Rozwiązania testu wiedzy algorytmicznej

XVIII OIJ, zawody I stopnia, tura testowa
26 października 2023

Olimpiada Informatyczna Juniorów

1. **Rozważmy poniższą funkcję:**

wersja C++

```
int f(int x, int y, int z) {
  return y * z + x;
}
```

wersja Python

```
def f(x, y, z):
  return y * z + x
```

Jaki będzie wynik wywołania `f(2, 3, 4)`**?**

14

Rozwiązanie: Funkcja przyjmuje parametry w kolejności (x, y, z) (wskazuje na to definicja funkcji podana w pierwszym wierszu kodu) i zwraca wartość wyrażenia $y \cdot z + x$. Dla podanych wartości $x \leftarrow 2$, $y \leftarrow 3$, $z \leftarrow 4$ jest to $3 \cdot 4 + 2 = 14$.

2. **Rozważmy poniższą funkcję:**

wersja C++

```
string f(string s) {
  string wynik;
  for (char c : s) {
    if (c == 'a') continue;
    wynik.push_back(c);
  }
  return wynik;
}
```

wersja Python

```
def f(s):
  wynik = ''
  for c in s:
    if c == 'a': continue
    wynik += c
  return wynik
```

Jaki będzie wynik wywołania `f("bajtazar")`**?**

bjtzr

Rozwiązanie: Funkcja przegląda kolejne znaki napisu podanego jako parametr (od lewej do prawej). Znak `'a'` jest ignorowany: instrukcja `continue` w obu językach powoduje przerwanie bieżącej iteracji pętli i wykonanie następnej iteracji od początku, już z nową wartością zmiennej iterującej (w tym przypadku zmiennej `c`). Wszystkie inne znaki zostaną doklejone na końcu wyniku. Ostatecznie uzyskamy napis jak w argumencie funkcji po pominięciu wszystkich wystąpień litery `'a'`.

Olimpiada finansowana jest ze środków Ministerstwa Edukacji i Nauki w ramach zadania publicznego „Organizacja i przeprowadzenie olimpiad i turniejów w latach szkolnych 2022/2023, 2023/2024, 2024/2025".

3. **Rozważmy poniższą funkcję:**

wersja C++

```
string f(string s) {
  string wynik;
  int ile = 1;
  for (char c : s) {
    for (int i = 0; i < ile; i++)
      wynik.push_back(c);
    ile++;
  }
  return wynik;
}
```

wersja Python

```
def f(s):
  wynik = ''
  ile = 1
  for c in s:
    for i in range(ile):
      wynik += c
    ile += 1
  return wynik
```

Jaki będzie wynik wywołania `f("test")`**?**

teesssttttt

Rozwiązanie: Funkcja przetwarza kolejne znaki napisu podanego na wejściu jako parametr funkcji i w każdym kroku dokleja na końcu wyniku kolejno jedną, dwie, trzy, ..., k kopii znaku odpowiednio w pierwszej, drugiej, trzeciej, ..., k-tej iteracji.

4. **Zaznacz wszystkie liczby parzyste.**

- ☐ 101010101_2 (liczba jest zapisana w systemie dwójkowym)
- ☒ 10^9
- ☐ $7^{10} \cdot 5^{11}$
- ☒ $1+2+3+\ldots+100$

Rozwiązanie: Liczby kończące się na 1 w systemie dwójkowym są nieparzyste, stąd pierwsza odpowiedź jest błędna. Liczba 10^9 (miliard) jest oczywiście parzysta, więc druga odpowiedź jest poprawna. Liczby 7 oraz 5 są nieparzyste, stąd iloczyn dowolnie wielu z nich jest liczbą nieparzystą, więc trzecia odpowiedź jest błędna. W sumie $1+2+3+\ldots+100$ jest pięćdziesiąt liczb nieparzystych i pięćdziesiąt parzystych. Suma dwóch liczb nieparzystych jest liczbą parzystą, a suma dowolnie wielu liczb parzystych jest liczbą parzystą. Z tego wynika, że ostatnia odpowiedź jest prawidłowa.

5. **Rozważmy poniższą funkcję:**

wersja C++

```
int f(int n) {
  int wynik = 0;
  for (int i = 0; i < n; i++)
    for (int j = 0; j < n; j++)
      for (int k = 0; k < n; k++)
        wynik++;
  return wynik;
}
```

wersja Python

```
def f(n):
  wynik = 0
  for i in range(n):
    for j in range(n):
      for k in range(n):
        wynik += 1
  return wynik
```

Jaki powinien być argument `n`**, aby** `f(n)` $= 64$**?**

4

Rozwiązanie: Funkcja (w bardzo nieoptymalny sposób) oblicza sześcian liczby podanej jako argument. Jeśli bowiem do funkcji podano n jako argument, na dokładnie n sposobów można wybrać wartość zmiennej i, a więc na n^2 sposobów można wybrać wartość pary zmiennych (i, j), czyli na n^3 sposobów wartość trójki (i, j, k). Każda taka trójka spowoduje powiększenie wyniku o 1. Wiedząc, że wynikiem funkcji jest 64, aby obliczyć argument, z jakim funkcja została uruchomiona, wystarczy obliczyć pierwiastek sześcienny z tej liczby. Odpowiedzią jest więc $\sqrt[3]{64} = 4$. Wartość tego pierwiastka można zgadnąć, próbując kolejne wartości potencjalnych argumentów lub przeprowadzając na kartce technikę wyszukiwania binarnego.

Olimpiada finansowana jest ze środków Ministerstwa Edukacji i Nauki w ramach zadania publicznego „Organizacja i przeprowadzenie olimpiad i turniejów w latach szkolnych 2022/2023, 2023/2024, 2024/2025".

6. **Bajtek ma zero dukatów i spotkał Alladyna. Jeżeli powie mu `A`, otrzyma jednego dukata, jeżeli zaś powie mu `B`, otrzyma drugie tyle dukatów ile już posiada. Bajtek może powiedzieć wiele zaklęć po kolei: przykładowo, jeżeli powie `AAABA`, to będzie miał 7 dukatów ($0 \to 1 \to 2 \to 3 \to 6 \to 7$). Wymień jedną literę w słowie `AAABA` na inną, żeby Bajtek zamiast siedmiu, uzyskał 9 dukatów. Jako odpowiedź podaj nowy, zmieniony, pięcioliterowy napis.**

AABBA

Rozwiązanie: Łatwo przekonać się (na przykład sprawdzając wszystkie pięć możliwości wymiany pojedynczego znaku na przeciwny), że jedynym rozwiązaniem jest `AABBA` ($0 \to 1 \to 2 \to 4 \to 8 \to 9$).

7. **Rozważmy poniższą funkcję rekurencyjną:**

wersja C++

```
int f(int n) {
  if (n <= 9) return n;
  int s = 0;
  while (n > 0) {
    s += n % 10;
    n /= 10;
  }
  return f(s);
}
```

wersja Python

```
def f(n):
  if n <= 9: return n
  s = 0
  while n > 0:
    s += n % 10
    n //= 10
  return f(s)
```

Ile jest różnych parametrów `n` (liczb naturalnych) z przedziału od 1 do $10\,000$ włącznie, dla których `f(n)` = 1?

1112

Rozwiązanie: Funkcja $f(\cdot)$ tak naprawdę (dla liczb nieujemnych) zwraca resztę z dzielenia argumentu przez 9 (lub wartość 9, gdy reszta wynosi 0). Wynika to z reguły podzielności przez 9: dowolna liczba daje taką samą resztę z dzielenia przez 9 co jej suma cyfr. A zatem pytanie upraszcza się do ustalenia liczby liczb naturalnych od 1 do $10\,000$, które dają resztę 1 przy dzieleniu przez 9. Dla liczb od 1 do $9\,999$ jest $\frac{9\,999}{9} = 1\,111$ liczb dających każdą z reszt $0, 1, \ldots, 8$, a liczba $10\,000$ daje resztę 1, stąd odpowiedzią jest $1\,111 + 1 = 1\,112$.

8. **Rozważmy poniższą funkcję:**

wersja C++

```
string f(string s) {
  string wynik;
  for (char c : s) {
    if (c == 'z')
      wynik.push_back('a');
    else
      wynik.push_back(c + 1);
  }
  return wynik;
}
```

wersja Python

```
def f(s):
  wynik = ''
  for c in s:
    if c == 'z':
      wynik += 'a'
    else:
      wynik += chr(ord(c) + 1)
  return wynik
```

Jaki był argument `s`, jeżeli wywołanie `f(s)` zwróciło `"abebojf"`?

zadanie

Rozwiązanie: Funkcja $f(\cdot)$ wykonuje szyfr Cezara z przesunięciem o 1 (litera `'z'` zmieniana jest na `'a'`, zaś każda inna litera przesuwana jest na następną w alfabecie angielskim, zgodnie z kolejnością znaków w tablicy ASCII). Aby odszyfrować napis wystarczy więc cofnąć każdy znak o jedną pozycję w alfabecie angielskim (pamiętając, żeby literę `'a'` przesunąć na `'z'`).

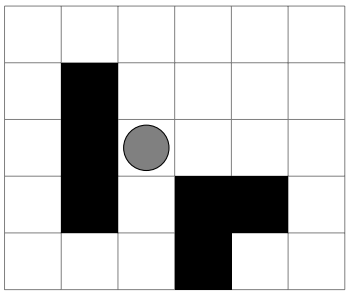
Olimpiada finansowana jest ze środków Ministerstwa Edukacji i Nauki w ramach zadania publicznego „Organizacja i przeprowadzenie olimpiad i turniejów w latach szkolnych 2022/2023, 2023/2024, 2024/2025".

9. **Ile jest liczb naturalnych z przedziału od 1 do 200 włącznie, w których w zapisie w systemie dwójkowym (bez zer wiodących) występuje przynajmniej pięć zer z rzędu?**

11

Rozwiązanie: Gdyby te pięć zer miało być na końcu zapisu dwójkowego, uzyskalibyśmy liczby podzielne przez 32. Jest ich $\lfloor \frac{200}{32} \rfloor = 6$ (największa z nich to $6 \cdot 32 = 192$). Gdyby jednak zapis dwójkowy kończył się na pięć zer i później jedynkę, mamy nowe trzy liczby (największa z nich to 193). Należy jeszcze doliczyć liczby $10000010_2 = 130$ oraz $10000011_2 = 131$. Razem jest $6 + 3 + 2 = 11$ liczb spełniających warunki zadania.

10. **Dana jest plansza złożona z czarnych i białych kwadratów. Pionek zaczyna od pola oznaczonego kółkiem i w jednym ruchu może przesunąć się jedynie na pole białe w jednym z czterech podstawowych kierunków: o jedno pole w górę, dół, lewo lub prawo. Ile jest pól, na które pionek może dojść w co najwyżej czterech ruchach? Wliczamy również pole początkowe.**



18

Rozwiązanie: Wystarczy dla każdego pola ustalić jego odległość (w sensie liczby ruchów) od pola początkowego. Można też pominąć pola z odległością większą niż 4.

4	3	2	3	4	
	■	1	2	3	4
	■	0	1	2	3
	■	1	■	■	4
4	3	2	■		

Olimpiada finansowana jest ze środków Ministerstwa Edukacji i Nauki w ramach zadania publicznego „Organizacja i przeprowadzenie olimpiad i turniejów w latach szkolnych 2022/2023, 2023/2024, 2024/2025".

11. **Rozważmy poniższą funkcję:**

wersja C++

```
int f(int n) {
  int wynik = 0;
  int mnoznik = 1;
  while (n > 0) {
    wynik += mnoznik * (n % 2);
    n /= 2;
    mnoznik++;
  }
  return wynik;
}
```

wersja Python

```
def f(n):
  wynik = 0
  mnoznik = 1
  while n > 0:
    wynik += mnoznik * (n % 2)
    n //= 2
    mnoznik += 1
  return wynik
```

Podaj najmniejszą dodatnią wartość parametru `n`**, dla której wywołanie** `f(n)` **zwróci** 17**.**

55

Rozwiązanie: Funkcja $f(\cdot)$ znajduje zapis dwójkowy liczby n i dodaje do wyniku $1, 2, 3, \ldots, k$, gdy pierwszy, drugi, trzeci, ..., k-ty bit od końca jest jedynką. Naszym celem jest więc znaleźć najmniejszą liczbę, która ma zapalone bity w taki sposób, aby uzyskać sumę 17. Najpierw zauważamy, że ponieważ $1+2+3+4+5 = 15 < 17$, potrzebujemy co najmniej sześciu bitów. Szósty bit od końca będzie wart 6 i musimy go zapalić. Z pozostałych pięciu bitów musimy uzbierać sumę 11. Można to osiągnąć na dwa sposoby: albo $1+2+3+5$ albo $2+4+5$. Pierwsza opcja odpowiada mniejszej liczbie, a zatem liczba której poszukujemy to $110111_2 = 55$.

Olimpiada finansowana jest ze środków Ministerstwa Edukacji i Nauki w ramach zadania publicznego „Organizacja i przeprowadzenie olimpiad i turniejów w latach szkolnych 2022/2023, 2023/2024, 2024/2025".

12. **Rozważmy poniższą funkcję:**

wersja C++

```
bool f(string s) {
  vector<bool> zaznaczone(15, false);
  int pozycja = 0;
  zaznaczone[pozycja] = true;
  for (char x : s) {
    if (x == 'a') pozycja += 5;
    else if (x == 'b') pozycja -= 2;
    else return false;
    if ((pozycja < 0) || (pozycja > 14))
      return false;
    if (zaznaczone[pozycja]) return false;
    zaznaczone[pozycja] = true;
  }
  for (int i = 0; i < 15; i++)
    if (!zaznaczone[i]) return false;
  return true;
}
```

wersja Python

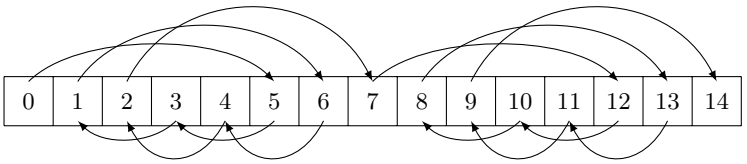
```
def f(s):
  zaznaczone = [False] * 15
  pozycja = 0
  zaznaczone[pozycja] = True
  for x in s:
    if x == 'a': pozycja += 5
    elif x == 'b': pozycja -= 2
    else: return False
    if (pozycja < 0) or (pozycja > 14):
      return False
    if zaznaczone[pozycja]: return False
    zaznaczone[pozycja] = True
  for i in range(15):
    if not zaznaczone[i]: return False
  return True
```

Podaj wartość parametru `s`, dla którego wywołanie `f(s)` zwróci `true`/`True`.

abbabbaabbabba

Rozwiązanie: Aby funkcja $f(\cdot)$ zwróciła prawdę, musimy rozwiązać ciekawą zagadkę. Rozważmy planszę o wymiarach 1×15, z polami ponumerowanymi $0, 1, 2, \ldots, 14$. Na początku mamy pionek na pozycji numer 0. W jednym ruchu możemy przesunąć pionek o pięć pól do przodu (wstawiając w napisie `s` literę `'a'`) lub o dwa pola do tyłu (wstawiając literę `'b'`). Naszym celem jest odwiedzić wszystkie pola planszy (funkcja sprawdza na końcu czy chociaż jedna komórka tablicy `zaznaczone[]` jest fałszem), ale każde dokładnie raz (funkcja sprawdza za każdym razem czy odwiedzane pole nie było wcześniej zaznaczone). Niedopuszczalne jest wyjście poza planszę (funkcja sprawdza zakres pozycji) oraz używanie znaków innych niż `'a'` lub `'b'`.

Na początku można wykonać co najwyżej dwa ruchy do przodu. Jeżeli wykonamy dokładnie dwa, będziemy na polu numer 10 (parzystym) i nie będziemy w stanie odwiedzić pola numer 1 nie przechodząc dwukrotnie po wcześniej odwiedzonym polu. Zatem pierwszy ruch musi być do przodu, a kolejny do tyłu. Metodą prób i błędów ustalamy (chcąc odwiedzić pole numer 1), że kolejny ruch również powinien być do tyłu. Następnie ruch musi być do przodu i znowu dwa do tyłu oraz jeden do przodu i w ten sposób odwiedziliśmy połowę planszy, kończąc w polu numer 7. Powtarzając to samo rozumowanie odwiedzimy drugą połowę planszy i zakończymy w polu 14. Jedyne możliwe rozwiązanie jest przedstawione na rysunku poniżej:



Olimpiada finansowana jest ze środków Ministerstwa Edukacji i Nauki w ramach zadania publicznego „Organizacja i przeprowadzenie olimpiad i turniejów w latach szkolnych 2022/2023, 2023/2024, 2024/2025".

13. **Rozważmy poniższą funkcję:**

wersja C++

```
string f(string s) {
  string wynik;
  for (char x : s) {
    if (x == 'b') wynik += "00";
    else if (x == 'd') wynik += "01";
    else if (x == 'e') wynik += "100";
    else if (x == 'o') wynik += "101";
    else if (x == 'r') wynik += "110";
    else if (x == 'z') wynik += "111";
    else wynik += "2";
  }
  return wynik;
}
```

wersja Python

```
def f(s):
  wynik = ''
  for x in s:
    if x == 'b': wynik += '00'
    elif x == 'd': wynik += '01'
    elif x == 'e': wynik += '100'
    elif x == 'o': wynik += '101'
    elif x == 'r': wynik += '110'
    elif x == 'z': wynik += '111'
    else: wynik = '2'
  return wynik
```

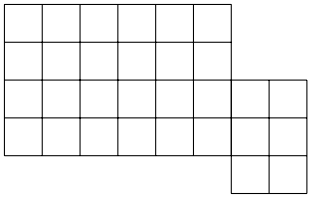
Podaj wartość parametru `s`**, dla którego wywołanie** `f(s) = "0110100110111100"`**.**

dobrze

Rozwiązanie: Funkcja zamienia podany napis na ciąg bitowy, dla uproszczenia zapisywany jako napis złożony z zer i jedynek. Zastosowano tutaj kodowanie bezprefiksowe (kod każdej litery nie jest początkowym fragmentem innego kodu), stąd dowolny napis można rozkodować jednoznacznie metodą zachłanną, poszukując najkrótszego początkowego fragmentu, który jest dopuszczalnym kodem (w tym przypadku `"01"`, reprezentujące literę `'d'`). Analogicznie rozkodowujemy resztę napisu.

Olimpiada finansowana jest ze środków Ministerstwa Edukacji i Nauki w ramach zadania publicznego „Organizacja i przeprowadzenie olimpiad i turniejów w latach szkolnych 2022/2023, 2023/2024, 2024/2025".

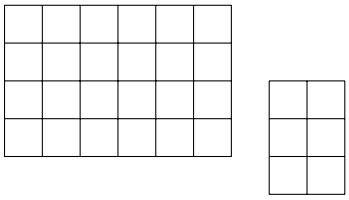
14. **Ile jest prostokątów na poniższym rysunku? Interesują nas prostokąty o dowolnych wymiarach, ale z bokami jedynie wzdłuż linii kratek. Przykładowo, na kratownicy o wymiarach 2×2 kwadraty jednostkowe znajduje się dziewięć prostokątów: cztery kwadraty 1×1, dwa prostokąty 2×1, dwa prostokąty 1×2 oraz jeden kwadrat 2×2.**



264

Rozwiązanie: Podzielmy najpierw rysunek na dwie części i policzmy z osobna:

- liczbę prostokątów w lewej części,
- liczbę prostokątów w prawej części,
- liczbę prostokątów przechodzących przez obie części.



Sposób 1. Rozważmy problem jednowymiarowy: na pasku $1 \times N$ jesteśmy w stanie wybrać spójny fragment od pewnego do pewnego kwadratu jednostkowego na $N + (N-1) + \ldots + 1 = \binom{N+1}{2}$ sposobów (jest $N+1$ linii pionowych definiujących N kwadratów jednostkowych). Jeżeli wrócimy teraz do problemu dwuwymiarowego to prawdopodobnie zauważymy, że wybór prostokąta to tak naprawdę wybór paska poziomego (zakresu indeksów współrzędnych poziomych, które będzie obejmował) oraz paska pionowego (zakresu współrzędnych pionowych). Każda możliwość pozioma z każdą możliwością pionową daje unikalny prostokąt.
Stąd prostokątów w lewej części jest $\binom{7}{2} \cdot \binom{5}{2} = 21 \cdot 10 = 210$. Analogicznie, prostokątów w prawej części jest $\binom{4}{2} \cdot \binom{3}{2} = 6 \cdot 3 = 18$.
Pozostaje zliczyć prostokąty przechodzące między częściami: wyboru zakresu współrzędnych pionowych możemy dokonać na trzy sposoby, początek zakresu współrzędnych x'owych na sześć sposobów, a koniec tego zakresu na dwa sposoby. Daje to nam $3 \cdot 6 \cdot 2 = 36$ różnych prostokątów.
Sumując uzyskane wartości otrzymujemy rozwiązanie zadania: są $210 + 18 + 36 = 264$ prostokąty.
Sposób 2. Dla każdego kwadratu jednostkowego ustalamy ile jest prostokątów, które mają zaczepiony lewy górny róg w tym kwadracie jednostkowym. Następnie sumujemy wszystkie rozważone wartości.

24	20	16	12	8	4		
18	15	12	9	6	3		
16	14	12	10	8	6	6	3
8	7	6	5	4	3	4	2
						2	1

Olimpiada finansowana jest ze środków Ministerstwa Edukacji i Nauki w ramach zadania publicznego „Organizacja i przeprowadzenie olimpiad i turniejów w latach szkolnych 2022/2023, 2023/2024, 2024/2025".

15. **Ile jest liczb naturalnych z przedziału od 1 do $3\,999$ włącznie, które zapisane w systemie rzymskim nie mają dwóch takich samych znaków obok siebie? Dla przypomnienia: w systemie rzymskim występują cyfry `I`, `V`, `X`, `L`, `C`, `D`, `M` oznaczające kolejno: 1, 5, 10, 50, 100, 500 oraz $1\,000$. Przykładowo, zapis liczby 4 to `IV`, zaś zapis liczby 99 to `XCIX`.**

431

Rozwiązanie: W rozwiązaniu poniżej zakładamy, że liczba 0 ma pusty zapis (bez żadnych znaków) w systemie rzymskim. Wśród liczb od 0 do 9 mamy sześć, które nie wymagają dwóch znaków obok siebie: $0, 1, 4, 5, 6, 9$ (odpowiednio: pusty zapis, `I`, `IV`, `V`, `VI`, `IX`). Podobnie jest w przypadku pełnych dziesiątek: z pełnych dziesiątek możemy wybrać $0, 10, 40, 50, 60, 90$ (odpowiednio: pusty zapis, `X`, `XL`, `L`, `LX`, `XC`). Analogicznie z cyfrą setek w liczbie arabskiej: tylko 0, 100, 400, 500, 600 oraz 900 nie zawierają dwóch jednakowych znaków w zapisie rzymskim (odpowiednio: pusty zapis, `C`, `CD`, `D`, `DC`, `CM`). Cyfrę tysięcy można wybrać albo 0 albo 1 (w przeciwnym wypadku powtórzą się litery `M`).
Zauważmy teraz, że konwersja liczby arabskiej na rzymską przebiega poprzez sklejenie zapisów odpowiednich cyfr arabskich. Skoro tak to każdą z cyfr: jedności, dziesiątek i setek możemy wybrać na sześć sposobów, a cyfrę tysięcy na dwa sposoby. Daje to $6^3 \cdot 2 = 216 \cdot 2 = 432$ różne liczby. W ten sposób jednak policzyliśmy liczbę 0, a w pytaniu chodziło o zakres liczb od 1 do $3\,999$, stąd ostateczna odpowiedź to $432 - 1 = 431$.

16. **Rozważmy poniższą funkcję:**

wersja C++

```
int f(int n) {
  n %= 7;
  n %= 5;
  n %= 3;
  return n;
}
```

wersja Python

```
def f(n):
  n %= 7
  n %= 5
  n %= 3
  return n
```

Wskaż najmniejszą naturalną wartość parametru `n` większą niż 2, dla której `f(n)` = 2.

9

Rozwiązanie: Niech n oznacza początkową liczbę, n_1 oznacza wartość liczby po wykonaniu operacji reszty z dzielenia przez 7, a n_2 oznacza wartość liczby po wykonaniu dwóch pierwszych operacji.
Możliwe jest rozwiązanie zadania „od końca". Chcemy, aby $n_2 \equiv 2 \pmod 3$. Ale $n_2 = n_1 \bmod 5$, stąd $n_1 \bmod 5 \equiv 2 \pmod 3$. Jest pięć możliwych wyników działania $n_1 \bmod 5$, są to: $0, 1, 2, 3, 4$. Jedynie reszta 2 przy dzieleniu przez 5 spełnia warunek $n_2 \equiv 2 \pmod 3$. Wnioskując analogicznie dla n oraz n_1, dojdziemy do wniosku, że dane w tym zadaniu zostały tak dobrane, że musi zachodzić $n \equiv 2 \pmod 7$. Skoro nie możemy dobrać $n = 2$, następnym kandydatem jest $n = 9$.

Olimpiada finansowana jest ze środków Ministerstwa Edukacji i Nauki w ramach zadania publicznego „Organizacja i przeprowadzenie olimpiad i turniejów w latach szkolnych 2022/2023, 2023/2024, 2024/2025".

17. **Rozważmy poniższą funkcję:**

wersja C++

```
int f(int n, string s) {
  for (char x : s) {
    if (x == 'a') n *= 2;
    if (x == 'b') n++;
    if (x == 'c') n /= 2;
  }
  return n;
}
```

wersja Python

```
def f(n, s):
  for x in s:
    if x == 'a': n *= 2
    if x == 'b': n += 1
    if x == 'c': n //= 2
  return n
```

Podaj najkrótszy możliwy napis `s`**, dla którego wywołanie** `f(30, s)` **zwróci** 29**.**

ccaab

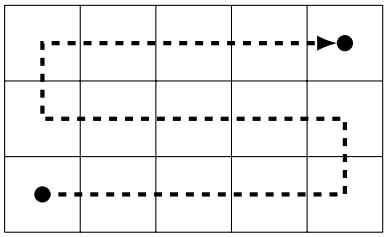
Rozwiązanie: Zaczynamy z $n = 30$ i naszym celem jest uzyskać $n = 29$ w jak najmniejszej liczbie ruchów. Możliwe jest wykonanie następujących ruchów:

- $n \leftarrow 2 \cdot n$ (litera `'a'`),
- $n \leftarrow n + 1$ (litera `'b'`),
- $n \leftarrow \lfloor \frac{n}{2} \rfloor$ (litera `'c'`),

Jeżeli chcemy uzyskać mniejszą liczbę, konieczne jest zastosowanie dzielenia przez 2 co najmniej jeden raz. Po jednym dzieleniu uzyskamy 15, a po dwóch: 7. Z wartości 15 potrzebowalibyśmy użyć operacji `'b'` aż czternaście razy. Gdyby jednak próbować z 7, moglibyśmy dotrzeć do 29 używając ciągu operacji `"aab"`. Razem z dwoma pierwszymi operacjami `'c'` daje to zaledwie pięć ruchów. Pozostaje jedynie przekonać się, że inny ciąg operacji nie jest krótszy: próba początkowego powiększania liczby n wydaje się bezcelowa (wymusi to większą liczbę operacji `'c'`), a większa liczba operacji `'c'` na początku prowadzi do dłuższych ciągów operacji.
Ostatecznie więc optymalnie jest wykonać ciąg `"ccaab"`, prowadzący do $30 \to 15 \to 7 \to 14 \to 28 \to 29$.

Olimpiada finansowana jest ze środków Ministerstwa Edukacji i Nauki w ramach zadania publicznego „Organizacja i przeprowadzenie olimpiad i turniejów w latach szkolnych 2022/2023, 2023/2024, 2024/2025".

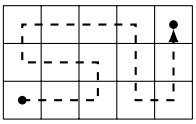
18. **Ile jest różnych ścieżek z lewego dolnego rogu do prawego górnego rogu poniższej planszy, które przechodzą przez każde z piętnastu pól dokładnie jeden raz? W jednym ruchu można przesunąć się o jedno pole w jednym z czterech podstawowych kierunków (góra, dół, lewo, prawo). Jedną z poprawnych ścieżek, którą również należy policzyć, pokazano na rysunku poniżej.**



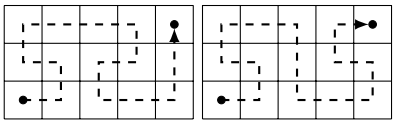
8

Rozwiązanie: Kluczowe jest systematyczne przeglądanie możliwości, aby mieć pewność, że każdą ścieżkę policzymy dokładnie raz.
Rozpatrzmy osobno przypadki w zależności od liczby ruchów $\rightarrow$ na początku ścieżki.
Jeżeli na początku zrobimy:

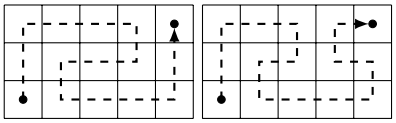
- cztery ruchy $\rightarrow$ to dalsza część ścieżki musi już wyglądać jak na obrazku z zadania (tylko jedna możliwość),
- trzy ruchy $\rightarrow$ i potem $\uparrow$ to prawe dolne pole planszy zostanie odcięte od reszty, stąd nie uda się zakończyć ścieżki w prawym górnym rogu i odwiedzić wszystkich pól (zero możliwości),
- dwa ruchy $\rightarrow$ i potem $\uparrow$: jest jedna ścieżka:



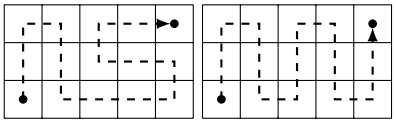
- jeden ruch $\rightarrow$ i potem $\uparrow$: musimy kontynuować $\leftarrow, \uparrow, \rightarrow, \rightarrow$, inaczej skończymy z niepoprawną ścieżką. Dalej możemy kontynuować na dwa sposoby:



- zero ruchów $\rightarrow$ i od razu $\uparrow$: musimy kontynuować do samej góry i potem w prawo $(\uparrow, \rightarrow)$.
  (a) jeżeli następny ruch to $\rightarrow$ to są dwie możliwości:



  (b) jeżeli zaś następny ruch to $\downarrow$ to następne ruchy muszą być $\downarrow, \rightarrow$. Wtedy sytuację również można dopełnić na dwa sposoby:



Ostatecznie, z przypadków wyszło nam $1 + 0 + 1 + 2 + 4 = 8$ sposobów.

Olimpiada finansowana jest ze środków Ministerstwa Edukacji i Nauki w ramach zadania publicznego „Organizacja i przeprowadzenie olimpiad i turniejów w latach szkolnych 2022/2023, 2023/2024, 2024/2025".

19. **Rozważmy poniższe funkcje:**

wersja C++

```
int f(int n) {
  if (n == 0) return 0;
  return f(n / 10) + n % 10;
}

int g(int n) {
  if (n == 0) return 0;
  return g(n - 1) + f(n);
}
```

wersja Python

```
def f(n):
  if n == 0: return 0
  return f(n // 10) + n % 10

def g(n):
  if n == 0: return 0
  return g(n - 1) + f(n)
```

Jaki jest wynik wywołania `g(200)`**?**

1902

Rozwiązanie: Funkcja $g(n)$ zwraca sumę $f(1) + f(2) + \ldots + f(n)$, zaś funkcja $f(n)$ zwraca sumę cyfr liczby n w zapisie dziesiętnym. Ostatecznie więc mamy do obliczenia sumę cyfr liczb $1, 2, \ldots, 200$.
Suma cyfr liczb od 1 do 9 włącznie wynosi $1 + 2 + \ldots + 9 = 45$. Suma cyfr liczb od 10 do 19 włącznie wynosi o dziesięć więcej $(45 + 10)$. Podobnie, suma cyfr liczb od 20 do 29 wynosi $45 + 20$. A zatem: suma cyfr liczb od 1 do 99 wynosi $45 \cdot 10 + 10 + 20 + 30 + \ldots + 90 = 450 + 450 = 900$. Analogicznie, suma cyfr liczb od 100 do 199 włącznie wynosi $900 + 100 = 1\,000$ (o sto więcej niż suma cyfr liczb od 1 do 99). Ostatecznie należało podać sumę cyfr od 1 do 200 czyli $900 + 1\,000 + 2 = 1\,902$.

20. **Jaka jest największa liczba naturalna o sumie cyfr 12 i wszystkich cyfrach różnych?**

63210

Rozwiązanie: Liczba, której szukamy nie może się składać z sześciu cyfr lub więcej. Istotnie, sześć najmniejszych cyfr to $0, 1, 2, 3, 4, 5$, a ich suma to $0+1+2+3+4+5 = 15 > 12$. Stąd ograniczamy się do szukania liczb co najwyżej pięciocyfrowych. Aby liczba była jak największa, jej najbardziej znacząca cyfra powinna być jak największa: jeżeli pozostałe cyfry to będą $0, 1, 2, 3$, możliwe jest postawienie cyfry 6 jako najbardziej znaczącej. Ustawiając cyfry od największej do najmniejszej, otrzymujemy rozwiązanie: $63\,210$.

Olimpiada finansowana jest ze środków Ministerstwa Edukacji i Nauki w ramach zadania publicznego „Organizacja i przeprowadzenie olimpiad i turniejów w latach szkolnych 2022/2023, 2023/2024, 2024/2025".