宿題2

・大規模なデータベースになると、ハッシュテーブルの大きさ自体も大きくなり再ハッシュが多くなる。またハッシュテーブルの中にハッシュテーブルという形も限界がくる

→大規模なデータベースだとハッシュテーブルにたどり着いて深さを辿るのにかかる計算量がO(logN)よりも大きくなることがありそう...?

→深さを辿っても結局の平均計算量はO(1)

オフィスアワー後まとめ

・スライドにあったパフォーマンスのグラフについて、」mostlyO(1)の結果で何個か時間がかかるものがあった

→再ハッシュしているため時間がかかる

→Nが大きくなっていくほど、時間がかかってしまうケース（処理が遅いなと感じる人・回数）が増加してしまう。

挑戦クイズ

常にO(1)で検索できるデータベースはあるか？

宿題3

・ハッシュテーブルの中に木構造？🤔

ハッシュテーブルの下に木構造をつくる

→左が古いもの、右が新しいもの

→新しいものが入ってきたら1番左の枝を切り落とす

💡これだとハッシュテーブルの意味がない？

テーブルが違ったら比較できない

オフィスアワー後まとめ

ノードを前後両方から辿れるようにする（ノードをリストみたいにインデックス管理できるような）

a <-> b <-> c <-> d <-> e

先頭のノード（新しいもの）と最後のノード（古いもの）を記憶しておく

与えられたものがあるかどうかを確認して、

あったら→もとのノードを消して、先頭にもってくる

ないと→最後のノードを消去、先頭に追加