Travis Henderson

CSS342A

Program 5 Analysis

16/05/17

**(2a)**

hendertj@uw1-320-10:~$ g++ driver.cpp

hendertj@uw1-320-10:~$ ./a.out

#faculty members: 10

contents:

-inf -inf -inf -inf -inf -inf

berger berger berger

cioch

erdly erdly erdly erdly erdly

fukuda

jackels

olson olson olson

stiber

sung

unknown unknown

zander zander

+inf +inf +inf +inf +inf +inf

deleting unknown

#faculty members: 9

contents:

-inf -inf -inf -inf -inf -inf

berger berger berger

cioch

erdly erdly erdly erdly erdly

fukuda

jackels

olson olson olson

stiber

sung

zander zander

+inf +inf +inf +inf +inf +inf

finding stiber = 1

create another list

finding stiber = 1

#faculty members: 9

cost of find = 104

**(2b)**

dlist's find cost = 6491439

mtflist's find cost = 66448

translist's find cost = 6422436

skip's find cost = 2078098

**(2c)**

The results honestly surprise me. I expected the skip list to have the lowest cost given that it essentially makes finding on it O(log(n)).

However, seeing doubly linked list cost the most makes sense to me. Given however many times something is accessed, dll does nothing to make the elements accessible, it only keeps them in a list.

I guess it’s not that crazy to see MTF list with the least cost given that it moves the most recently accessed index to the top. Since statistics.cpp finds the items that it just accessed with a higher probability than all other indexes, it makes sense then that it would have the lower cost. Also given how translist uses a similar idea as MTF, I can see how it also did well on the given statistics.cpp. Not as good as MTF, though, since it only gradually moves accessed indexes towards the top.

However, if we were just finding the cost of finding an index on a random list, I would expect skip list to be the winner. But, again, since statistics is based off of finding an item based on how often it is accessed, it makes sense that MTF does come out on top to me. Or rather, I’ve convinced myself that it makes sense!