Opracowanie: e/
Problem E: Kompresor bitowy

HISTORIA:
- wersja 1.1: 2007, Tomasz Idziaszek
- wersja 1.0: 2007, Karolina Sołtys, Marcin Balcerzak

dokument systemu SINOL 1.3.1

1 Definicje

Wprowadźmy wpierw kilka używanych później pojęć. Przez podciąg rozumiemy wszędzie podciąg spójny.

Definicja 1 Kodowaniem *nazywamy kompresję zdefiniowaną w treści zadania.*

Definicja 2 Odkodowaniem *nazywamy dekompresję — proces odwrotny do kompresji.*

Definicja 3 Ciągiem wyjściowym *nazywamy ciąg zerojedynkowy, jaki program otrzymuje jako dane wejściowe (do odkodowania).*

Definicja 4 Ciągiem wejściowym (odkodowanym) *nazywamy ciąg mający odpowiednią długość i zawierający odpowiednią liczbę jedynek, ktory po zakodowaniu daje ciąg zerojedynkowy (wyjściowy) jaki program otrzymuje jako dane wejściowe. Zadanie polega na sprawdzeniu czy dla danych danych wejściowych liczba możliwych ciągów wejściowych wynosi* 0, 1, *czy jest większa niż* 1. *Wtedy odpowiednio zwracamy na wyjściu: NO, YES, NOT UNIQUE.*

Definicja 5 Blokiem (bloczkiem) *danego ciągu nazywamy pewien jego spójny podciąg postaci* $1 \ldots 10 \ldots 0$ *taki, że nie istnieje spójny podciag tej postaci dłuższy od niego i go zawierający.*

2 Obserwacje

Dokonajmy teraz kilku spostrzeżeń, które będziemy w dalszej cześci uważali za oczywiste.

Fakt 1 *Odkodowane spójne podciągi jedynek nie stykają się ze sobą (kodowane były takie podciągi jedynek, że nie istniały większe je zawierające).*

Fakt 2 *Podciąg dwóch jedynek (nie stykający się z jedynkami dla rozkompresowanej pozostałej części liczby) mógł powstać zarówno z* 11 *jak i* 111.

Fakt 3 *Podciągi w ciągu wyjściowym kodujące podciągi jedynek z ciągu wejściowego zaczynają się cyfrą* 1, *na lewo od nich nie występuje oczywiście cyfra* 1, *zaś kończą cyfrą* 1 *poprzedzającą* 0 *(lub ostatnią w ciągu) bądź też cyfrą* 0 *o ile nie poprzedza ona cyfry* 1.

3 Podstawowa technika

Mając dane: długość ciągu wejściowego, liczbę jedynek w ciągu wejściowym i ciąg wyjściowy znamy liczbę zer, jaka pojawiła się w wyniku kodowania. Rozwiązanie polega na przejrzeniu możliwych poprawnych (tj. prowadzących do ciągów wejściowych zgodnych z warunkami zadania) rozdysponowań tej puli zer pomiędzy zera w ciągu wyjściowym i policzeniu zgodnych z warunkami zadania odkodowań.

4 Implementacja

Trzymamy tablicę (maksymalnie 21 elementów) opisującą bloczki i konkatenacje bloczków. Opis ten zawiera liczbę jedynek uzyskaną przy rozkompresowaniu danego bloczka (przy czym wartości początkowe nie uwzględniają oczywiście wykorzystania zer w bloczku) oraz ilość zer, jaką mamy do dyspozycji w bloczku (inicjowane ilością wszystkich zer w bloczku).

Rozdzielamy pulę zer powstałych z kompresji rekurencyjnie, mając pulę początkową i zaczynając od pierwszego (licząc od lewej) bloczka. Zauważamy, że gdy liczba zer w bloczku jest nie większa niż liczba zer w dostępnej aktualnie puli do rozdysponowania, trzeba skleić ze sobą bloczek z bloczkiem następnym (następny ma zatem zmodyfikowany opis stanu) i rozważać problem dla następnego bloczka — aby nie zgubić możliwych rozwiązań. Później oczywiście zajmujemy się także klasycznym rozdysponowaniem puli w obrębie bloczka i z pomniejszoną pulą przechodzimy do kolejnego bloczka.

Liczba znalezionych rozwiązań oraz liczba jedynek powstała po rozkodowaniu wszystkiego na lewo od aktualnie przetwarzanego bloczka jest pamiętana globalnie.

Rozwiązanie jest poprawne, gdy po ostatnim bloczku rozdysponowaliśmy całą pulę zer i długośc otrzymanego ciągu wejściowego oraz liczba jedynek się zgadza z danymi wejściowymi.

5 Poprawność algorytmu

5.1 Warunek stopu

Algorytm zatrzymuje się po przetworzeniu wszystkich bloczków, przy czym liczba wszystkich możliwych rozdysponowań zer pomiędzy bloczki jest skończona, zatem czas działania algorytmu jest skończony.

5.2 Częściowa poprawność

Znajdowane odkodowania są w oczywisty sposób poprawne. Przestrzeń stanów przeszukujemy w sposób pełny, czyli nie pomijamy żadnego poprawnego odkodowania.

6 Złożoność algorytmu

Złożoność pamięciowa jest stała, gdyż nie alokujemy dynamicznie żadnych struktur danych. Maksymalna pesymistyczna głębokość wywołań rekurencyjnych funkcji badającej bloczki to 21. Pesymistyczna złożoność czasowa jest wykładnicza w stosunku do iloczynu długości wszystkich maksymalnych podciągow złożonych z zer powiększonych o 1 (iloczyn ten jest maksymalny gdy podciągi te są równej długości — wynika to z nierówności między średnią arytemtyczną a geometryczną — więc dla n podciągów szacuje się to przez $(\frac{40-n}{n}+1)^n$, co przyjmuje maksimum — prawie 2^{22} — dla $n = 15$).

7 Testy

Integralną część rozwiązania stanowią zestawy testów. Poniżej znajduje się ich krótki opis:

1.in sprawdza warunki brzegowe dla niewielkich danych

2.in testy poprawnościowe dla danych długości 5

3.in testy poprawnościowe dla danych długości 10

4.in testy poprawnościowo–wydajnościowe dla danych długości 15

5.in testy poprawnościowo–wydajnościowe dla danych długości 20

6.in testy poprawnościowo–wydajnościowe dla danych długości 30

7.in testy poprawnościowe, dane losowe o długości 40

8.in testy poprawnościowe, dane losowe o długości 40

9.in testy poprawnościowe, dane losowe o długości 30

10.in testy poprawnościowe, dane losowe o długości 20