Lista 5

Zadanie 3.

Zaproponuj strukturę danych Q dla dynamicznych zbiorów liczb, w której można wykonywać operacje $Min - Luka$ wyznaczającą odległość między dwoma najbliższymi sobie liczbami w Q. Jeśli np. $Q = \{1, 5, 9, 15, 18, 22\}$, tp $Min - Luka(Q)$ daje w wyniku $18 - 15 = 3$. Zaimplementuj jak najefektywniej operacje $Insert$, $Delete$, $Search$ oraz $Min - Luka$ i wykonaj analizę ich złożoności czasowej.

1. **Rozwiązanie.**

Wybrałem drzewo czerwono-czarne, aby operacje takie jak $Insert, Delete, Search$ były jak najbardziej efektywne dzięki zrównoważonej budowie drzewa – złożoność czasowa $\log(n)$. Jednakże zwykłe drzewo czerwono-czarne nie wystarczy, aby móc w mało kosztowny sposób wyznaczyć najmniejszą odległość między liczbami w strukturze Q. Wybrane drzewo czerwono-czarne będzie musiało przechowywać dodatkowe informacje – *informacje agregujące* – w każdym węźle, opisując całe poddrzewo. Każdy węzeł x w naszej wzbogaconej strukturze będzie miał następujące atrybuty:

- parent – wskazanie na węzeł rodzica(RBT)
- key – wartość kluczowa liczby (RBT) kay i value to to samo
- value – wartość w węźle (RBT)
- left – wskazanie na lewy węzeł potomstwa (RBT)
- right – wskazanie na prawy węzeł potomstwa (RBT)
- color – kolor węzła (RBT)
- min – minimalna wartość w poddrzewie,
  gdzie x to korzeń poddrzewa (informacja agregująca)
- max – maksymalna wartość w poddrzewie,
  gdzie x to korzeń poddrzewa (informacja agregująca)
- min-luka – minimalna luka pomiędzy liczbami w poddrzewie,
  gdzie x to korzeń poddrzewa (informacja agregująca)

Przyjmujemy, że wartość minimalnej przerwy na liściu to ∞ (nieskończoność).

Wyznaczenie minimalnej wartości w poddrzewie z korzeniem x, to minimum w lewym poddrzewie, jeśli istnieje, lub zwracamy wartość w węźle x. Wartość maksymalna jest wyznaczana analogicznie, szukamy maksimum w prawym poddrzewie, jeśli istnieje, lub zwracamy wartość w węźle x. Jest to konsekwencja faktu, że drzewo czerwono-czarne ma własności drzewa BST (*Binary Search Tree*). A zatem wartości atrybutów min i max wyznaczane są na podstawie wartości atrubutów w danym węźle i jego synach

Operacja na wyznaczenie minimalnej luki dla węzła x wygląda następująco:

$$minLuka(x) = min \begin{cases} minLuka(x.left) \\ minLuka(x.right) \\ |x.value - x.left.max| \\ |x.value - x.right.min| \end{cases}$$

Zasadniczo mamy dwa przypadki:

1. Wartość w węźle x nie jest jednym z elementów „tworzących" minimalną lukę, to $minLuka(x)$ będzie równa mniejszej wartości min-luka w lewym lub prawym poddrzewie.
2. Wartość min-luka to różnica wartości w węźle x i wartości jego poprzednika lub następnika.

Dlatego, aby wyliczyć minimalną lukę pomiędzy elementami należy wziąć minimum ze wszystkich czterech możliwych wartości.

Koszt obliczenia dodanych atrybutów jest stałe – $O(1)$ – przez co nie wpływają asymptotycznie na wydajność i działanie całej struktury. Dzięki temu operacje $Insert$ i $Delete$ nadal mają koszt $O(\log n)$ jak w przypadku zwykłego drzewa RBT. Oczywiście przy samoorganizacji drzewa (rotacje), informacje agregujące wymagają zaktualizowania, ale ich obliczenia również są stałego kosztu. Wynika to z faktu, że atrybuty: min, max oraz min-luka w danym węźle x wyliczane są wyłącznie na podstawie atrybutów samego węzła x oraz jego dzieci. Informacje agregujące w węzłach nie wpływają na wyszukiwanie danych w strukturze opartej na drzewie czerwono-czarnym, więc jego koszt również jest $O(\log n)$. Natomiast dzięki temu, że drzewo przechowuje w każdym węźle dane o minimalnej odległości pomiędzy liczbami, to wywołanie operacji $Min\_Luka(Q)$, to wyciągnięcie wartości z atrybutu $min - luka$ z korzenia drzewa, które jest stałego kosztu czasu $O(1)$.