科目名: プログラミング実習III

課題: B6

番号: 27020856

氏名: 趙 聖化

作成: 2023年11月12日

----------------------------------------

[課題 6.1]

結果：

static list\_node\_t \*insert\_node(list\_node\_t \*n, int val)

*/\* n の指す節点の直後に, 値 val を保持する新しい節点を挿入し,*

*その節点へのポインタを返す \*/*

{

list\_node\_t \*new\_node = create\_node(val);

*/\* ノード n (nの指す節点) の直後に挿入 (この部分を埋めよ) \*/*

new\_node->next = n->next;

n->next = new\_node;

*return* new\_node;

}

void list\_print(list\_node\_t \*head\_p)

*/\* リストの内容を表示する \*/*

{

list\_node\_t \*p = head\_p->next; *// 헤드 노드 다음의 첫 번째 노드부터 시작*

printf("[");

*for* (; p != NULL; p = p->next)

{

printf(" %d ", p->val);

}

printf("]");

printf("\n"); *// 개행을 통해 출력을 마무리합니다.*

}

input = 3

[ 3 ]

input = 5

[ 5 3 ]

input = 4

[ 4 5 3 ]

input = 3

[ 3 4 5 3 ]

input = %

chosunghwa@joseonghwaui-MacBookAir 6-1 %

感想：

Insert\_nodeを利用してリストの中に新しいノードを追加する方法を学ぶことができました。

list\_node\_t \*p = head\_p->next

head\_p -> nextから始まらないと最初にhead\_pを作った時のダミーデータから出力されるのもわかりました。

----------------------------------------

[課題 6.2]

結果：

list\_node\_t \*list\_find(list\_node\_t \*head\_p, int val)

*/\* リスト中に値が val の節点を探す*

*見つかればその節点のポインタを, なければ NULL を返す \*/*

{

list\_node\_t \*p = head\_p->next;

*for* (; p != NULL; p = p->next)

{

*if* (p->val == val)

{

*return* p;

exit(0);

}

}

*return* NULL;

}

chosunghwa@joseonghwaui-MacBookAir 6-2 % ./a.out

input = 3

[ 3 ]

input = 4

[ 4 3 ]

input = 5

[ 5 4 3 ]

input = 3

[ 5 4 -3 ]

input = 4

[ 5 -4 -3 ]

input = -4

[ 5 4 -3 ]

input = -3

[ 5 4 3 ]

input = %

chosunghwa@joseonghwaui-MacBookAir 6-2 %

感想：

リストの中をfor文を使って入力した数字を探す方法を学ぶことができました。

----------------------------------------

[課題 6.3]

結果：

list\_node\_t \*list\_insert\_uniq(list\_node\_t \*n, int val)

{

list\_node\_t \*p = list\_find(n, val);

*if* (p == NULL)

{

list\_insert(n, val);

}

*return* p;

}

chosunghwa@joseonghwaui-MacBookAir 6-3 % ./a.out

input = 3

[ 3 ]

input = 5

[ 5 3 ]

input = 3

[ 5 3 ]

input = 4

[ 4 5 3 ]

input = 4

[ 4 5 3 ]

input = %

chosunghwa@joseonghwaui-MacBookAir 6-3 %

感想：

List\_insert\_uniqの中で先ほど作成したlist\_insertを使って重複検査をする方法を真びました。

----------------------------------------

[課題 6.4]

hosunghwa@joseonghwaui-MacBookAir 6-4 % ./a.out

n: 5000

(1) cpu = 0.037608 [sec]

(2) cpu = 0.025475 [sec]

(3) cpu = 0.01964 [sec]

chosunghwa@joseonghwaui-MacBookAir 6-4 % ./a.out

n: 10000

(1) cpu = 0.104627 [sec]

(2) cpu = 0.064973 [sec]

(3) cpu = 0.06486 [sec]

chosunghwa@joseonghwaui-MacBookAir 6-4 % ./a.out

n: 15000

(1) cpu = 0.180666 [sec]

(2) cpu = 0.144198 [sec]

(3) cpu = 0.144145 [sec]

chosunghwa@joseonghwaui-MacBookAir 6-4 % ./a.out

n: 20000

(1) cpu = 0.311885 [sec]

(2) cpu = 0.255846 [sec]

(3) cpu = 0.255473 [sec]

chosunghwa@joseonghwaui-MacBookAir 6-4 % ./a.out

n: 25000

(1) cpu = 0.435129 [sec]

(2) cpu = 0.39849 [sec]

(3) cpu = 0.396034 [sec]

chosunghwa@joseonghwaui-MacBookAir 6-4 %

感想：

（２）、（３）と見ればわかるように乱数が異なる乱数でもかかる時間がほぼ一緒だと思われ、乱数と計算時間とは関係性が低いと考えられました。

また、リストに新しいデータを挿入する際、全体を走査して重複がないか確認するため、リストが長くなればなるほど挿入に時間がかかると思います。

----------------------------------------

[課題 6.5]

結果：

chosunghwa@joseonghwaui-MacBookAir 6-5 % gcc list.c list5test.c

chosunghwa@joseonghwaui-MacBookAir 6-5 % ./a.out

input = 3

[ 3 ]

input = 5

[ 3 5 ]

input = 3

[ 5 ]

input = 4

[ 5 4 ]

input = 4

[ 5 ]

input =

感想：

普通にremove(p)をしたら一番前のリストが削除してしまい

その値を持つ節点の直前の節点を指すポインタ pを求めることで一番工夫しました

----------------------------------------

[課題 6.9]

結果：

chosunghwa@joseonghwaui-MacBookAir 6-9 % gcc tree.c tree9test.c

chosunghwa@joseonghwaui-MacBookAir 6-9 % ./a.out

input = 3

[ 3 ]

input = 5

[ 3 5 ]

input = 3

[ 3 5 ]

input = 4

[ 3 4 5 ]

input =

感想：

二分木探索のアルゴリズムに関して知ることができました。

----------------------------------------

[課題B6の感想]

前回のポイントに加えてもっと難しいリスト処理に関して学び、リストというのを知ることができました。特にリスト処理を通じて二分木探索アルゴリズムに関して勉強ができたのが一番興味深いでした。