Homework Assignment 8

Due: June 3, 9:00 pm

이름 : 이상혁

학번 : B898061

For this homework, we will try to figure out some properties of the "ladders" graph based on words.dat. Based on the data file and programs in hw8.zip, write programs to answer the following questions and submit a report that includes your code, answers, and any of your comments.

여기에 제시된 과제는 `words.dat`를 기반으로 한 "사다리" 그래프의 여러 특성을 분석하는 것을 목표로 합니다. `hw8.zip`에 포함된 데이터 파일과 프로그램을 사용하여 다음 질문에 답하는 프로그램을 작성하고, 코드, 답변 및 추가 코멘트를 포함한 보고서를 제출하라는 내용입니다.

1. On the "ladders" graph based on words.dat,

(a) print out all the words adjacent to hello. What is the degree of hello?

(b) print out all the words adjacent to graph. What is the degree of graph?

1. `words.dat`을 기반으로 한 "사다리" 그래프에서:

(a) 'hello'와 인접한 모든 단어를 출력하고, 'hello'의 차수(degree)는 얼마인가?

(b) 'graph'와 인접한 모든 단어를 출력하고, 'graph'의 차수는 얼마인가?

```
Number (1)
Case of 'hello'
Adjacency words of 'hello':
cello
hallo
hells
hullo
jello
Degree of 'hello': 5

Case of 'graph'
Adjacency words of 'graph':
grape
grapy
Degree of 'graph': 2
```

2. Compute the table of distribution of degrees. That is, make a table of the number of degree-0 vertices, the number of degree-1 vertices, . . . , etc. For example, the table should look like

2. 차수의 분포 테이블을 계산합니다. 즉, 차수 0인 정점의 수, 차수 1인 정점의 수 등을 나타내는 테이블을 만듭니다. 예를 들어 테이블은 다음과 같아야 합니다:

```
Number (2)
Degree distribution table:
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
```

3. What is the maximum degree?

3. 최대 차수는 얼마인가?

```
Number (3)
Maximum degree: 25
```

4. What are the words that have the maximum degree?

4. 최대 차수를 가진 단어는 무엇인가?

```
Number (4)
Words having maximum degree:
bares
cores
```

5. What is the average degree?

5. 평균 차수는 얼마인가?

```
Number (5)
Total degree:  28270
Average degree: 4.91
```

6. How many nodes does our adjacency list have?

6. 인접 리스트에는 몇 개의 노드가 있는가?

```
Number (6)
Number of total nodes in adjacency list: 28270
```

7. What is the minimum possible size required of POOL SIZE in backend.c?

7. `backend.c`에서 POOL_SIZE의 최소 가능 크기는 얼마인가?

```
Number (7)
Minimum possible size of POOL_SIZE: 28270
```

8. Include the source code for hw8() in your report, and submit as a pdf file.

8. 보고서에 `hw8()`의 소스 코드를 포함하고, PDF 파일로 제출하라.

```
/********** Homework 8 *****************/
void hw8()
{
    printf("₩n");

    //*****************Number(1)*******************

    printf("Number (1)₩n");
    char words[2][6] = { "hello", "graph" }; // 검사할 단어들
    for (int i = 0; i < 2; i++)
    {
        char* current_word = words[i];
        int index = search_index(current_word); // 단어의 인덱스 찾기

        struct node* p = adj_list[index]; // 인접 리스트에서 해당 인덱스의 첫 번째 노드
        int degree = 0;

        printf("Case of '%s'₩n", current_word);
        printf("Adjacency words of '%s':₩n", current_word);
        while (p != NULL) {
            print_word(p->index); // 인접한 단어 출력
            printf("₩n");
            degree++; // 차수 증가
            p = p->next; // 다음 노드로 이동
        }
        printf("Degree of '%s': %d₩n₩n", current_word, degree);
    }

    //*****************Number(2)*******************

    printf("Number (2)₩n");
    int degree_count[N] = { 0 };  // 모든 가능한 차수를 저장할 배열 초기화
    int max_degree = 0;  // 최대 차수를 추적

    // 각 단어의 차수 계산
    for (int i = 0; i < N; i++)
    {
        struct node* current = adj_list[i];
        int degree = 0;
        while (current != NULL)
        {
            degree++;
            current = current->next;
        }
        if (degree > max_degree)
        {
            max_degree = degree;  // 최대 차수 업데이트
        }
        degree_count[degree]++;  // 해당 차수의 카운트 증가
    }
    // 차수 분포 테이블 출력
```

```
printf("Degree distribution table:₩n");
for (int i = 0; i <= max_degree; i++)
{
    if (degree_count[i] > 0)
    {
        printf("%d: %d₩n", i, degree_count[i]);
    }
}
printf("₩n");

//*****************Number(3)*******************

printf("Number (3)₩n");
max_degree = 0;
struct node* p;
int degree;

for (int i = 0; i < N; i++) // 최대 차수 계산
{
    degree = 0;
    for (p = adj_list[i]; p != NULL; p = p->next)
    {
        degree++;
    }
    if (degree > max_degree)
    {
        max_degree = degree;
    }
}
printf("Maximum degree: %d", max_degree); // 최대 차수 출력
printf("₩n");

//*****************Number(4)*******************

printf("Number (4)₩n");
printf("Words having maximum degree:₩n"); // 최대 차수를 가진 단어는

for (int i = 0; i < N; i++)
{
    degree = 0;
    for (p = adj_list[i]; p != NULL; p = p->next) // 최대 차수를 가진 단어 찾기
    {
        degree++;
    }

    if (degree == max_degree) // 최대 차수와 같은 차수를 가진 단어 출력
    {
        print_word(i);
        printf("₩n");
    }
}
printf("₩n");
//*****************Number(5)*******************

printf("Number (5)₩n");
```

```
    int total_degree = 0;
    double average_degree;

    for (int i = 0; i < N; i++)
    {
        degree = 0;
        for (p = adj_list[i]; p != NULL; p = p->next)
        {
            degree++;
        }
        total_degree += degree;  // 차수의 총합에 현재 단어의 차수 더하기
    }

    if (N > 0)
    {
       average_degree = (double)total_degree / N;
    }
    else
    {
        average_degree = 0.0;
    }
    printf("Total degree: % d₩n", total_degree); // 총 차수 출력
    printf("Average degree: %.2f₩n", average_degree); // 평균 차수 출력
    printf("₩n");

    //******************Number(6)*********************

    printf("Number (6)₩n");
    int total_nodes = 0;
    for (int i = 0; i < N; i++) // 노드의 갯수 = 차수의 갯수 아닌가?
    {
        int node_count = 0;
        struct node* current = adj_list[i];
        while (current != NULL)
        {
            node_count++;
            current = current->next;
        }
        total_nodes += node_count;
    }
    printf("Number of total nodes in adjacency list: %d₩n", total_nodes); // 인접 리스트의
노드의 총 갯수 = 총 차수랑 같지 않나?
    printf("₩n");

    //******************Number(7)*********************

    printf("Number (7)₩n");
    printf("Minimum possible size of POOL_SIZE: %d₩n", total_nodes); // 최속 가능 POOL_SIZE는
사용중인 노드의 갯수와 같지 않나?
    printf("₩n");
}
```

9. Implement search index() using a binary search, instead of sequential search, and include your implementation in the report.

9. `search_index()`를 순차 검색 대신 이진 검색을 사용하여 구현하고, 구현한 코드를 보고서에 포함하라.

```
int search_index(char key[5])
{
    int l = 0;
    int h = N - 1;
    int m;

    while (l <= h)
    {
        m = l + (h - l) / 2; // 중간 인덱스 계산

        int result = compare(key, word[m]);
        if (result == 0)
        {
            return m; // key가 중간 인덱스의 단어와 일치할 경우
        }
        else if (result < 0)
        {
            h = m - 1; // key가 중간 단어보다 앞에 있을 경우
        }
        else
        {
            l = m + 1; // key가 중간 단어보다 뒤에 있을 경우
        }
    }
    return -1; // 찾지 못한 경우
}
```