Algorytmy i Struktury Danych, 12. ćwiczenia

2020-01-08

Spis treści

1 B-drzewa definicja

- każdy węzeł ma następujące pola n, $c[]$, $key[]$,
- każdy węzeł wewnętrzny utrzymuje n kluczy i $n+1$ wskaźników do synów,
- klucze są uporządkowane rosnąco,
- klucz w poddrzewie $c[i]$ mają wartości pochodzą z przedziału $[key[i-1], key[i]]$ (definiujemy $key[0] = -\infty$, $key[n+1] = \infty$),
- wszystkie liście mają leżą na tej samej głębokości,
- każdy węzeł zawiera nie więcej niż $2t-1$ kluczy,
- każdy węzeł oprócz korzenia zawiera co najmniej $t-1$ kluczy.

B-drzewa usuwanie

- jeśli klucz k jest w węźle x i x jest liściem, to usuń k z węzła,
- jeśli klucz k jest w węźle wewnętrznym x, to:
  - niech y_1 syn x poprzedzający k, y_2 syn x występujący po k, k_1 poprzednik k w drzewie, k_2 następnik k w drzewie,
  - jeśli węzeł y_1 ma co najmniej t kluczy, to rekurencyjnie usuń k_1 i zastąp k przez k_1,
  - w przeciwnym przypadku, jeśli węzeł y_2 ma co najmniej t kluczy, to rekurencyjnie usuń k_2 i zastąp k przez k_2,

- w przeciwnym przypadku, y_1 i y_2 mają po $t-1$ kluczy, scal węzeł y_1, klucz k i węzeł y_2 otrzymując węzeł y', usuń rekurencyjnie k z węzła y.

- jeśli klucz k nie występuje w węźle wewnętrznym x, to:
  - znajdź odpowiednie poddrzewo y w którym może znajdować się k,
  - jeśli y ma co najmniej t kluczy, usuń rekurencyjnie k z y,
  - wpp., jeśli y ma $t-1$ kluczy, ale jeden z sąsiadów y ma t kluczy, to dodaj jeden klucz do y (jeden klucz przechodzi z x do y, jeden z brata y do x),
  - wpp., scal y z dowolnym bratem i usuń k z tak utworzonego węzła (jeśli x jest korzeniem, to może to spowodować zmniejszenie wysokości drzewa).

2 Klasówka 2015 (2), zadanie 2

Niech A będzie skończonym, dynamicznie zmieniającym się ciągiem, którego elementami są liczby ze zbioru $\{-1, 0, 1\}$. Podciąg kolejnych elementów A nazwiemy dobrym, gdy jego suma jest równa zero. Podciąg jest super-dobry, gdy jest dobry i suma elementów w każdym jego prefiksie jest nieujemna.

Przykład: W ciągu A = [1,1,0,-1,0,0,1,-1,1,1,-1], podciąg -1,1,1,-1 jest dobry, ale nie super-dobry. Super-dobrym podciągiem jest na przykład 1,0,-1.

1. Zaproponuj algorytm, który w czasie liniowym obliczy długość najdłuższego super-dobrego podciągu danego ciągu A.

2. Zaproponuj strukturę danych, która pozwoli na wydajne wykonywanie następujących operacji na A:

   - Ini(A):: A := []; //wykonywana tylko raz, na początku
   - Wstaw(A,e,i):: wstaw nowy element e jako i-ty w A, $1 \leq i \leq |A|+1$;
   - Usuń(A,i):: usuń i-ty element z A, $1 \leq i \leq |A|$
   - SuperDobry(A,i,j):: sprawdź, czy podciąg A[i..j] jest super-dobry, $1 \leq i \leq j \leq |A|$;

W każdym zadaniu uzasadnij poprawność swoich rozwiązań i dokonaj analizy złożoności obliczeniowej zaproponowanych algorytmów.

3 Klasówka 2016 (2), zadanie 1

Zaprojektuj strukturę danych, która implementuje standardowe operacje słownika reprezentującego zbiór S liczb całkowitych (Insert, Delete, Find) oraz dodatkowo operację:

- $MaxSubset(d, S)$:: podaj rozmiar maksymalnego podzbioru zbioru S, w którym każde dwie liczby różnią się co najmniej o d, gdzie d jest zadaną z góry stałą całkowitą.

Podaj rozwiązanie dla:

- (4 punkty) d=2,
- (7 punktów) d=10.

Uzasadnij poprawność swoich rozwiązań i przeanalizuj złożoność czasową poszczególnych operacji na strukturze danych względem n (aktualnego rozmiaru struktury).

4 Klasówka 2017 (2), zadanie 1

Niech S będzie skończonym podzbiorem różnych, dodatnich liczb całkowitych, a d dodatnią liczbą całkowitą.

1. (5 punktów) Przyjmij, że elementy zbioru zapisano w uporządkowanej malejąco tablicy a[1..n], gdzie $n = |S|$. Zaprojektuj wydajny algorytm, który policzy liczbę (nieuporządkowanych) par różnych elementów z S, których suma jest nie większa od d. Przykład Dla $S = \{6, 5, 2, 1\}$ i $d = 7$, liczba takich par wynosi 4.
2. (5 punktów) Załóżmy, że d jest ustalone (ale może być bardzo duże), natomiast S jest zbiorem dynamicznym. Zaprojektuj efektywną strukturę danych, która umożliwi wydajne wykonywanie na zbiorze S następujących operacji:
   - Ini(S):: $S := \emptyset$; // operacja wykonywana raz, na samym początku obliczeń;
   - Insert(x, S):: $S := S \cup \{x\}$;
   - Delete(x, S):: $S := S - \{x\}$;
   - Pairs(S):: return liczba par różnych elementów z S, których suma jest nie większa od d.

Zaproponuj rozwiązanie, w którym operacje Insert i Delete są wykonywane w czasie $O(\log |S|)$, a operacja Pairs w czasie stałym, niezależnie od wielkości d. Możesz przyjąć, że operacje arytmetyczne i porównania na liczbach są wykonywane w stałym czasie.