HW9 REPORT

B911026 김민욱

1. 인접한 단어(node)들을 구하기 위해 인접 리스트을 사용했다. 인접 행렬이 아닌 인접 리스트를 사용한 이유는 인접 행렬은 인접한 노드들을 구하기 위해 N(=5757)번 만큼의 연산을 수행해야 하지만 인접 리스트는 인접한 노드의 수만큼만 연산을 수행하기 때문에 시간 측면에서 이점이 있다고 생각했기 때문이다.

(a)

```c
/********** 1-(a) **********/
int word_index = search_index("hello");
struct node *word_list = adj_list[word_index];
struct node *r = word_list;
int degree = 0;

while (r != NULL)
{
    print_word(r->index);
    printf(" is adjacent to ");
    print_word(word_index);
    printf(".\n");
    degree++;
    r = r->next;
}

printf("\n****degree of ");
print_word(word_index);
printf(" is %d.****\n\n", degree);
/********** end of 1-(a) **********/
```

```
cello is adjacent to hello.
hallo is adjacent to hello.
hells is adjacent to hello.
hullo is adjacent to hello.
jello is adjacent to hello.

****degree of hello is 5.****
```

(b)

```c
/********** 1-(b) **********/
word_index = search_index("graph");
word_list = adj_list[word_index];
r = word_list;
degree = 0;

while (r != NULL)
{
    print_word(r->index);
    printf(" is adjacent to ");
    print_word(word_index);
    printf(".\n");
    degree++;
    r = r->next;
}

printf("\n****degree of ");
print_word(word_index);
printf(" is %d.****\n\n", degree);
/********** end of 1-(b) **********/
```

```
grape is adjacent to graph.
grapy is adjacent to graph.

****degree of graph is 2.****
```

2. degree에 해당하는 노드의 개수를 저장하기 위한 배열 degOfVerties를 생성했다. 각 word는 5글자이므로 130(한 글자가 다른 경우의 수(알파벳의 수: 26) * 5)을 배열의 크기로 초기화했다. 또한 이 배열을 채우기 위해 인접 리스트를 순회하는 방식을 선택하였으며 순회하는 동안 maximum degree를 구할 수 있도록 코드를 작성하였다.

3. 아래의 코드 결과, maximum degree는 25이다.

```c
/********** 2 ,3 **********/
int degOfVerties[130] = {0,}; // 130 = number of alphabet * 5
int i,j;
int maxDegree = 0;

for (i = 0; i < N; i++)
{
    r = adj_list[i];
    degree = 0;
    while (r != NULL)
    {
        degree++;
        r = r->next;
    }
    degOfVerties[degree]++;
    if(degree>=maxDegree){
        maxDegree = degree;
    }

}
printf("maximum degree is %d\n\n", maxDegree);

for (i = 0; i <= maxDegree; i++)
{
    printf("%d: %d\n", i, degOfVerties[i]);
}
/********** end of 2 ,3**********/
```

```
maximum degree is 25

0: 671
1: 774
2: 727
3: 638
4: 523
5: 428
6: 329
7: 280
8: 249
9: 213
10: 188
11: 162
12: 120
13: 116
14: 102
15: 75
16: 53
17: 32
18: 32
19: 20
20: 8
21: 6
22: 4
23: 2
24: 3
25: 2
```

4. maximum degree를 가지는 word를 출력하기 위해 다시 한번 인접 리스트를 순회하는 과정을 거쳤다. 한 개의 list를 순회할 때마다 degree가 3번에서 구한 maximum degree와 같다면 해당 word를 출력하도록 코드를 작성하였다.
실행 결과, maximum degree(25)를 가지는 word 2개는 babes, bores 이다.

```
/********** 4 **********/
for (i = 0; i < N; i++)
{
    degree = 0;
    r = adj_list[i];
    while (r != NULL)
    {
        r = r->next;
        degree++;
    }
    if(degree == maxDegree){
        printf("word that has maximum degree is ");
        print_word(adj_list[i]->index);
        printf(".\n");
    }
}
/********** end of 4 **********/
```

```
word that has maximum degree is babes.
word that has maximum degree is bores.
```

5. average degree는 모든 node들의 degree의 합 / node의 개수이다. 따라서 모든 node들의 degree의 합을 구하기 위해 2번 문제에서 사용한 degOfVerties 배열을 사용하여 코드를 작성하였다. 실행 결과 average degree는 약 4.9이다.

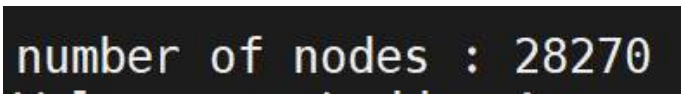
```
/********** 5 **********/
double averageDegree = 0.0;

for (i = 0; i <= maxDegree; i++)
{
    averageDegree += i * degOfVerties[i];
}
averageDegree /= N;
printf("average degree is %f.\n", averageDegree);
/********** end of 5 **********/
```

```
average degree is 4.910544.
```

6. 인접 리스트의 노드의 개수를 구하기 위해 2번에서 사용한 인접리스트를 순회하는 코드를 일부 재사용했다. 실행 결과 인접리스트의 노드의 개수는 28270개이다.

```
/********** 6 **********/
int numberOfNodes = 0;
for (i = 0; i <= N; i++)
{
    r = adj_list[i];
    while (r != NULL)
    {
        numberOfNodes++;
        r = r->next;
    }
}
printf("number of nodes : %d\n", numberOfNodes);
/********** end of 6 *********/
```



7. POOL_SIZE의 가능한 최솟값은 노드의 개수인 28270이다. 이때, pool의 크기를 계산해보자. 구조체 node는 int형 index 멤버변수 1개, struct point형 next 멤버변수 1개를 가지므로 총 4+4=8byte의 크기를 가진다. 이를 POOL_SIZE와 곱한다면 28270*8=226160byte이다. 즉 words.dat을 인접 리스트에 저장했을 경우 약 226kbyte만큼의 메모리를 사용한다. 인접 행렬을 사용하는 경우 char형 변수 N*N개가 필요하므로 1*5757*5757 = 33,143,049byte만큼의 메모리가 필요하다. 즉 인접 행렬이 메모리를 100배 정도 더 사용한다. 하지만 두 노드가 인접하는지를 알아내는데 필요한 연산은 인접 행렬이 압도적으로 빠르므로 두 방식의 장단점이 존재한다.

```
void hw9()
{
    /********** 1-(a) **********/
    int word_index = search_index("hello");
    struct node *word_list = adj_list[word_index];
    struct node *r = word_list;
    int degree = 0;

    while (r != NULL)
    {
        print_word(r->index);
        printf(" is adjacent to ");
        print_word(word_index);
        printf(".\n");
        degree++;
        r = r->next;
    }

    printf("\n****degree of ");
    print_word(word_index);
    printf(" is %d.****\n\n", degree);
    /********** end of 1-(a) **********/

    /********** 1-(b) **********/
    word_index = search_index("graph");
    word_list = adj_list[word_index];
    r = word_list;
    degree = 0;

    while (r != NULL)
    {
        print_word(r->index);
        printf(" is adjacent to ");
        print_word(word_index);
        printf(".\n");
        degree++;
        r = r->next;
    }
```

```
    printf("\n****degree of ");
    print_word(word_index);
    printf(" is %d.****\n\n", degree);
    /********** end of 1-(b) **********/

    /********** 2 ,3 **********/
    int degOfVerties[130] = {0,}; // 130 = number of alphabet * 5 , but 25 is maximum value of degree.
    int i,j;
    int maxDegree = 0;

    for (i = 0; i < N; i++)
    {
        r = adj_list[i];
        degree = 0;
        while (r != NULL)
        {
            degree++;
            r = r->next;
        }
        degOfVerties[degree]++;
        if(degree>=maxDegree){
            maxDegree = degree;
        }

    }
    printf("maximum degree is %d\n\n", maxDegree);

    for (i = 0; i <= maxDegree; i++)
    {
        printf("%d: %d\n", i, degOfVerties[i]);
    }
    /********** end of 2 ,3**********/

    printf("\n");

    /********** 4 **********/
    for (i = 0; i < N; i++)
    {
        degree = 0;
```

```
    r = adj_list[i];
    while (r != NULL)
    {
        r = r->next;
        degree++;
    }
    if(degree == maxDegree){
        printf("word that has maximum degree is ");
        print_word(adj_list[i]->index);
        printf(".\n");
    }
}
/********** end of 4 **********/

printf("\n");

/********** 5 **********/
double averageDegree = 0.0;

for (i = 0; i <= maxDegree; i++)
{
    averageDegree += i * degOfVerties[i];
}
averageDegree /= N;
printf("average degree is %f.\n", averageDegree);
/********** end of 5 **********/

printf("\n");

/********** 6 **********/
int numberOfNodes = 0;
for (i = 0; i <= N; i++)
{
    r = adj_list[i];
    while (r != NULL)
    {
        numberOfNodes++;
        r = r->next;
    }
}
printf("number of nodes : %d\n", numberOfNodes);
```

```
    /********** end of 6 **********/
}
```