探索

3I11 鵜飼笑太

リニアサーチ

配列の先頭から順番に探していく力こそ正義なアルゴリズム

バイナリサーチ

配列を始めにソートし、二分探索を用いて探していくスマートなアルゴリズム

演習の結果＆考察

データ	**20293**	**7789**	**4021**	**6586**	**30000**
リニアサーチ	-	-	-	-	-
10	○：走査回数(1)	×：走査回数(10)	×：走査回数(10)	×：走査回数(10)	×：走査回数(10)
100	○：走査回数(1)	○：走査回数(11)	×：走査回数(100)	×：走査回数(100)	×：走査回数(100)
1000	○：走査回数(1)	○：走査回数(11)	○：走査回数(101)	×：走査回数(1000)	×：走査回数(1000)
10000	○：走査回数(1)	○：走査回数(11)	○：走査回数(101)	○：走査回数(1001)	×：走査回数(10000)
バイナリサーチ	-	-	-	-	-
10	○：走査回数(2)	×：走査回数(5)	×：走査回数(5)	×：走査回数(5)	×：走査回数(5)
100	○：走査回数(7)	○：走査回数(7)	×：走査回数(8)	×：走査回数(7)	×：走査回数(7)
1000	○：走査回数(9)	○：走査回数(7)	○：走査回数(10)	×：走査回数(11)	×：走査回数(11)
10000	○：走査回数(13)	○：走査回数(13)	○：走査回数(12)	○：走査回数(12)	×：走査回数(14)

結果から、計算量が概ね想定通りであることが確認できた

ハッシュ法

あるデータをハッシュ関数を用いて決める特定の場所に保存し、検索する際はそのハッシュ関数を使い格納場所を特定するアルゴリズム

この方法ではデータが違うのにハッシュ関数の出力が同じになってしまい衝突(コリジョン)が発生する可能性がある。この解決方法として、再ハッシュを行うオープンアドレス法と連結リストとして繋いでいくチェイン法がある

意見・感想

リニアサーチはデータ数が少ない(10個以下)なら十分速く、ソートするときの操作を考えるとバイナリサーチより有効だと思った

今回は演習でやらなかったハッシュ法はものすごく速そうだったがハッシュ関数の計算が複雑だと時間がかかりそうで調整する必要がありそうだと思った