Koszt zamortyzowany

Adam MALINOWSKI[*]

[*]Instytut Informatyki, Uniwersytet Warszawski

W ekonomii określenie „amortyzacja” oznacza, z grubsza rzecz biorąc, rozłożenie jakiejś wartości na pewien okres. Z amortyzacją mamy do czynienia, kiedy spłacamy raty kredytu, kiedy spółdzielnia mieszkaniowa zbiera składki na fundusz remontowy przed przystąpieniem do kosztownej naprawy dachu, albo kiedy operator telefonii komórkowej przenosi niewykorzystane przez nas „darmowe minuty” z jednego miesiąca na następny.

W informatyce pojęcia tego używa się zazwyczaj w kontekście analizy czasu wykonywania ciągu operacji o różnych kosztach. *Koszt zamortyzowany* dla sekwencji n operacji to *średni koszt przypadający na pojedynczą operację*, czyli łączny koszt wszystkich operacji podzielony przez n. Zauważmy, że chociaż mówimy tu o wartości średniej, powyższa definicja nie ma nic wspólnego z rozważaniami probabilistycznymi: uśrednianie dotyczy danego ciągu operacji, a nie zbioru możliwych kosztów pojedynczej operacji.

Licznik binarny

Rozważmy następujący, klasyczny przykład. Wiadomo, że każda liczba naturalna n ma jednoznaczne przedstawienie w postaci sumy potęg dwójki. Załóżmy, że mamy (nieskończoną) tablicę bitów B reprezentującą taką liczbę: $B[i] = 1$ wtedy i tylko wtedy, gdy we wspomnianej sumie występuje składnik 2^i. Oto pseudokod operacji dodania jedynki do liczby reprezentowanej przez B:

```
Inc(B)
    i ← 0
    dopóki B[i] = 1 wykonuj
        B[i] ← 0
        i ← i + 1
    B[i] ← 1
```

Czas działania tej procedury zależy oczywiście od zawartości tablicy B. Jeśli pierwszych k bitów w B to jedynki, operacja `Inc` działa w czasie rzędu $k + 1$. Jeśli zatem liczba reprezentowana przez B jest z zakresu od 0 do n, to górnym oszacowaniem czasu działania pojedynczej operacji `Inc` jest $O(\lg n)$, ponieważ długość reprezentacji binarnej liczby n wynosi dokładnie $\lfloor \log_2 n \rfloor + 1$.

Załóżmy teraz, że wywołujemy procedurę `Inc` po kolei n razy, zwiększając licznik od 0 do n. Stosując jedynie górne ograniczenie kosztu pojedynczej operacji, dostajemy oszacowanie łącznego kosztu $O(n \lg n)$, które – chociaż poprawne – można wzmocnić.

Szacowanie kosztu zamortyzowanego

Najprostszy sposób dokładnego oszacowania sumarycznego kosztu powyższego ciągu operacji opiera się na następującej obserwacji: bit $B[0]$ jest zmieniany przy *każdej* operacji, bit $B[1]$ przy *co drugiej*, bit $B[2]$ przy *co czwartej* itd. Zatem łączny koszt związany ze zmianą bitu $B[0]$ podczas rozważanych n operacji wynosi n, koszt zmian bitu $B[1]$ to $\lfloor n/2 \rfloor$, koszt zmian bitu $B[2]$ to $\lfloor n/4 \rfloor$ itd. Sumując te koszty, widzimy, że koszt zmian na *wszystkich* pozycjach nie przekracza $2n$ (koszt zamortyzowany operacji jest zatem *stały*!).

Znacznie ogólniejszy sposób szacowania kosztu zamortyzowanego, noszący nazwę *Metody Potencjału*, jest oparty na analogii do energii potencjalnej w układzie fizycznym. *Potencjał* to pewna funkcja $\Phi : D \longrightarrow \mathbb{R}$, gdzie D to zbiór wszystkich możliwych stanów struktury danych zmienianej przez wykonywane operacje (u nas: zbiór możliwych zawartości tablicy B). Oznaczmy przez Φ_i potencjał po wykonaniu i-tej operacji, a przez c_i – jej faktyczny koszt. *Koszt zamortyzowany i-tej operacji* jest zdefiniowany jako *koszt faktyczny plus zmiana potencjału*: $c_i + \Phi_i - \Phi_{i-1}$. Sumując koszty zamortyzowane wszystkich operacji, dostajemy

$$\sum_{i=1}^{n} (c_i + \Phi_i - \Phi_{i-1}) = \sum_{i=1}^{n} c_i + \Phi_n - \Phi_0.$$

Jeśli zatem, określając funkcję potencjału, zadbamy o to, żeby potencjał stanu początkowego Φ_0 był zerowy oraz $\Phi_i \geqslant 0$ dla każdego i, to mamy gwarancję, że suma kosztów zamortyzowanych stanowi górne oszacowanie sumy kosztów faktycznych.

Oczywiście, cała trudność polega na dobraniu odpowiedniej funkcji potencjału, dającej najlepsze oszacowanie. Nie ma tu sztywnych reguł, ale ogólna intuicja jest następująca: tania operacja powinna powodować niewielki wzrost potencjału, natomiast droga operacja – duży spadek potencjału, rekompensujący poniesiony koszt faktyczny. W naszym przykładzie z licznikiem binarnym możemy określić potencjał jako *łączną liczbę jedynek* w tablicy B. Wtedy jeśli operacja `Inc` spowodowała wyzerowanie k początkowych bitów i ustawienie $(k + 1)$-go bitu, to jej koszt faktyczny wyniósł $k + 1$, a zmiana potencjału $1 - k$ (ubyło k jedynek, a potem jedna przybyła). Koszt zamortyzowany jest zatem stały, taki sam jak w poprzednim oszacowaniu.

Zastanówmy się jeszcze, co by było, gdyby koszt zmiany i-tego bitu w tablicy nie był jednostkowy, ale wynosił 2^i (każdy kolejny bit jest dwa razy cięższy od poprzedniego, więc jego odwrócenie jest dwa razy droższe). Wtedy koszt pojedynczej operacji w sekwencji n kolejnych wywołań procedury `Inc` może wynieść nawet n. Podobnie jak poprzednio, można jednak

wykazać, że koszt zamortyzowany przypadający na jedną operację to tylko $O(\lg n)$: łączny koszt zmian bitu $B[0]$ wynosi n, bit $B[1]$ jest zmieniany (z kosztem 2) w co drugiej operacji, co łącznie kosztuje co najwyżej $2 \cdot n/2 = n$ itd. Suma tych kosztów to $O(n \lg n)$, więc koszt zamortyzowany jest zgodny z zapowiedzią.

Zamortyzowany słownik

Słownik to struktura danych, która umożliwia wyszukiwanie i wstawianie elementów. Wyszukiwanie binarne w posortowanej tablicy, zawierającej n elementów, jest szybkie – wymaga czasu $O(\lg n)$ – ale wstawianie jest kosztowne (pesymistycznie rzędu n). Oto bardzo prosta realizacja słownika, w której wyszukiwanie trwa co prawda nieco dłużej ($O(\lg^2 n)$), ale za to (zamortyzowany) koszt wstawiania jest znacznie niższy: $O(\lg n)$.

Nasz słownik to zbiór tablic `A0`, `A1`, `A2`, ..., w którym i-ta tablica ma rozmiar 2^i. Każda tablica jest albo pusta, albo całkowicie wypełniona. Wszystkie tablice są posortowane, ale nie zakładamy żadnych zależności między elementami z różnych tablic. Oto przykładowa zawartość słownika:

```
A0:  [3]
A1:  [2, 5]
A2:  pusta
A3:  [1, 4, 7, 8, 9, 10, 11, 12]
```

W celu wyszukania elementu wykonujemy wyszukiwanie binarne w każdej wypełnionej tablicy. W najgorszym razie wymaga to czasu $O(\lg^2 n)$ (bo liczba wypełnionych tablic to $O(\lg n)$).

Wstawianie wykonujemy następująco. Tworzymy nową tablicę rozmiaru 1, zawierającą wstawiany element, powiedzmy 6. Jeśli tablica `A0` była pusta, to umieszczamy nową tablicę w miejsce `A0`. Jeśli nie (tak jak w powyższym przykładzie), to *scalamy* nową tablicę z `A0`, uzyskując nową tablicę rozmiaru 1 (u nas: `[3, 6]`). Tablica `A0` zostaje przy tym opróżniona. Jeśli tablica `A1` była pusta, to umieszczamy nową tablicę w miejsce `A1`, a jeśli nie (tak jak u nas), to scalamy nową tablicę z `A1`, sprawdzamy, czy tablica `A2` jest pusta, itd. W naszym przykładzie w wyniku wstawienia elementu 6 otrzymamy słownik o następującej zawartości:

```
A0:  pusta
A1:  pusta
A2:  [2, 3, 5, 6]
A3:  [1, 4, 7, 8, 9, 10, 11, 12]
```

Ile kosztuje taka operacja? Przyjmijmy, że koszt utworzenia początkowej tablicy rozmiaru 1 wynosi 1, a scalenie dwóch posortowanych tablic rozmiaru r kosztuje $2r$. Nietrudno zauważyć, że koszt wstawienia i-tego elementu do słownika jest taki sam jak koszt i-tego zwiększenia „obciążonego” licznika binarnego, zatem zamortyzowany koszt wstawienia jest logarytmiczny względem rozmiaru słownika.

Wyniki XXIV Ogólnopolskiego Sejmiku Matematyków – Wisła, 31 V – 3 VI 2007

Konkurs polega na przedstawieniu opracowania jednego z tematów zaproponowanych przez Jury (wraz z bibliografią) lub tematu własnego oraz – w przypadku zakwalifikowania się do finału – krótkim, publicznym zreferowaniu tego opracowania.

W roku 2007/8 zaproponowane przez Jury tematy to: metody probabilistyczne w matematyce dyskretnej, wielomianowe algorytmy rozstrzygania pierwszości, dowody z wiedzą zerową, cechy podzielności liczb, mozaiki – między matematyką a sztuką, pewnik Archimedesa, po co kongruencje? czym najlepiej przybliżać? ile kul zmieści się w sześcianie? twierdzenia o punkcie stałym, nietypowe zbiory.

Sejmiki organizuje Pracownia Matematyki Pałacu Młodzieży w Katowicach we współpracy z Uniwersytetem Śląskim; www.pm.katowice.pl/pracownia/matematyka

Jury w składzie: prof. dr hab. Maciej Sablik – przewodniczący, dr Marian Podhorodyński – zastępca przewodniczącego, dr Lech Bartłomiejczyk, dr Tomasz Bielaczyc, mgr Włodzimierz Fechner, mgr Żywilla Fechner, dr Erwin Kasparek, prof. dr hab. Mieczysław Kula, dr Tomasz Kochanek, mgr Barbara Przebieracz, dr Anna Szczerba-Zubek, dr Marta Tyran-Kamińska przyznało

I miejsce **Mateuszowi Plucie** z V LO w Krakowie za pracę *Wykorzystanie inwersji względem okręgu w dowodzie twierdzenia o $n+2$ okręgach stycznych*;

II miejsce ex aequo
Tomaszowi Kobosowi z V LO w Krakowie za pracę *Zbiory kresowe dla wielomianów o współczynnikach całkowitych*;
Łukaszowi Musze z III LO w Chorzowie za pracę *Uśrednić*;

oraz wyróżnienia
Adrianowi Łańcuckiemu z I LO w Legnicy za pracę *Problem szczęśliwego zakończenia*;
Piotrowi Misce z Gimn. nr 5 w Sosnowcu za pracę *Liczby pierwsze w ciągu Fibonacciego i ciągu Lukasa*;
Jackowi Rzeniewiczowi z I LO w Gdańsku za pracę *Szyfrowanie i szyfrowywacz*.

W głosowaniu nauczyciele nagrodzili **Adriana Łańcuckiego**, a uczniowie **Marcina Oczeretko** z I LO w Legnicy za pracę *Sieci przepływowe*.