Algorytmy i Struktury Danych, 8. ćwiczenia

2019-11-20

Spis treści

1 Licznik binarny

1.1 Analiza kosztu zamortyzowanego metodą potencjału

(Cormen, rozdział 17.3, strona 419)
Niech Φ_i oznacza potencjał po wykonaniu i-tej operacji. Zakładamy, że $\Phi_0 = 0$, oraz $\Phi_i \geq 0$.
Koszt zamortyzowany i-tej operacji oznaczamy przez:

$$\hat{c}_i = c_i + \Phi_i - \Phi_{i-1}$$

Koszt wykonania n kolejnych operacji:

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

1.2 Analiza kosztu zamortyzowanego dla licznika binarnego metodą potencjałów

(Cormen, 17.1, strona 414-416 i 421–422)

Jako funkcję potencjału wybieramy liczbę jedynek w liczniku.

2 Tablice dynamiczne

(Cormen, 17.4, strona 423–432)
Potrzebujemy tablicy, która umożliwia:

- swobodny (w czasie $O(1)$ dostęp) do wszystkich zapisanych elementów,
- rozszerzenie tablicy o następny element (na jej końcu),
- zmniejszenie tablicy o ostatni element,

Dodawanie nowych elementów możemy wykonać w następujący sposób:

- jeśli są jeszcze wolne miejsca w tablicy to dodajemy nowy element,
- jeśli tablica jest pełna, to alokujemy dwa razy większą tablicę i przepisujemy wszystkie stare elementy do nowej tablicy, oraz dodajemy nowy element.

Usuwanie:

- jeśli współczynnik zapełnianie jest większy niż $1/4$, to usuwamy element,
- jeśli współczynnik zapełnienia spadnie poniżej $1/4$, to alokujemy dwa razy mniejszą tablicę, i przepisujemy tam wszystkie stare elementy (oczywiście oprócz usuwanego).

Potencjał:

- jeśli współcznnik zapełnienia $\geq 1/2$, to $\Phi = 2 * num - size$,
- jeśli współcznnik zapełnienia $< 1/2$, to $\Phi = size/2 - num$,

3 Kolejka za pomocą stosów

Chcemy za pomocą dwóch stosów (+ jeden pomocniczy) symulować kolejkę na której możemy wykonywać następujące operacje:

- dodaj element na początek kolejki,
- dodaj element na koniec kolejki,
- usuń element z początku kolejki,
- usuń element z końca kolejki.

Utrzymujemy dwa stosy, na czubku pierwszego jest koniec kolejki, na czubku drugiego początek kolejki. Dodawanie wykonujemy przez dodanie elementu na odpowiednim stosie. Przy usuwaniu, jeśli odpowiedni stos nie jest pusty, to po prostu zdejmujemy odpowiedni element, wpp. mamy sytuacje, gdzie wszystkie elementy są na jednym stosie. W takim przypadku przy pomocy pomocniczego stosu przenosimy połowę elementów na drugi stos.

4 Rozgłaszanie komunikatów

Dane drzewo T, należy obliczyć czas potrzebny na przesłanie komunikatów do wszystkich węzłów drzewa. Przesłanie komunikatu po jednej krawędzi zajmuje 1 jednostkę czasu.
Algorytm $O(n \log n)$:

- jeśli wierzchołek jest liściem to $czas = 0$,
- wpp. rekurencyjnie oblicz czas potrzebny na rozgłoszenie w poddrzewach,
- posortuj malejąco otrzymane czasy: $t_1, \ldots, t_k$
- $czas = max\{i + t_i : 1 \leq i \leq k\}$

Aby otrzymać algorytm $O(n)$ trzeba sprytnie obliczać wartości atrybutu $czas$.

- $Q = \{$ liście $T \}$,
- while $root \notin Q$ do
  - $x = Q.extractMin()$
  - dodaj $x.czas$ do kolejki $parent(x)$,
  - jeśli $parent(x)$ ma już pełną listę poddrzew, to policz $parent(x).czas$ i dodaj $parent(x)$ do kolejki.

Kolejkę Q można zaimplementować w tablicy (i–ty element tablicy zawiera listę wierzchołków o wartości $x.czas = i$). Sumarycznie operacje $extractMin$ zajmą czas $O(n)$. Dodawanie do kolejki zajmuje czas $O(1)$.

5 Kolorowanie krawędziowe grafów dwudzielnych

Niech G regularny graf dwudzielny o stopniu wierzchołków 2^k.

- podziel graf G na G_1 i G_2, tak że G_1, G_2 są stopnia 2^{k-1} — oblicz cykl Eulera, G_1 otrzymuje krawędzie parzyste, G_2 nieparzyste.
- pokoloruj graf G_1 kolorami $1, \ldots, 2^{k-1}$, pokoloruj graf G_2 kolorami $2^{k-1} + 1, \ldots, 2^k$.

Czas: $T(m) = m + 2 * T(m/2)$, czyli $T(m) = m \log m$.

Algorytm dla dowolnych grafów dwudzielnych

Jeśli graf nie jest regularny, to można, dodając nowe krawędzie i ewentualnie wierzchołki przerobić go na regularny.
Algorytm ma złożoność $O(M(V, E)+E) \log \triangle)$, gdzie $M(V, E)$ to czas potrzebny na obliczenie skojarzenia.

Znajdowanie skojarzenie w czasie $O(E\triangle)$

Każdej krawędzi przyznajemy wagę $w(e)$, początkowo wszystkie wagi są równe 1. W kolejnych krokach prawdziwe będą następujące niezmienniki:

- dla każdej krawędzi e, $0 \leq w(e) \leq \triangle$

Algorithm 1 EDGE-COLORING(G,[a,...,b])

1: **if** $\triangle(G) = 1$ **then**
2: **for all** $e \in E(G)$ **do**
3: $color[e] \leftarrow a$
4: **end for**
5: **else if** $\triangle(G)$ jest parzysta **then**
6: $(G_1, G_2) \leftarrow$ EULER-SPLIT(G)
7: EDGE-COLORING$(G_1, [a, \ldots, \lfloor (a+b)/2 \rfloor])$
8: EDGE-COLORING$(G_2, [\lfloor (a+b)/2 \rfloor + 1, \ldots, b])$
9: **else**
10: $M \leftarrow$ MATCHING(G)
11: **for all** $e \in E(M)$ **do**
12: $color[e] \leftarrow a$
13: **end for**
14: EDGE-COLORING$(G - M, [a+1, \ldots, b])$;
15: **end if**

- dla każdego wierzchołka v, $\sum_{u \in adj(v)} w((v,u)) = \triangle$.

Algorytm kończy działanie, gdy dla każdej krawędzi $w(e) \in \{0, \triangle\}$.

Do analizy złożoności potrzebna jest funkcja potencjału:

$$\Phi = \sum_{e \in E} w(e)^2$$

Dodatkowo wiemy, że $\Phi_0 = n\triangle/2$, a $\Phi_{koniec} = n\triangle^2/2 = m\triangle$.

Po znalezieniu cyklu $C = S \cup T$, potencjał zmienia się o:

$$\sum_{e \in S}((w(e)+1)^2 - w(e)^2) + \sum_{e \in T}((w(e)-1)^2 - w(e)^2) = |S| + |T| + 2(w(S) - w(T)) \geq |C|$$

Algorithm 2 MATCHING(G)

1: $P \leftarrow \emptyset$
2: $M \leftarrow \emptyset$
3: $(C, P) \leftarrow$ DFS-CYCLE(G, P)
4: **while** $C \neq \emptyset$ **do**
5: $(S, T) \leftarrow$ EULER-SPLIT(C)
6: **if** $w(E[S]) \geq w(E[T])$ **then**
7: $(N, H) \leftarrow$ REDISTRIBUTE-WEIGHTS$(S, T, 1)$
8: **else**
9: $(N, H) \leftarrow$ REDISTRIBUTE-WEIGHTS$(T, S, 1)$
10: **end if**
11: $M \leftarrow M \cup N$
12: $G \leftarrow G - (H \cup N)$
13: $(C, P) \leftarrow$ DFS-CYCLE(G, P)
14: **end while**
15: **return** M