Kompresja danych – zadania maturalne

- Wprowadzenie
- Przeczytaj
- Prezentacja multimedialna
- Sprawdź się
- Dla nauczyciela

Kompresja danych – zadania maturalne

Źródło: Life-Of-Pix, domena publiczna.

Kompresja danych to zagadnienie, z którym spotykamy się na co dzień, np. kiedy przesyłamy zdjęcia, filmy czy nagrania audio w internecie. Z e-materiałów dotyczących kompresji danych wiemy już, kiedy się ją stosuje, na czym ona polega, czym jest współczynnik kompresji, a także czym różni się kompresja stratna od bezstratnej. W tym e-materiale skupimy się na rozwiązywaniu zadań maturalnych dotyczących tych zagadnień.

Więcej informacji dotyczących kompresji danych znajdziesz w pozostałych materiałach z serii:

- Kompresja danych,
- Współczynnik kompresji danych,
- Kompresja stratna i bezstratna,
- Kompresja multimediów.

Twoje cele

- Skompresujesz bezstratnie kilka wybranych plików.
- Rozwiążesz zadanie typu maturalnego związane z kompresją i odkodowywaniem danych.
- Zdekompresujesz kilka wybranych plików.

Przeczytaj

Informacje wstępne

W trakcie lekcji poświęconych kompresji ustaliliśmy, że jej podstawą jest zmiana sposobu kodowania, czyli zmiana sposobu zapisu informacji.

Przykładowo, zapisanie liczb w systemie rzymskim w wielu przypadkach spowoduje zwiększenie liczby znaków potrzebnych do zapisu danej liczby. Liczba „2020", mogłaby zostać zapisana jako: MMXX.
Natomiast liczba „8" zostałaby zapisana jako VIII.

W pierwszym przypadku nie ma różnicy, a w drugim przypadku liczba znaków jest już różna.

W zdecydowanej większości przypadków korzystanie z systemu rzymskiego spowoduje zwiększenie liczby wymaganych znaków. O ile liczba „2020" dała się w sposób bezstratny konwertować, o tyle już liczba „2222" w systemie rzymskim prezentuje się następująco: MMCCXXII, co jest już wyraźnie dłuższym zestawem znaków.

Przykładowe zadanie typu maturalnego

Zadanie 1

W pewnym zakładzie obliczeniowym są przechowywane dane zapisane w postaci łańcuchów znaków. Metoda ta nie jest optymalna, ale ze względu na specyfikę działania tego konkretnego zakładu – konieczna. Zakład chce jednak zmniejszyć objętość plików, które muszą przechowywać, ale sposób kompresji powinien być w dalszym ciągu łatwo dekodowalny przez człowieka.

Przykładowe dane wejściowe:

```
5000, 2424222, 400, 562, 1245,
```

Podpunkt 1

Napisz program, w wybranym języku programowania, pseudokodzie lub w postaci listy kroków, który na podstawie dostarczonego pliku utworzy pseudoskompresowane pliki tekstowe, w których znajdą się informacje na temat powtarzających się pojedynczych liczb.

Specyfikacja:

Dane:

- `dane` – ciąg znaków, zawierający w sumie 119749 różnych znaków i przecinki; są w nim zapisane liczby

Wynik:

Pseudozakodowany plik, w którym poszczególne powtarzające się liczby zostały zapisane jako: „NxL", gdzie N to liczba powtórzeń danej cyfry, a L to cyfra. Między poszczególnymi liczbami powinny znajdować się również przecinki je oddzielające.

Poprawny wynik dla przykładowych danych wejściowych:

1x5 3x0, 1x2, 1x4 1x2, 1x4 3x2, 1x4 2x0, 1x5 1x6 1x2, 1x1 1x2 1x4 1x5,

Przedstawimy rozwiązanie w postaci pseudokodu. Załóżmy, że informacje z pliku zapisaliśmy w tablicy dwuwymiarowej. Pamiętaj jednak, że pisząc własny program, musisz zadbać o prawidłowe wczytanie danych z pliku.

Rozwiązanie:

W rozwiązaniu tym będziemy zliczać wystąpienia identycznych znaków występujących po sobie i po wystąpieniu innego znaku od razu będziemy dopisywać odpowiednie informacje do końca nowo utworzonego ciągu znaków.

```
wczytaj dane
znak := dane[0]
i := 0
licznik := 0
dla wszystkich znaków w dane:
    jeżeli dane[i] == znak:
        licznik := licznik+1
    w przeciwnym wypadku jeżeli czyJestCyfrą(znak):
        wyjsciowyCiagZnakow := wyjsciowyCiagZnakow + " " + liczni
        licznik := 1
        znak = dane[i]
    w przeciwnym wypadku:
        wyjsciowyCiagZnakow := wyjsciowyCiagZnakow + " " + liczni
        wyjsciowyCiagZnakow := wyjsciowyCiagZnakow + znak
        licznik := 1
    i := i + 1
zwroc wyjsciowyCiagZnakow
```

Podpunkt 2

Napisz program w wybranym języku programowania, pseudokodzie lub w postaci listy kroków, który przekoduje wynik z poprzedniego podpunktu tak, żeby odtworzyć dane wejściowe.

Specyfikacja:

Dane:

- `wejsciowyCiagZnakow` – ciąg znaków, w którym zakodowano **n** różnych liczb, w ten sposób, że np. liczba 100334440 zostałaby zakodowana jako: „1x1 2x0 2x3 3x4 1x0,”.

Wynik:

Ciąg znaków utworzony na podstawie zakodowanej wersji, w postaci:
100334440, ...

Rozwiązanie:

W poprzednim podpunkcie zakodowaliśmy dane, w tym podpunkcie wykonujemy operację odwrotną.

```
wczytaj wejsciowyCiagZnakow
liczba := wejsciowyCiagZnakow[0]
i := 0
czyTrafionoNaX := fałsz
wyjsciowyCiagZnakow := ""
dla wszystkich znaków w wejsciowyCiagZnakow:
    jeżeli czyJestCyfrą(wejsciowyCiagZnakow[i]) i czyTrafionoNaX
        liczba := liczba*10 + wejsciowyCiagZnakow[i]
    w przeciwnym wypadku, jeżeli wejsciowyCiagZnakow[i] == x:
        czyTrafionoNaX := prawda
    w przeciwnym wypadku, jeżeli czyJestCyfrą(wejsciowyCiagZnakow
        dla j = 0, 1, 2 ... liczba-1 wykonuj:
            wyjsciowyCiagZnakow := wyjsciowyCiagZnakow + wejsciow
    w przeciwnym wypadku:
        wyjsciowyCiagZnakow := wejsciowyCiagZnakow[i]
zwroc wyjsciowyCiagZnakow := ""
```

Słownik

> **kod**

ciąg jednoznacznie identyfikowalnych kombinacji znaków i symboli, dzięki którym możemy odczytać informację w sposób jednoznaczny

kompresja

proces polegający na zmianie sposobu zapisu informacji w ten sposób, aby zmniejszyć redundancję oraz objętość zbioru

plik

uporządkowany zbiór danych o skończonej długości stanowiący dla użytkownika systemu operacyjnego całość; może również posiadać atrybuty lub inne cechy, takie jak uprawnienia, czy nazwa, które dodatkowo go charakteryzują; możliwy zestaw cech i atrybutów zależy od konkretnego systemu

Prezentacja multimedialna

Zadanie 2.

> Zadanie utworzone na podstawie zadania 32 ze zbioru zadań do informatyki autorstwa CKE.

W zakładzie obliczeniowym doszło do rewolucji, gdy okazało się, że zmiana kodowania z poprzedniego zadania spowodowała jedynie zwiększenie rozmiaru plików, zamiast ich kompresji. Dlatego też postanowiono ponownie zmienić sposób zapisu informacji. Teraz dwukrotnie powtarzające się ciągi cyfr będą zastępowane pojedynczym wystąpieniem. Przykładowo, liczba: 11111111 mogłaby zostać skompresowana do formy (1111), lub (11)(11) lub dowolnej innej, z której po dekompresji otrzymalibyśmy wejściową liczbę.

Przykładowo, dla liczb:

```
55, 555, 443, 213, 12110397, 111122223333, 1111222233331111222 3

```

powinniśmy uzyskać ciągi:

```
(5), (5)5, (4)3, 213, 12(1)0397, (11)(22)(33), (111122223333)

```

Podpunkt 1

Napisz program, w wybranym języku programowania, pseudokodzie lub w postaci listy kroków, który, mając na wejściu skompresowany napis zapisany w tablicy, wypisze na wyjściu kolejne cyfry.

Specyfikacja:

Dane:

- `n` – dodatnia liczba całkowita
- `liczby[1..n]` – tablica skompresowanych liczb, metodą opisaną wcześniej

Wynik:

Zdekompresowana wartość liczbowa.

Polecenie 1

Zapoznaj się z prezentacją przedstawiającą rozwiązanie zapisane za pomocą pseudokodu. Następnie zaimplementuj algorytm w wybranym języku programowania.

1

Na egzaminie maturalnym możesz spotkać się z wieloma typami zadań otwartych, np. zadaniami z luką, zadaniami krótkiej odpowiedzi oraz zadaniami rozszerzonej odpowiedzi. Poniżej znajduje się przykładowe rozwiązanie zadania rozszerzonej odpowiedzi.
Źródło: Pixabay, CC0.

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PRmdvWUyt

W zadaniu tym występuje charakterystyczny wzorzec. Musimy odnaleźć wszystkie cyfry znajdujące się w nawiasie, a następnie wypisać je dwukrotnie na wyjście.

2

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PRmdvWUyt

Załóżmy, że naszym ciągiem do zdekompresowania jest 12(1)0(39). Poprawnym wynikiem byłoby w tym przypadku 121103939.

3

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PRmdvWUyt

Zaczynamy więc analizę naszej liczby – pierwszym znakiem jest „1”, więc dopisujemy 1 do ciągu wyjściowego.

Ciąg wyjściowy: 1

4

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PRmdvWUyt

Przechodzimy do kolejnego znaku, którym ponownie jest liczba. Tym razem jest to 2, więc też wypisujemy tę cyfrę tylko jeden raz na wyjście tak jak w poprzednim przypadku.

Ciąg wyjściowy: 12.

5

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PRmdvWUyt

Kolejnym znakiem, na jaki trafimy, będzie tym razem „(”, tego znaku nie dopisujemy do ciągu wyjściowego, jednakże sygnalizuje on nam, że kolejne cyfry, aż do pojawienia się znaku „)” będą musiały być podwojone. Nie możemy jednak po prostu w trakcie odczytu dopisać ich dwukrotnie na wyjście. Musimy utworzyć dodatkowy ciąg znaków, do którego kolejne cyfry spomiędzy nawiasów będziemy dopisywać, a dopiero ciąg uzyskany w taki sposób będziemy dopisywać do ciągu wyjściowego.

6

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PRmdvWUyt

Uzasadnienie tego jest takie, że jeżeli byśmy dopisywali od razu, zamiast przez kolejny ciąg, to przykładowo ciąg (121) zamiast do 121121 zdekodowalibyśmy do 112211.

7

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PRmdvWUyt

Następnym znakiem w naszym przykładzie jest 1, które jak przed chwilą omawialiśmy, dopisujemy do ciągu pomocniczego.

Ciąg pomocniczy: 1

Ciąg wyjściowy: 12

8

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PRmdvWUyt

Kolejnym znakiem jest „)”, oznacza to, że ciąg pomocniczy dopisujemy do ciągu wyjściowego dwa razy, a następnie zerujemy ciąg pomocniczy. W tym konkretnym przypadku akurat dopisanie dwukrotne jedynki do ciągu wyjściowego nic by nie zmieniło, ale z racji tego, że układamy algorytm, to powinniśmy tworzyć go w sposób jak najbardziej ogólny, tak żeby nie zagłębiać się w niepotrzebne szczegóły.

Ciąg wyjściowy: 1211

9

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PRmdvWUyt

Kolejną liczbą jest 0, które dopisujemy do ciągu wyjściowego.

Ciąg wyjściowy: 12110

10

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PRmdvWUyt

Następnie napotykamy ponownie otwarcie nawiasu, więc wszystkie cyfry do napotkania nawiasu zamykającego musimy umieszczać w ciągu pomocniczym.

11

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PRmdvWUyt

Napotkaną liczbę „3”, zgodnie z tym, co wspominaliśmy wcześniej, umieszczamy w ciągu pomocniczym.

Ciąg pomocniczy: 3

Ciąg wyjściowy: 12110

12

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PRmdvWUyt

Napotykamy ponownie cyfrę, więc tak jak w poprzednim kroku dodajemy ją do ciągu pomocniczego.

Ciąg pomocniczy: 39

Ciąg wyjściowy: 12110

13

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PRmdvWUyt

Spotykamy zamknięcie nawiasu, które jest również ostatnim znakiem w naszym dekompresowanym tekście.

Dodajemy więc dwukrotnie do naszego ciągu wyjściowego ciąg pomocniczy, uzyskując:

Ciąg wyjściowy: 121103939

14

Więcej zadań znajdziesz w zbiorze dostępnym na stronie internetowej Okręgowej Komisji Egzaminacyjnej w Warszawie.
Źródło: OKE, oke.waw.pl, tylko do użytku edukacyjnego.

```
1 czyWNawiasie := falsz
2 ciagPomocniczy := ""
3 ciagWyjsciowy := ""
4 dla i = 1, 2, 3,.. n
  wykonuj:
5     jeżeli liczby[i] == "
  (" wykonaj:
6         czyWNawiasie :=
  prawda
```

```
    w przeciwnym razie, jeżeli jestCyfrą(liczby[i]) wykonaj:
        jeżeli czyWNawiasie wykonaj:
            ciagPomocniczy := ciagPomocniczy + liczby[i]
        w przeciwnym wypadku wykonaj:
            ciagWyjsciowy := ciagWyjsciowy + liczby[i]
    w przeciwnym razie, jeżeli liczby[i] == ")" wykonaj:
        czyWNawiasie := falsz
        ciagWyjsciowy := ciagWyjsciowy + ciagPomocniczy + ciagPomocniczy
        ciagPomocniczy := ""

Gdzie: funkcja jestCyfrą(), sprawdza, czy podany znak jest cyfrą.
Operacja „+” oznacza w powyższych przypadkach konkatenacje.
```

Sprawdź się

Pokaż ćwiczenia:

Ćwiczenie 1

Dokończ zdanie.

Dla ciągu: „5, 31, 245, 33356, 355557, 23456”, korzystając z omówionego algorytmu kompresji, moglibyśmy uzyskać ciąg...

- ◯ 5, 31, 245, (33)56, 3(55)(55)7, 23456.
- ◯ (5), (3)(1), (2)(4)(5), (3)(3)(5)(6), (3)(5)(5)(5)(5)(7), (2)(3)(4)(5)(6).
- ◯ 5, 31, 245, (333)56, 3(5555)7, 23456.
- ◯ 5, 31, 245, (3)356, 3(55)7, 23456.

Ćwiczenie 2

Zdekompresuj liczby, które zostały skompresowane zgodnie z metodą używaną w poprzednim poleceniu.

Liczby skompresowane	Liczby zdekompresowane
12(1134)	1211341134
84(1)(44)	
123(24)354	
1092(1344)43	
(44)84(1)	
343214	

Ćwiczenie 3

Wskaż, czy zastosowana przez nas metoda kompresji może spowodować zwiększenie liczby użytych znaków zamiast jej zmniejszenia.

- ◯ Tak, ponieważ w skrajnym przypadku liczba znaków może zwiększyć się nawet trzykrotnie.
- ◯ Nie, ponieważ kompresja zawsze powoduje zmniejszenie kompresowanej wartości.

Ćwiczenie 4

Skompresuj liczby z tabeli w sposób optymalny (to znaczy tak, żeby w jak największym stopniu uniknąć redundancji). Skorzystaj z metody kompresji, jak w dwóch poprzednich poleceniach.

Liczby zdekompresowane	Liczby skompresowana
1211341134	12(1134)
12110397	
111112222	
9996654	
414841	
112233445566	

Ćwiczenie 5

Wybierz właściwe dokończenia zdania.
Dla ciągu cyfr: 00011123456666...

- ☐ fragment „000...” po kompresji ma taką samą długość jak przed kompresją.
- ☐ fragment „...6666” po kompresji ma taką samą długość jak przed kompresją.
- ☐ fragment „...111...” po kompresji zwiększył swoją długość.
- ☐ liczba użytych znaków po kompresji nie zwiększy się.
- ☐ liczba użytych znaków po kompresji zwiększy się.

Ćwiczenie 6

Określ, dla ilu powtórzeń liczby z rzędu omówiona wcześniej kompresja rzeczywiście pozwala zmniejszyć rozmiar liczby.

- ◯ dla 4 i więcej powtórzeń
- ◯ dla 6 i więcej powtórzeń
- ◯ dla 5 i więcej powtórzeń
- ◯ dla 3 i więcej powtórzeń

Ćwiczenie 7

Uzupełnij tekst

Dla omówionej metody kompresji charakterystyczne jest, że próbujemy zmniejszyć redundancję przez zapisywanie powtórzonych zlepek [ ]. Metoda ta jednak nie zawsze jest skuteczna, ponieważ w wyniku kompresji pojawiają się dodatkowo [ ] nawiasy, które również wliczamy do długości danej liczby.
Z tego powodu, w przypadku gdy kompresujemy występujące po sobie [ ] cyfry, to liczba się zwiększa, zamiast się zmniejszać.

dwie | tych samych cyfr | trzy | ciągów | cztery | dwa

Ćwiczenie 8

Wskaż, czy dla liczby „99999” kompresja spowoduje zmniejszenie się jej rozmiaru.

- ◯ tak
- ◯ nie

Dla nauczyciela

Autor: Maurycy Gast
Przedmiot: Informatyka

Temat: Kompresja danych – zadania maturalne

Grupa docelowa:

Szkoła ponadpodstawowa, liceum ogólnokształcące, technikum

Podstawa programowa:

Treści nauczania – wymagania szczegółowe

I + II. Zakres rozszerzony. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:

2) wykorzystuje znane sobie algorytmy przy rozwiązywaniu i programowaniu rozwiązań następujących problemów:

a) rozkładania liczby na czynniki pierwsze,

b) wykonywania działań na liczbach w systemach innych niż dziesiętny,

c) znajdowania w ciągu podciągów o różnorodnych własnościach, np. najdłuższego spójnego podciągu niemalejącego, spójnego podciągu o największej sumie,

d) zamiany wyrażenia na postać w odwrotnej notacji polskiej i obliczanie jego wartości na podstawie tej postaci,

e) badania przecinania się odcinków, przynależności punktu do trójkąta,

f) obliczanie przybliżonej wielkości pola obszarów zamkniętych;

Kształtowane kompetencje kluczowe:

- kompetencje cyfrowe;
- kompetencje osobiste, społeczne i w zakresie umiejętności uczenia się;
- kompetencje matematyczne oraz kompetencje w zakresie nauk przyrodniczych, technologii i inżynierii.

Cele operacyjne (językiem ucznia):

- Skompresujesz bezstratnie kilka wybranych plików.

- Rozwiążesz zadanie typu maturalnego związane z kompresją i odkodowywaniem danych.
- Zdekompresujesz kilka wybranych plików.

Strategie nauczania:

- konstruktywizm;
- konektywizm.

Metody i techniki nauczania:

- dyskusja;
- rozmowa nauczająca z wykorzystaniem multimedium i ćwiczeń interaktywnych;
- ćwiczenia praktyczne.

Formy pracy:

- praca indywidualna;
- praca w parach;
- praca w grupach;
- praca całego zespołu klasowego.

Środki dydaktyczne:

- komputery z głośnikami, słuchawkami i dostępem do internetu;
- zasoby multimedialne zawarte w e-materiale;
- tablica interaktywna/tablica, pisak/kreda;
- oprogramowanie dla języka C++, w tym kompilator GCC/G++ 4.5 (lub nowszej wersji) i Code::Blocks 16.01 (lub nowszej wersji), Orwell Dev-C++ 5.11 (lub nowszej wersji) lub Microsoft Visual Studio;
- oprogramowanie dla języka Java SE 8 (lub nowszej wersji), w tym Eclipse 4.4 (lub nowszej wersji);
- oprogramowanie dla języka Python 3 (lub nowszej wersji), w tym PyCharm lub IDLE.

Przebieg lekcji

Przed lekcją:

1. **Przygotowanie do zajęć.** Nauczyciel loguje się na platformie i udostępnia e-materiał: „Kompresja danych – zadania maturalne". Uczniowie zapoznają się z treściami w sekcji „Przeczytaj" w kontekście programowania.

Faza wstępna:

1. Nauczyciel wprowadza uczniów szczegółowo w temat lekcji i jej cele. Może posłużyć się wyświetloną na tablicy zawartością sekcji „Wprowadzenie".

2. **Rozpoznanie wiedzy uczniów.** Nauczyciel zadaje uczniom pytanie dotyczące ich aktualnego stanu wiedzy w obszarze poruszanego tematu, opartego o programowanie.

Faza realizacyjna:

1. **Praca z tekstem.** Nauczyciel wyświetla zawartość sekcji „Przeczytaj". Na forum klasy uczniowie analizują przedstawione w niej rozwiązania zadanie: „W pewnym zakładzie obliczeniowym są przechowywane dane zapisane w postaci łańcuchów znaków. Metoda ta nie jest optymalna, ale ze względu na specyfikę działania tego konkretnego zakładu – konieczna. Zakład chce jednak zmniejszyć objętość plików, które muszą przechowywać, ale sposób kompresji musi być w dalszym ciągu łatwo dekodowalny przez człowieka."
2. **Praca z multimedium.** Nauczyciel wyświetla zawartość sekcji „Prezentacja multimedialna". Uczniowie wspólnie zapoznają się rozwiązaniem przestawiającym jak napisać program, w wybranym języku programowania, pseudokodzie lub w postaci listy kroków, który mając na wejściu skompresowany napis zapisany w tablicy, wypisze na wyjściu kolejne cyfry.
3. **Ćwiczenie umiejętności.** Liga zadaniowa - uczniowie wykonują indywidualnie na czas ćwiczenia nr 1-8 z sekcji „Sprawdź się", a następnie omawiają zadania na forum.

Faza podsumowująca:

1. Na koniec zajęć z programowania nauczyciel prosi uczniów o rozwinięcie zdania: „Na dzisiejszych zajęciach nauczyłam/łem się jak...".

Praca domowa:

1. Napisz (w postaci listy kroków, schematu blokowego, pseudokodu lub w wybranym języku programowania) algorytm obliczający rozmiar skompresowanego tekstu.

Materiały pomocnicze:

- Oficjalna dokumentacja techniczna dla języka C++.
- Oficjalna dokumentacja techniczna dla kompilatora GCC/G++ 4.5 (lub nowszej wersji).
- Oficjalna dokumentacja techniczna dla oprogramowania Code::Blocks 16.01 (lub nowszej wersji), Orwell Dev-C++ 5.11 (lub nowszej wersji) lub Microsoft Visual Studio.
- Oficjalna dokumentacja techniczna dla języka Java SE 8 (lub nowszej wersji).
- Oficjalna dokumentacja techniczna dla oprogramowania Eclipse 4.4 (lub nowszej wersji).
- Oficjalna dokumentacja techniczna dla języka Python 3 (lub nowszej wersji).
- Oficjalna dokumentacja techniczna dla oprogramowania PyCharm lub IDLE.

Wskazówki metodyczne:

- Uczniowie mogą wykorzystać multimedium w sekcji „Prezentacja multimedialna” do przygotowania się do lekcji powtórkowej.