NCKU PD1 HW10

學號：H24134055
姓名：陳韋綸
系級：統計117

前置Functions

genNode()

將輸入的 IP 數據和其他參數存入節點結構

- 邏輯

– 分配記憶體並初始化節點屬性。

– 如果有前驅節點，則更新鏈表結構。

– 返回新創建的節點。

```
Prefix *genNode(Prefix *prev, unsigned data, unsigned a, unsigned char len)
{
    Prefix *node = malloc(sizeof(Prefix));
    (node->ip)[FULL] = data; (node->ip)[MSB_8] = a;
    node->len = len; node->next = NULL;
    if (prev != NULL) prev->next = node;
    return node;
}
```

input()

```
Prefix *input(char filepath[])
{
    FILE *fp; fp = fopen(filepath, "r");
    if (fp == NULL) printf("bad\n");
    char format[30]; fscanf(fp, "%s", format);
    unsigned a, b, c, d; unsigned char k;
    sscanf(format, "%u.%u.%u.%u/%hhu\n", &a, &b, &c, &d, &k);
    unsigned num = (a<<24)|(b<<16)|(c<<8)|d;
    Prefix *head = genNode(NULL, num, a, k), *prev = head;
    strcpy(head->format, format);
    //print_bit(num);
    //printf(" is %u.%u.%u.%u/%hhu\n", a, b, c, d, k);
    while (fscanf(fp, "%s", format) != EOF){
        sscanf(format, "%u.%u.%u.%u/%hhu", &a, &b, &c, &d, &k);
        num = (a<<24)|(b<<16)|(c<<8)|d;
        Prefix *node = genNode(prev, num, a, k);
        prev = node; strcpy(node->format, format);
        //print_bit(num);
        //printf(" is %u.%u.%u.%u/%hhu\n", a, b, c, d, k);
    }
    fclose(fp);
    return head;
}
```

讀取 `routing_table.txt` 文件，解析其中的 `a.b.c.d/k` 格式資料，並將其轉換為包含以下內容的節點，藉genNode()建構成Linkedlist

- 32 位元的整數型態表示 (`a.b.c.d` 經過位移和 OR 運算轉換而來)。
- `a` 值 (即最左側 8 位元，MSB 端)。
- 預計匹配的 Prefix 長度 (`k`)。
- 原始字串。

insert()

```c
void insert(Prefix *node, Prefix **head)
{
    if (node == NULL) return;
    Prefix **x = head;
    while ((*x) != NULL && ((*x)->ip)[FULL] < (node->ip)[FULL])
        x = &((*x)->next);
    node->next = (*x);
    (*x) = node;
}
```

將節點插入鏈表，並保證鏈表按 IP 升冪排序

邏輯

-遍歷鏈表找到適合位置。

-插入節點並保持鏈表有序。

delete()

```
void delete(unsigned full_ip, Prefix **head)
{
    if (*head == NULL || head == NULL) return;
    Prefix **x = head;
    while (*x != NULL){
        if ((*x)->ip[0] == full_ip){
            Prefix *temp = *x;
            *x = (*x)->next;
            free(temp);
        }else
            x = &((*x)->next);
    }
}
```

從鏈表中刪除與指定 IP 匹配的節點

邏輯:

- 遍歷鏈表找到目標節點。
- 移除節點並釋放記憶體。

group_len()

```
Prefix **group_len(Prefix **head)
{
    Prefix **g = calloc(33, sizeof(Prefix*));
    while (*head != NULL){
        Prefix *tmp = *head; *head = (*head)->next;
        tmp->next == NULL; insert(tmp, g+tmp->len);
    }
    return g;
}
```

將所有節點按 Prefix 長度 (`k`) 分類

處理流程:

- 建立一個長度為 33 的指標陣列 (`g`)。

- 遍歷節點，藉由insert()按 `k` 將其插入對應的鏈表。

prefix_search()

讀取目標 IP 地址文件，並依次對目標進行搜索

處理流程:

- 遍歷文件中的每個目標地址。
- 調用search()進行匹配。

```
void prefix_search(char filepath[], Prefix ***G)
{
    FILE *fp; fp = fopen(filepath, "r");
    unsigned a, b, c, d;
    while (fscanf(fp, "%u.%u.%u.%u", &a, &b, &c, &d) != EOF){
        search(a, b, c, d, G);
    }
    fclose(fp);
}
```

match()

```
short match(unsigned aim, unsigned reference, unsigned char len)
{
    unsigned i = (1<<31); short check = SUCCESS;
    for (int _ = 0; _ < len; _++, i >>= 1){
        if ((aim & i) != (reference & i)){
            check = FAILED; break;
        }
    }
    return check;
}
```

對目標IP進行prefix matching

邏輯

- 按 MSB 比較兩個 IP 地址的前 `k` 位是否相同。
- 匹配成功則返回 1，否則返回 0。

核心Functions

segment()

```
Prefix ***segment(Prefix **head)
{
    Prefix ***G = calloc(33, sizeof(Prefix**)), **g = group_len(head);
    for (int i = 0; i < 33; i++){
        if (g[i] != NULL){
            G[i] = calloc(256, sizeof(Prefix*));
            while (g[i] != NULL){
                Prefix *tmp = g[i]; g[i] = g[i]->next;
                tmp->next == NULL; insert(tmp, &G[i][(tmp->ip)[MSB_8]]);
            }
        }
    }
    return G;
}
```

依prefix length和MSB前8位對routing table做分類

- 檢查從group_len()得來的指標陣列，若該prefix length的欄位有prefix，則生成長度為 256 的指標陣列

- 將節點按 `a` 值分類到對應鏈表中，保證升冪排列，方式為調用insert()

prefix_insert()

讀取 `inserted_prefixes.txt` 文件，按 `a` 和 `k` 分類節點並插入到對應位置

處理流程:

- 讀取每條記錄，解析出 `a.b.c.d/k`。
- 利用 insert()將節點插入對應鏈表。

```
void prefix_insert(char filepath[], Prefix ***G)
{
    FILE *fp; fp = fopen(filepath, "r");
    char format[30];
    unsigned a, b, c, d; unsigned char k;
    while (fscanf(fp, "%s", format) != EOF){
        sscanf(format, "%u.%u.%u.%u/%hhu", &a, &b, &c, &d, &k);
        Prefix *p = genNode(NULL, (a<<24)|(b<<16)|(c<<8)|d, a, k);
        strcpy(p->format, format);
        insert(p, &G[p->len][(p->ip)[MSB_8]]);
    }
    fclose(fp);
}
```

prefix_delete()

從 `deleted_prefixes.txt` 中讀取需要刪除的前綴，並刪除對應的節點

處理流程:

- 讀取 `a.b.c.d/k`，找到對應的 `G[k][a]`。
- 調用 delete()進行刪除。

```
void prefix_delete(char filepath[], Prefix ***G)
{
    FILE *fp; fp = fopen(filepath, "r");
    unsigned a, b, c, d; unsigned char k;
    while (fscanf(fp, "%u.%u.%u.%u/%hhu", &a, &b, &c, &d, &k) != EOF)
        delete((a<<24)|(b<<16)|(c<<8)|d, &G[k][a]);
    fclose(fp);
}
```

search()

```
void search(unsigned a, unsigned b, unsigned c, unsigned d, Prefix ***G)
{
    short check = 0;
    for (int i = 0; i < 33; i++){
        if (G[i] != NULL){
            for (Prefix *p = G[i][a]; p != NULL; p = p->next){
                if (match((a<<24)|(b<<16)|(c<<8)|d, p->ip[0], p->len)){
                    printf("%u.%u.%u.%u matched successfully\n", a, b, c, d);
                    check = 1; break;
                }
            }
            if (check) break;
        }
    }
    if (!check) printf("%u.%u.%u.%u failed to match\n", a, b, c, d);
}
```

搜索是否有與目標地址匹配的前綴

處理流程:

- 遍歷G中所有的Linkedlist

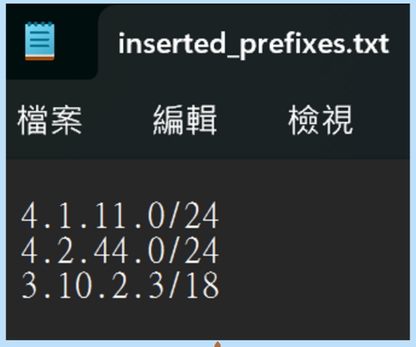
- 按 `k` 決定匹配長度，依次調用match()進行匹配。

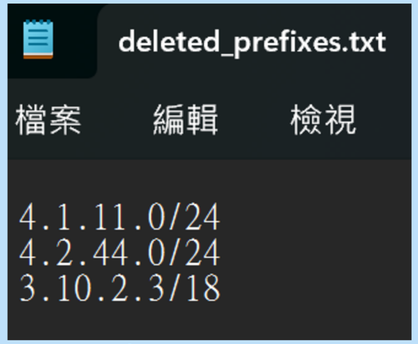
- 匹配成功則輸出對應信息，否則輸出失敗信息。

成果

main()

```c
int main()
{
    char routing_table_filepath[] = "C:\\Users\\michu\\Downloads\\hw10-2024-12-26\\hw10-2024-12-26\\routing_table.txt";
    char inserted_prefixes_filepath[] = "C:\\Users\\michu\\Downloads\\hw10-2024-12-26\\hw10-2024-12-26\\inserted_prefixes.txt";
    char deleted_prefixes_filepath[] = "C:\\Users\\michu\\Downloads\\hw10-2024-12-26\\hw10-2024-12-26\\deleted_prefixes.txt";
    char trace_file[] = "C:\\Users\\michu\\Downloads\\hw10-2024-12-26\\hw10-2024-12-26\\trace_file.txt";
    Prefix *head = input(routing_table_filepath);
    Prefix ***G = segment(&head);

    printf("-------initial routing table-------\n");
    print_illustration(G); printf("\n");
    printf("-------routing table after insertion-------\n");
    prefix_insert(inserted_prefixes_filepath, G);
    print_illustration(G); printf("\n");
    printf("-------routing table after insertion and deletion-------\n");
    prefix_delete(deleted_prefixes_filepath, G);
    print_illustration(G); printf("\n");
    printf("-------address matches-------\n");
    prefix_search(trace_file, G);

    return 0;
}
```





檔案

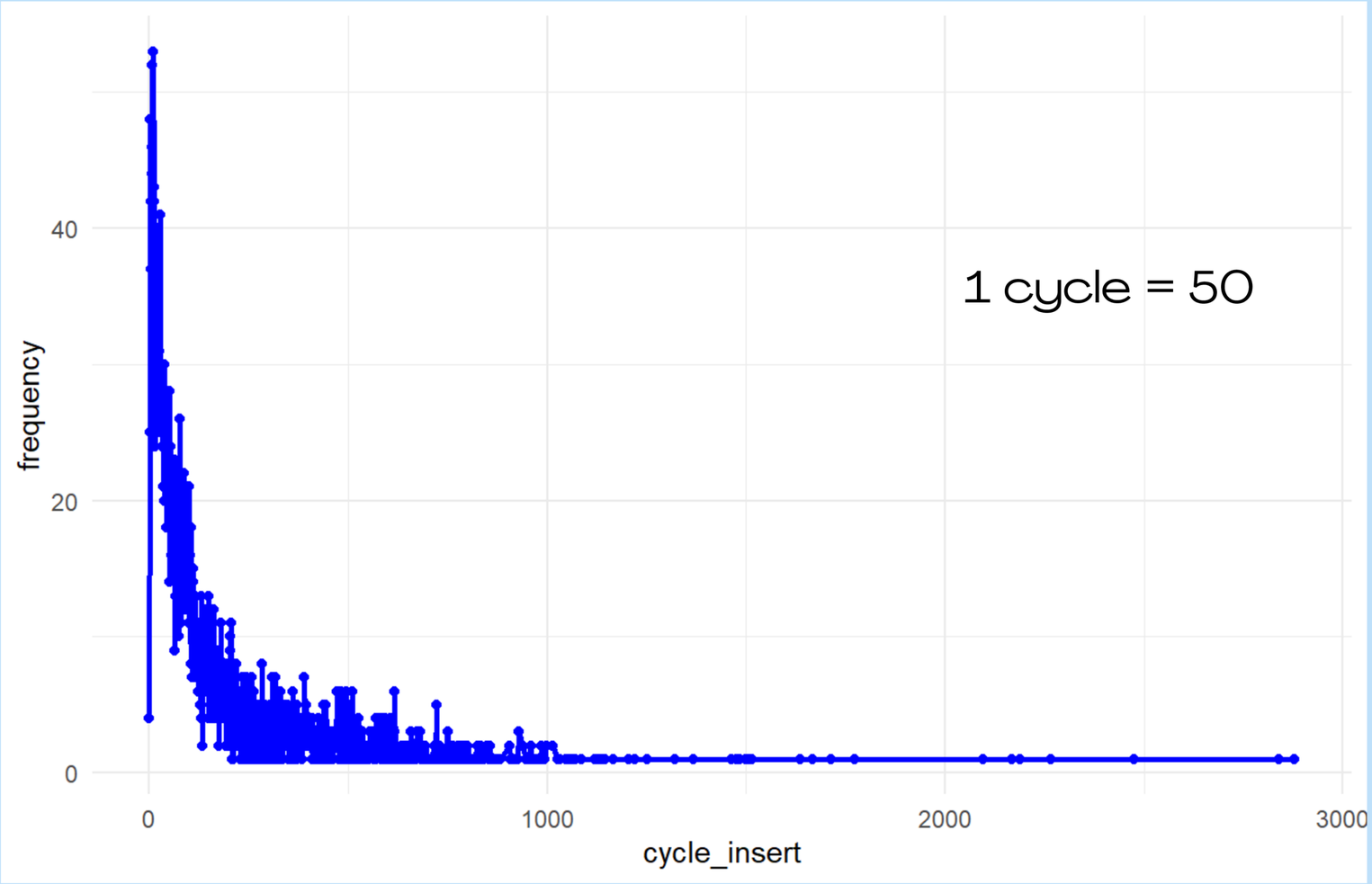
插入linkellist的head, middle, tail

刪除linkedlist的head, middle, tail

輸出結果

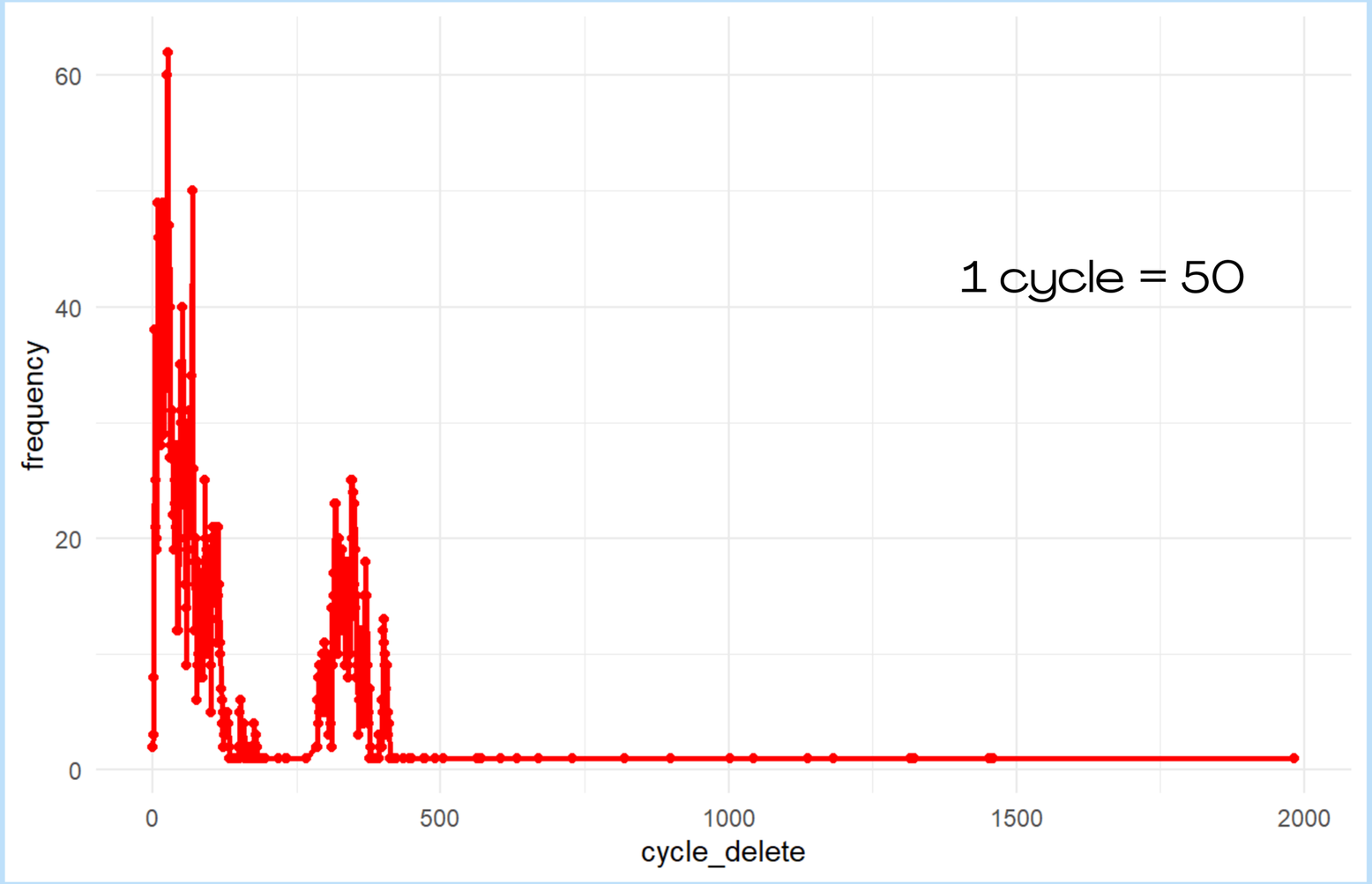
```
C:\C\NCKU\HW10>.\pB
-------initial routing table-------
Prefix: 18, MSB_8: 3 |000000110000000000001011000000000(3.0.11.1/18),
Prefix: 24, MSB_8: 4 |000001000000000100010010000000000(4.1.18.0/24), 000001000000001000111110000000000(4.2.62.0/24),

-------routing table after insertion-------
Prefix: 18, MSB_8: 3 |000000110000000000001011000000000(3.0.11.1/18), 000000110000101000000010000000000(3.10.2.3/18),
Prefix: 24, MSB_8: 4 |000001000000000100010110000000000(4.1.11.0/24), 000001000000000100010010000000000(4.1.18.0/24), 0000
010000000100010110000000000(4.2.44.0/24), 000001000000001000111110000000000(4.2.62.0/24),

-------routing table after insertion and deletion-------
Prefix: 18, MSB_8: 3 |000000110000000000001011000000000(3.0.11.1/18),
Prefix: 24, MSB_8: 4 |000001000000000100010010000000000(4.1.18.0/24), 000001000000001000111110000000000(4.2.62.0/24),

-------address matches-------
3.10.2.3 failed to match
4.7.6.8 failed to match
3.0.11.1 matched successfully
4.2.44.0 failed to match
7.8.9.5 failed to match
4.2.62.0 matched successfully
```

Distiribution of clock cycle of insertion



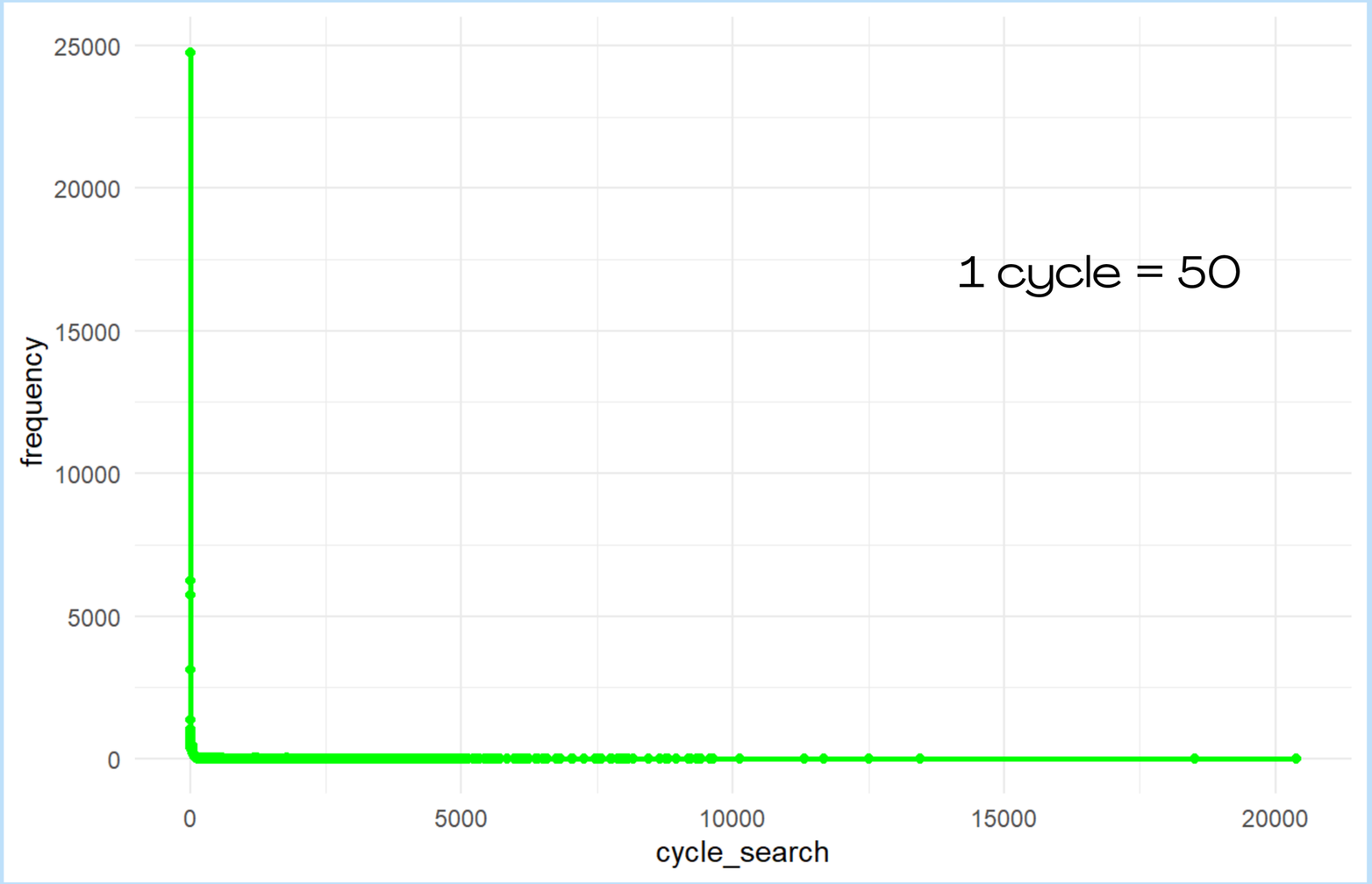
average clock cycles of a insertion = 178

Distiribution of clock cycle of deletion



average clock cycles of a deletion = 139.7

Distiribution of clock cycle of search



average clock cycles of a search = 278.3