		10000			15000			20000			25000			30000		
		BST	AVL	Linked List	BST	AVL	Linked List	BST	AVL	Linked List	BST	AVL	Linked List	BST	AVL	Linked List
random	Insert	0.0016	0.0087	0.2443	0.0025	0.0132	0.8076	0.0034	0.0175	1.7049	0.0044	0.2166	2.8038	0.0058	0.0266	4.1744
	Search	0.0014	0.0013	0.2746	0.0024	0.0021	0.9987	0.0033	0.0030	2.0290	0.0044	0.0039	3.2697	0.0057	0.0051	4.7081
	Delete	0.0015	0.0019	0.2619	0.0027	0.0034	0.9123	0.0039	0.0051	1.8171	0.0051	0.0071	2.8338	0.0068	0.0096	4.0592
ordered	Insert	0.0021	0.0082	0.2258	0.0032	0.0124	0.6257	0.0042	0.1740	1.3091	0.0063	0.0210	2.2227	0.0056	0.0256	3.4141
	Search	0.0017	0.0009	0.1966	0.0027	0.0014	0.5427	0.0034	0.0020	1.0846	0.0055	0.0025	1.7789	0.0049	0.0034	2.6982
	Delete	0.0010	0.0017	0.1023	0.0017	0.0030	0.2818	0.0025	0.0043	0.5577	0.0031	0.0057	0.8774	0.0040	0.0073	1.2830

Wnioski: Lista liniowa jest najwolniejsza, wynika to z tego że pojedyncze operacje mają złożoność liniową w przeciwieństwie do drzew, gdzie te same operacje mają złożoność logarytmiczną.

Dla losowych danych drzewo BST jest kilka razy szybsze przy wstawianiu i ok. 25% szybsze przy usuwaniu, jednak operacje wyszukiwania są o ok. 10% szybsze dla drzewa AVL więc w sytuacjach gdzie dominującą operacją jest wyszukiwanie, użycie drzewa AVL może być opłacalne.

Dla danych częściowo uporządkowanych drzewo BST staje się mniej zbalansowane przez co traci na wydajności ale dalej zachowuje przewagę dla operacji usuwania I wstawiania, jednak drzewo AVL staje się prawie dwukrotnie szybsze od drzewa BST w wyszukiwaniu.

Dominik Kowalczyk, kd36082