Pewne konstrukcje drzew binarnych

Wojciech Rytter*† Sebastian Smyczyński†

Najciekawszymi zagadnieniami związanymi ze strukturą drzew binarnych są problemy optymalizacyjne, w których dla podanego ciągu kluczy wraz ze skojarzonymi wagami należy skonstruować drzewo spełniające określone warunki o minimalnym koszcie. Koszt drzewa definiowany jest jako suma kosztów wierzchołków, w których umieszczone są klucze, zaś koszt wierzchołka jest proporcjonalny do głębokości wierzchołka w drzewie i wagi umieszczonego w nim klucza.

Gdy ograniczymy się do binarnych drzew poszukiwań (BST, z ang. Binary Search Tree), historycznie jako pierwszy rozpatrywany był problem konstrukcji optymalnego drzewa alfabetycznego (OABT, z ang. Optimal Alphabetic Binary Tree). Drzewo alfabetyczne jest szczególnym przypadkiem drzewa BST, w którym wszystkie klucze znajdują się w liściach.

W roku 1971 Donald Knuth uogólnił ten problem dla konstrukcji optymalnych drzew BST (OBST) i zaproponował algorytm rozwiązujący ogólny problem OBST działający w czasie $O(n^2)$.

Do dnia dzisiejszego jedynymi znanymi algorytmami konstrukcji OBST są algorytmy oparte na programowaniu dynamicznym i o złożoności $O(n^2)$.

Inaczej wygląda postęp w badaniach nad problemem konstrukcji OABT. W 1971 roku Hu oraz Tucker podali algorytm tworzący OABT w czasie $O(n \log n)$. Kilka lat później Garsia i Wachs podali prostszy algorytm na OABT również o złożoności $O(n \log n)$. Dalszą poprawę złożoności czasowej algorytmu konstrukcji OABT przynosi dopiero praca Larmore i Przytyckiej z roku 1998 , w której podano nowy algorytm oparty na algorytmie Garcia-Wachsa i używający ekstensywnie struktury drzewa kartezjańskiego.

Ostatecznie problem OABT doczekał się liniowego algorytmu w 2005 roku, gdy Hu, Larmore oraz Morgenthaler podali algorytm konstrukcji OABT dla dziedzin wag wejściowych, które mogą być posortowane w czasie liniowym.

Intrygujący jest fakt, iż wszystkie wspomniane algorytmy konstrukcji OABT wykorzystują ten sam schemat postępowania. Najpierw tworzone jest drzewo pośrednie, nie będące drzewem alfabetycznym, z którego odczytywane są głębokości liści, a następnie przy użyciu tej informacji rekonstruowane jest docelowe drzewo alfabetyczne.

Ciekawa jest również rozbieżność złożoności algorytmów konstrukcji OBST i OABT.

W poszukiwaniu bardziej efektywnego algorytmu dla OBST napotkaliśmy następujący problem rekonstrukcyjny: mając dany posortowany ciąg n kluczy oraz ciąg indeksów wyznaczający kolejność wstawiania tych kluczy do początkowo pustego binarnego drzewa wyszukiwań, zrekonstruować ostateczną postać tego drzewa po wykonaniu wszystkich wstawień.

Oczywisty, naiwny algorytm wstawiający klucze według podanej kolejności ma złożoność $O(n^2)$. Stosując pomocnicze struktury danych (np. drzewo AVL) jesteśmy w stanie poprawić złożoność naiwnego algorytmu do $O(n \log n)$.

Podany przez nas liniowy algorytm jest wyjątkowo prosty i co więcej definiuje pewien schemat postępowania, który może zostać użyty do rozwiązania większej klasy problemów.

Pokażemy, że przy drobnej modyfikacji algorytm ten dla dowolnej permutacji konstruuje odpowiadające jej drzewo kartezjańskie (w sensie minimum lub maksimum). Ponadto dla pewnych szczególnych ciągów wag pokażemy, jak problem OBST można rozwiązać w czasie liniowym.

*Instytut Informatyki, Uniwersytet Warszawski, ul. Banacha 2, 02-097 Warszawa
†Wydział Matematyki i Informatyki, Uniwersytet Mikołaja Kopernika, ul. Chopina 12/18, 87-100 Toruń