7.6. Wprowadzenie do odśmiecania bazującego na śledzeniu

Odśmiecacze śledzące nie zakładają natychmiastowego odśmiecania, gdy tylko dane śmieciowe zostaną stworzone – zamiast tego uruchamiają się okresowo w celu wyszukania nieosiągalnych obiektów i zwolnienia zajmowanej przez nie pamięci. Zazwyczaj odśmiecacz taki jest uruchamiany, gdy zaczyna brakować wolnej pamięci lub gdy jej ilość spadnie poniżej pewnego progu.

Zaczniemy od przedstawienia najprostszego algorytmu „oznakuj i zamieć" (*mark and sweep*). Następnie opiszemy różne śledzące algorytmy, posługując się koncepcją czterech stanów, w których mogą znajdować się fragmenty pamięci. Podrozdział ten przedstawia też kilka ulepszeń podstawowego algorytmu, również takich, w których relokacja obiektu stanowi element funkcjonalności odśmiecania pamięci.

7.6.1. Podstawowy odśmiecacz Mark and Sweep

Algorytmy odśmiecania pamięci typu *Mark and Sweep* (oznacz i zamieć) są prostymi, oczywistymi procedurami, które wyszukują wszystkie nieosiągalne obiekty i umieszczają je na liście wolnej pamięci. Algorytm 7.12 odwiedza i „znakuje" wszystkie osiągalne obiekty w pierwszym kroku śledzenia, po czym „zamiata" całą stertę, zwalniając nieosiągalne obiekty.

Algorytm 7.14, który omówimy po przedstawieniu ogólnych założeń dla algorytmów opartych na śledzeniu, stanowi efekt optymalizacji algorytmu 7.12. Dzięki użyciu dodatkowej listy do przechowania wszystkich alokowanych obiektów odwiedza on każdy osiągalny obiekt tylko raz.

Algorytm 7.12: Odśmiecacz typu Mark and Sweep.

WEJŚCIE: Zestaw podstawowy obiektów, sterta oraz lista wolnej pamięci, nazwana *Wolne*, zawierająca wszystkie niealokowane fragmenty sterty. Podobnie jak w podrozdziale 7.4.4, wszystkie fragmenty pamięci są oznakowane znacznikami granic wskazującymi ich status wolny/zajęty oraz rozmiar.

WYJŚCIE: Zmodyfikowana lista *Wolne* po usunięciu wszystkich danych śmieciowych.

METODA: Algorytm pokazany na rysunku 7.21 używa kilku prostych struktur danych. Lista *Wolne* zawiera obiekty, o których wiadomo, że są wolne. Lista *Nieprzeskanowane* przechowuje obiekty, co do których ustaliliśmy, że są osiągalne, ale których następcy nie zostali jeszcze sprawdzeni. Inaczej mówiąc, nie przeskanowaliśmy jeszcze tych obiektów, aby sprawdzić, czy można za ich pośrednictwem osiągnąć inne obiekty. Lista *Nieprzeskanowane* jest początkowo pusta. Dodatkowo każdy obiekt zawiera bit wskazujący, czy udało się do niego

```
    /* faza znakowania */
1)  dodaj każdy obiekt wskazywany przez zestaw podstawowy
    do listy Nieprzeskanowane i ustaw jego bit osiągalności na 1
2)  while (Nieprzeskanowane ≠ ∅) {
3)      usuń pewien obiekt o z listy Nieprzeskanowane;
4)      for ((każdy obiekt o' wskazywany w o) {
5)          if (o' jest nieosiągalny, tzn. jego bit osiągalności to 0) {
6)              ustaw bit osiągalności o' na 1;
7)              umieść o' na Nieprzeskanowane;
            }
        }
    }
    /* faza zamiatania */
8)  Wolne = ∅;
9)  for ((dla każdego fragmentu pamięci o na stercie) {
10)     if (o nie został osiągnięty, czyli jego bit osiągalności to 0)
            dodaj o do Wolne;
11)     else ustaw bit osiągalności o na 0;
    }
```

RYSUNEK 7.21: Odśmiecacz „oznakuj i zamieć”

dotrzeć (bit osiągalności). Przed uruchomieniem algorytmu wszystkie alokowane obiekty mają bit osiągalności ustawiony na 0.

W wierszu (1) pseudokodu z rysunku 7.21 inicjujemy listę *Nieprzeskanowane*, umieszczając na niej wszystkie obiekty należące do zestawu podstawowego. Bit osiągalności dla tych obiektów jest również ustawiany na 1. Wiersze od (2) do (7) to pętla, w której kolejno badamy każdy obiekt o, który znalazł się na liście *Nieprzeskanowane*.

Pętla *for* w wierszach od (4) do (7) implementuje skanowanie obiektu o. Sprawdzamy każdy obiekt o', dla którego znajdziemy referencję w obiekcie o. Jeśli o' został już wcześniej osiągnięty (jego bit osiągalności to 1), wówczas nie musimy nic robić z o'; albo został już przeskanowany wcześniej, albo znajduje się na liście *Nieprzeskanowane* i zostanie sprawdzony później. Jednak jeśli wcześniej nie dotarliśmy do o', wówczas musimy ustawić jego bit osiągalności na 1 w wierszu (6) i dodać go do listy *Nieprzeskanowane* w wierszu (7).

Rysunek 7.22 ilustruje ten proces. Pokazuje listę *Nieprzeskanowane* z czterema obiektami. Pierwszy obiekt na tej liście, odpowiadający obiektowi o w powyższym omówieniu, jest w trakcie skanowania. Przerywane linie odpowiadają trzem rodzajom obiektów, które mogą być osiągalne z o:

1. Wcześniej przeskanowany obiekt, którego nie trzeba skanować ponownie.
2. Obiekt znajdujący się obecnie na liście *Nieprzeskanowane*.
3. Element, który jest osiągalny, ale wcześniej był uważany za nieosiągalny.