HW8

자료구조 5분반 C111113 이나영

#1.

(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
(a) : The words adjacent to 'hello'
cello  hallo  hells  hullo  jello
The degree of the word 'hello' is : 5
(b) : The words adjacent to 'graph'
grape  grapy
(b) : The degree of the word 'graph' is : 2
```

1) search_index 함수를 이용해 단어 'hello'와 'graph'에 해당하는 index를 찾았다.

2) 그 후 adj_list에서 위의 index를 가리키는 pointer를 이용해 연결리스트를 순회하며 hello와 graph에 인접한 단어들을 출력하고 개수를 출력했다.

#2. Compute the table of distribution of degrees.

```
#2
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
```

1) Degree에 대한 배열을 만들어서 각 degree를 저장하고, 그 degree를 Index로 갖는 배열 degreeList를 만들어 해당 degree를 갖는 단어의 수를 저장했다.
2) 이후 degree 배열 내에서 max 값을 찾아 maximum degree값을 계산했다.
3) 그 후 0부터 maximum degree까지 순회하며 해당 degree를 갖는 단어의 수를 출력했다.

#3. What is the maximum degree?

```
#3
The maximum degree is 25.
```

- 위의 2번 문제에서의 1), 2)번 과정을 통해 계산된 maximum degree값이다.

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

```
#4
The words that have the maximum degree.
bares  cores
```

- 문제 2번에서와 거의 같은 코드를 사용했다. 0부터 N까지 순회하며 degree를 구하고 max_degree와 같은 degree일 때 단어를 출력하도록 했다.

5. What is the average degree?

```
#5.
The average degree is 4.910544.
```

- Average degree는 0부터 max_degree까지의 (degree * (해당 degree에 있는 노드의 개수))의 합을 N으로 나눈것이다.
- 문제의 더 정밀한 출력을 위해 float으로 형을 지정했다.

6. How many nodes does our adjacency list have?

```
#6
Adjacent list have 28270 nodes.
```

- Adj_list에서 노드의 총 개수는 각 노드의 degree의 합과 같다. 왜냐하면 각 단어와 인접한 단어의 수 만큼 노드가 생기고, 이는 곧 degree를 의미하기 때문이다.

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

```
#7
The minimum possible size required of POOL SIZE is 28270.001953.
```

- minimum possible pool size = (평균 degree) * (단어의 수)로 구할 수 있다.

```
actedOk!  I am doing homework!!!
#1
(a) : The words adjacent to 'hello'
cello  hallo  hells  hullo  jello
The degree of the word 'hello' is : 5
(b) : The words adjacent to 'graph'
grape  grapy
(b) : The degree of the word 'graph' is : 2
#2
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
The maximum degree is 25.
#4
The words that have the maximum degree.
bares  cores
#5.
The average degree is 4.910544.
#6
Adjacent list have 28270 nodes.
#7
The minimum possible size required of POOL SIZE is 28270.001953.
Welcome to Ladders!
```

#8. Source code – hw8() in backend.c

```
/********** Homework 8 *****************/
void hw8()
{
  printf("Ha ha ha.  Let's do homework!!!\n" );
  // You may use this space for your homework :)
  print_word(38);

  printf("Ok!  I am doing homework!!!\n");
  char hello[5] = "hello";
  char graph[5] = "graph";
  struct node *p = adj_list[search_index(hello)];
  struct node *q = adj_list[search_index(graph)];
  int counter = 0;


  // 1 -(a)
  printf("#1\n");
  printf("(a) : The words adjacent to 'hello'\n");
  while(p!=NULL){
    counter++;
    print_word(p->index);
    printf("  ");
    p= p->next;
  }
  printf("\n");
  printf("The degree of the word 'hello' is : %d\n", counter);


  // 1 -(b)
  counter = 0; // init counter
  printf("(b) : The words adjacent to 'graph'\n");

  while(q!=NULL){
    counter++;
    print_word(q->index);
    printf("  ");
    q=q->next;
  }
  printf("\n");
  printf("(b) : The degree of the word 'graph' is : %d\n", counter);

  // 2 & 3
  printf("#2\n");
  int degree[N] = {0,};
  int degreeList[N] = {0, };
  struct node *r = NULL;

  for(int i=0; i<N; i++){
```

```
  counter = 0;
  r = adj_list[i];
  while (r!=NULL){
    counter++;
    r = r->next;
  }
  degree[i] = counter;
  degreeList[counter]++;
}

int max_degree = degree[0];

for(int i=1; i<N; i++){ //degree 배열에서 max_degree 찾기
  if(max_degree<degree[i]){
    max_degree = degree[i];
  }
}

for(int i=0; i<max_degree+1; i++){ //max_degree 까지의 각 degree 별 단어 개수 출력
  printf("%d : %d\n", i, degreeList[i]);
}
printf("#3\n");
printf("The maximum degree is %d.\n",max_degree);

//4
printf("#4\n");
printf("The words that have the maximum degree.\n");

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

  if(max_degree == counter){
    print_word(i);
    printf("  ");
  }
}

//5
printf("\n#5.\n");

float avg_degree=0; //평균 degree 저장할 변수
int tmp=0; // 평균 degree 계산위해 사용되는 임시변수
int nodes_cnt = 0; //노드의 개수
```

```
  for(int i=0; i<max_degree+1; i++){
    tmp = i * degreeList[i];
    avg_degree += tmp;
  }
  avg_degree = avg_degree / N;
  printf("The average degree is %f.\n", avg_degree);

  //6
  printf("#6\n");
  for (int i=0; i<N; i++){
    nodes_cnt += degree[i];
  }
  printf("Adjacent list have %d nodes.\n", nodes_cnt);

  //7
  printf("#7\n");
  float min_pool = 0;
  min_pool = avg_degree * N;
  printf("The minimum possible size required of POOL SIZE is %f.\n", min_pool);




}
```

#9. Search_index를 sequential search가 아닌 binary search로 구현

```
int search_index(char key[5])
{
  int m = 0; // middle
  int l=0; // lowest value
  int r = N-1; //highest value

  // binary search
  while (l<=r){
    m = (l+r)/2;
    if (compare(key,word[m])>0){
      l= m+1;
    }
    else if (compare(key,word[m])<0){
      r = m-1;
    }
    else{
      return m; // success
    }
  }
  return -1; // unsuccess

}
```