

Wstęp do Algorytmów

Kierunek: Inżynieria Systemów

Semestr Letni – 2021/2022

Lista 4 – Drzewa Binarne, Kolejki, Stosy

Aby uzyskać ogólną (ale sensowną) wiedzę o strukturach danych warto zajrzeć do *Wprowadzenia do Algorytmów* Cormena. Lista korzysta z pojęć **kolejki** i **stosu** – struktur danych łatwych w implementacji w oparciu o listy.

Kolejka to taka specjalna lista o której myślimy, że ma początek i koniec (dość naturalne dla list). Często do początku kolejki odnosimy się, korzystając z pojęcia *head*, a do końca kolejki, korzystając z pojęcia *tail*. Do kolejki nowe elementy dodajemy zawsze na jej koniec (czyli wstawiamy za stary ogon), a jeśli chcemy coś z niej zabrać, to zabieramy z początku (usuwamy starą głowę). Dokładnie tak jak w uczciwej kolejce sklepowej, gdzie nowi klienci trafiają na koniec, a obsługiwani (i usuwani z kolejki) są ci, którzy dotarli na jej początek. Obowiązuje to tak zwana zasada FIFO, tzn. *First In First Out*.

Stos to taka specjalna lista, o której myślimy że jest ułożona na podłodze i jeden element leży na drugim. Stos ma więc szczyt stosu (*stack top*). Nowe elementy dokładamy na wierzch stosu (w pewnym sensie możemy o tym myśleć jak o dokładaniu elementów na początek kolejki, zamiast na jej koniec), a obsługiwane elementy zabieramy z wierzchu stosu. Mówimy, że w stosie obowiązuje zasada LIFO (*Last In First Out*). W materiałach operację dodawania do stosu standardowo nazywa się *push*, a operację zabierania elementu ze stosu nazywa się *pop*.

Ze stosów i kolejek korzystamy między innymi podczas przeszukiwania drzew. Zapoznaj się z PDFami do wykładu, dotyczącymi drzew i ich przeszukiwania, tj. przeszukiwania w głąb (DFS - Depth First Search) oraz przeszukiwania wszerz (BFS - Breadth First Search).

Zwróć uwagę, że korzystaliśmy z przeszukiwania w głąb między innymi w poprzednim semestrze w algorytmie sprawdzania tautologiczności formuł metodą opartą o sekweny Gentzena. Szliśmy w głąb w poszukiwaniu interesującego nas węzła. Jeśli w danej gałęzi się nie udało, zaczynaliśmy próbować w innej.

Przeszukiwanie wszerz kieruje się inną intuicją. Tam próbujemy znaleźć wierzchołek o interesujących nas własnościach, który jest jak najbliżej naszego wierzchołka startowego (w przypadku drzew zazwyczaj wierzchołkiem startowym jest korzeń).

- (4 pkt) Skorzystaj z udostępnionego kodu, by zaimplementować drzewo binarne (o głębokości 3 lub 4), a następnie:**
 - Zaimplementuj metodę, która wyświetla listę kolejno odwiedzonych wierzchołków drzewa w przeszukiwaniu wszerz (BFS – *breadth first search*). W tym celu skorzystaj ze struktury kolejki (odpowiednio modyfikując znany wcześniej kod listy jednokierunkowej).
 - Zaimplementuj metodę, która wyświetla listę kolejno odwiedzonych wierzchołków drzewa w przeszukiwaniu w głąb (DFS – *depth first search*). W tym celu skorzystaj ze struktury stosu (odpowiednio modyfikując znany wcześniej kod listy jednokierunkowej).
- (3 pkt) Czasami chcemy przejrzeć drzewo binarne począwszy od innego wierzchołka. Zwróć uwagę, że jeśli pewien liść drzewa binarnego uznamy za korzeń nowego drzewa i pozostawimy wszystkie połączenia, uzyskamy nowe drzewo binarne.**
 - Zaimplementuj metodę, która dla wskazanego liścia starego drzewa binarnego tworzy nowe (w pewnym sensie zachowujące strukturę) drzewo binarne, gdzie korzeniem jest zadany liść starego drzewa.
 - Zwróć uwagę na fakt, że nie zawsze da się to dla danego wierzchołka zrobić. Czy może to mieć jakiś związek z pełnością drzewa binarnego?
- (3 pkt) Zaimplementuj drzewo binarne czteropoziomowe niepełne (nie zapomnij jakoś zapewnić, że mimo niepełności będą cztery poziomy).**
 - W zaproponowany sposób ‘narysuj’ uzyskane drzewo.
 - Wyświetl liczbę węzłów/wierzchołków na każdym z poziomów drzewa. Wyświetl liczbę liści.
 - Wyznacz długość najkrótszej ścieżki od korzenia do liścia. Wyświetl wszystkie liście znajdujące się na tej głębokości.