Zadanie Abc – LOGIA 17 (2016/17), etap 3

Treść zadania

Jaś przygotował naszyjnik dla Małgosi. Małgosia poprosiła go o usunięcie niektórych koralików tak, aby wszystkie niebieskie koraliki były przed czerwonymi i zielonymi, a wszystkie czerwone – przed zielonymi. Pozostałe koraliki mogą występować w dowolnych miejscach.

Napisz jednoparametrową funkcję **abc**, której wynikiem jest minimalna liczba koralików do usunięcia. Parametrem funkcji jest niepuste słowo złożone z małych liter alfabetu łacińskiego opisujące kolory kolejnych koralików w oryginalnym naszyjniku. Kolejne znaki oznaczają kolory: n – niebieski, c – czerwony, z - zielony, a pozostałym literom przypisane są inne kolory. Liczba koralików jest nie większa niż 10 000.

Przykłady:

wynikiem **abc('nncnnbffbbbccczzzcz')** jest **2** (wystarczy usunąć trzeci i przedostatni koralik),

wynikiem **abc('zzzznnnnz')** jest **4** (wystarczy usunąć wszystkie niebieskie koraliki).

Omówienie rozwiązania

Zadanie można sprowadzić do policzenia długości najdłuższego ciągu koralików, który spełnia warunki zadania. Taki ciąg będziemy znajdować kolejno przeglądając koraliki, przy czym trzeba pamiętać najdłuższy możliwy ciąg koralików złożony tylko z niebieskich koralików `max_n`, z niebieskich i czerwonych `max_c` i z niebieskich, czerwonych oraz zielonych – `max_z`. W pętli w każdym z kroków:

- Gdy napotkamy koralik inny niż niebieski, czerwony lub zielony, zwiększamy wszystkie trzy liczniki.
- Gdy napotkamy koralik niebieski, zwiększamy licznik niebieskich, pozostałych nie zwiększamy, bo po czerwonych i zielonych, nie mogą występować niebieskie.
- Gdy napotkamy koralik czerwony, obliczamy nową wartość licznika czerwono-niebieskich (`max_c`) jako maksimum niebieskich i czerwonych zwiększony o 1. Może być sytuacja taka, że ciąg niebieskich koralików, będzie teraz ciągiem niebieskich i czerwonych.
- Gdy napotkamy zielony koralik, znajdujemy maksimum z ciągu koralików niebieskich, niebieskich i czerwonych oraz niebieskich – czerwonych – zielonych, zwiększony o 1.

Wynikiem zadania będzie różnica długości wejściowego ciągu i największej z wyliczonych liczb.

Warto zauważyć, że opisana metoda rozwiązania zadania jest dynamicznym podejściem. W programowaniu dynamicznym rozbijamy problem na podproblemy i na podstawie prostszego rozwiązania wnioskujemy o trudniejszym. Przedstawione rozwiązanie jest algorytmem o złożoności liniowej – czas działania algorytmu zależy proporcjonalnie od rozmiaru danych.

Rozwiązanie w języku Python

```python
def abc(napis):
    max_n = 0
    max_c = 0
    max_z = 0
    for e in napis:
        if e not in "ncz":
            max_n += 1
            max_c += 1
            max_z += 1
        else:
            if e == "n":
                max_n += 1
            if e == "c":
                max_c = 1 + max(max_n, max_c)
            if e == "z":
                max_z = 1 + max(max_n, max_c, max_z)

    return len(napis) - max(max_n, max_c, max_z)
```

Testy

Testy obejmują przypadki brzegowe, o danych wygenerowanych losowo dla zwiększającej się długości napisu – 20, 100, 1000 oraz regularne dla dużych danych. Warto zauważyć, że opisany wyżej algorytm daje wyniki w stosunkowo krótkim czasie dla wszystkich testów. Jeśli jednak rozwiązanie będzie w czasie kwadratowym lub wielomianowym, dla dużych testów czas istotnie się wydłuży.

Wywołanie - Python	**Wynik**
abc("n")	0
abc("zcc")	1
abc("zzz")	0
abc("nznzccczzzz")	2
abc("qwerty")	0
abc("ncnznqnnncccnyznnzcn")	7
abc("ncczqczunnczncccznyzbzzcgognnznanozmjzbzzncnnzwzqzcznyciztnzcvqzznce zccnztknzcnznncnnczzznnczpkngzzcln")	40
abc("yczznvlzkzzwzfcncsnznnznznxznzcpznckннknnzclzccjccccnzccccnvnonznnnzh cnzcdnrnccqcnhcindczccnnonozzczczzcnzjwnzznccznnzzzncuncqznusfczcnbzznnnz cbzznceceзindzccncnczclczzcozzjnczmcczcbcnrzzcnnfcnnezjznhzdennnczctpxvnzzz nnzanzcccnzznnnzcczcnxznczjenzznnnvzhcznznnzznnznnznngzccncnnnczccnzwnz cbcnncncncnfnccpcczcnvhcznanzcuwcccnzzzwncbcznbczedxccnzznzbzrzzczcnccnu cnnncacczzzznncapnnnzbcnrcncicennncnnzznczncnfznzncnevfbcznznzknnzccncznhcc zzdzlcnnzenccvcngcimynzpzpnzndycpczczcnvcczzczcxrjnvzznzzcnnnnczznlkncncnc zcocqznccncznxzhcnndftnyzcadccbchfacctcncnccnznnzzzqnczzznczzfcczzzcnsce bcbcrnznznmzzcznccznzczccdczyntcnzszocxnnnzcnfyczeeekccnzzanzcnncjzznac cczcccpzzczmzczddrcwnnnczcznvnnncnynzcgcinncnyrnzcccznnlnnzcccpzzccnzzjz nelcnqzcnnzwnzzdznntcnczzzccczccchzzzczrznfznzawcmzocnpcczznhcniztcznqzz czrynznzpjtcczvnnwzzczccncncyncgccmhnznzcncnzeznznzzfccqcnnncxnzzcnnncc cnnznczznzitnzcccdcnzecnnzzqncdcccknnkzzcczenclzxhcnczzznzzzcehnnnnnonntz nccnczorzcozzcznnnnzdlnzcnnzznocicctsncnnkczczonzzznczz")	486
abc("n"*3333+"c"*3333+"z"*3333+"c")	1
abc("ncz"*3333)	6664