreds, … to justify why the addition algorithm works.

ANSWER

Write each number by place value. For example, $473 = 3 + 7 \times 10 + 4 \times 100$ and $695 = 5 + 9 \times 10 + 6 \times 100$. Adding the groups separately:

$$473 + 695 = (3 + 5) + (7 + 9) \times 10 + (4 + 6) \times 100 = 8 + 16 \times 10 + 10 \times 100$$

When a group has 10 or more, 10 items of that group form 1 item of the next bigger group, so we keep the units digit and carry 1 to the next group: 16 tens = 6 tens + 1 hundred, so the hundreds group becomes 4 + 6 + 1 = 11 hundreds = 1 hundred + 1 thousand. Hence $473 + 695 = 1 \times 1000 + 1 \times 100 + 6 \times 10 + 8 = 1168$.

In general, let the numbers have digits a_i and b_i in the 10^i place ($i = 0$ for units). Let $c_0 = 0$, let $s_i = a_i + b_i + c_i$ be the column sum, and let $c_{i+1} = 1$ if $s_i \geq 10$ and 0 otherwise. The digit written is $d_i = s_i - 10c_{i+1}$. Then $a_i + b_i = d_i + 10c_{i+1} - c_i$, so

$$\Sigma\,(a_i + b_i)\,10^i = \Sigma\,d_i\,10^i + \Sigma\,c_{i+1}\,10^{i+1} - \Sigma\,c_i\,10^i$$
$$= \Sigma\,d_i\,10^i + (\text{the last carry}) \times 10^{n+1}$$

Here the carry terms cancel in pairs (the carry c_{i+1} formed in one column is counted as $10c_{i+1}$ there and subtracted again as c_{i+1} in the next column), leaving only the last carry, since $c_0 = 0$. The right-hand side is exactly the number whose digits d_i the algorithm writes down, with the final carry written on the left (Step 5).

Adding the units, tens, hundreds, … groups separately and carrying each newly formed ten to the next group gives exactly the sum of the two numbers.

NCERT Archive

THINK AND REFLECT 11.1 · Q2

How would you modify the algorithm to add two decimal fractions?

ANSWER

Algorithm to add two decimal fractions

1. Add zeros at the end of the number with fewer decimal places, so that both numbers have the same number of decimal places.
2. Write the numbers one below the other so that the decimal points are in one vertical line (digits of equal place value are in the same column).
3. Add the digits column by column from right to left, with carries, exactly as in Steps 2–4 of the algorithm; the decimal point is simply skipped.
4. Write the decimal point in the answer directly below the decimal points above.
5. If the final carry is 1, write 1 to the left of the bottom row.

Example: 12.75 + 8.6 = 12.75 + 8.60. Hundredths: 5 + 0 = 5. Tenths: 7 + 6 = 13 → write 3, carry 1. Units: 2 + 8 + 1 = 11 → write 1, carry 1. Tens: 1 + 0 + 1 = 2. Putting the decimal point below the others gives **21.35**. Check: 12.75 + 8.6 = 21.35.

Align the decimal points (padding with zeros if needed), add as before, and place the decimal point in the answer below the others.

THINK AND REFLECT 11.2 · Q1

Try to execute the algorithm to compute the divisors of 15, 135, and 775. How does the amount of work increase as the numbers grow?

ANSWER

A **divisors(15)**

Start with list-of-divisors = [ ]. Check j = 1, 2, …, 15. 1 divides 15 → [1]; 2 does not; 3 divides → [1, 3]; 4 does not; 5 divides → [1, 3, 5]; 6, 7, …, 14 do not; 15 divides → [1, 3, 5, 15].

divisors(15) = [1, 3, 5, 15]

NCERT Archive

B **divisors(135)**

Check j = 1, 2, ..., 135. The list grows as: [1] → [1, 3] → [1, 3, 5] → [1, 3, 5, 9] → [1, 3, 5, 9, 15] → [1, 3, 5, 9, 15, 27] → [1, 3, 5, 9, 15, 27, 45] → [1, 3, 5, 9, 15, 27, 45, 135]. All other numbers from 1 to 135 are skipped (they do not divide 135).

divisors(135) = [1, 3, 5, 9, 15, 27, 45, 135]

C **divisors(775)**

Check j = 1, 2, ..., 775. The list grows as: [1] → [1, 5] → [1, 5, 25] → [1, 5, 25, 31] → [1, 5, 25, 31, 155] → [1, 5, 25, 31, 155, 775].

divisors(775) = [1, 5, 25, 31, 155, 775]

The algorithm checks every j from 1 to n, so it makes n checks: 15, 135 and 775 checks for the three numbers. Going from 15 to 135 (about 9 times bigger) the work is about 9 times as much, and for 775 it is more than 50 times as much as for 15. When n gets one more digit (about 10 times larger), the work becomes about 10 times as much.

The work is proportional to the value of n, so it grows about 10 times for every extra digit, which is far slower than the addition algorithm, where each extra digit adds just one more column.

SECTION 11.3.4 · CHECK

Check that this would happen for any similar problem of the form gcd(2*k* + 1, 2). (Using the improved algorithm, gcd(99, 2) needed only two reduction steps.)

ANSWER

Take any natural number k and apply the improved Euclid's algorithm to gcd(2k + 1, 2). Here 2k + 1 is odd, so it leaves remainder 1 when divided by 2: (2k + 1) mod 2 = 1.

gcd(2k + 1, 2): 2k + 1 ≥ 2 (for k ≥ 1), so no reversal; Step 3 → gcd(2, (2k + 1) mod 2) = gcd(2, 1)

gcd(2, 1): 2 mod 1 = 0, so Step 3 → gcd(1, 0)

gcd(1, 0): n = 0, so Step 2 reports the answer m = 1

NCERT Archive

Example: k = 49 gives gcd(99, 2) → gcd(2, 1) → gcd(1, 0) = 1. For k = 0, gcd(1, 2) is first reversed to gcd(2, 1) and then continues in the same way.

For every k, exactly two reduction steps are needed, so gcd(2k + 1, 2) = 1, however large k is.

NCERT Archive