자료구조 프로그래밍 과제2 제출

호크마교양대학 장예서

Huffman Code of Korean Syllable

1. 선택한 국어 요소와 빈도수

음절 통계 중 빈도수 상위 100개 선택:

번호	빈도	음절
1	314869	이
2	265071	다
3	232384	는
4	172126	의
5	171493	에
6	162579	을
7	152862	고
8	150918	가
9	144051	하
10	133691	지
11	113920	로
12	113150	한
13	108164	그
14	106129	은
15	103121	서
16	102709	기
17	99943	어
18	88314	도
19	86764	나
20	84774	를
21	84247	사
22	80043	아
23	78532	리
24	77858	있
25	75346	자
26	69423	들
27	67855	대
28	66390	으
29	64261	인
30	62338	시
31	60136	해
32	58485	라
33	58386	것
34	58165	수
35	57152	니
36	56945	게
37	54407	정
38	49019	보
39	45932	일
40	45475	적
41	44672	만
42	43880	부
43	43879	주
44	43685	과
45	43631	제
46	42568	면
47	40355	었
48	39553	전
49	39529	여
50	39345	상
51	37885	장
52	37391	요
53	36967	구
54	34109	문
55	33964	내
56	33145	우
57	31692	성
58	31282	거
59	31041	동
60	30364	생
61	30317	되
62	30184	마
63	30051	러
64	29241	국
65	29105	소
66	29068	화
67	28918	신
68	28660	무
69	28176	했
70	27582	원
71	26804	스
72	26341	오
73	26009	야
74	25964	위
75	25932	경
76	25618	말
77	25168	와
78	25055	조
79	24975	없
80	24764	모
81	24512	데
82	24266	비
83	23706	할
84	23653	계
85	23152	안
86	23016	까
87	22916	중
88	22764	세
89	22618	때
90	22373	미
91	22040	려
92	22033	회
93	21990	학
94	21976	관
95	21973	선
96	21939	간
97	21910	치
98	21719	진
99	21307	공
100	21280	실

2. 작성 프로그램 소스 코드

```c
// 자료구조 프로그래밍 과제 2 허프만 코드
// 호크마교양대학 장예서

#include <stdio.h>
#include <stdlib.h>
#define MAX_ELEMENT 200
// 헤더 파일 포함, MAX_ELEMENT 정의

typedef struct TreeNode {
        int weight;
        char* ch;
        struct TreeNode* left;
        struct TreeNode* right;
} TreeNode;
//TreeNode Struct 정의

typedef struct {
        TreeNode* ptree;
        char* ch;
        int key;
} element;
// element struct 정의

typedef struct {
        element heap[MAX_ELEMENT];
        int heap_size;
} HeapType;
// HeapType struct 정의

HeapType* create() {
        return (HeapType*)malloc(sizeof(HeapType));
}
// HeapType 생성 함수 정의

void init(HeapType* h) {
        h->heap_size = 0;
}
// HeapType 초기화 함수 정의

void insert_min_heap(HeapType* h, element item)
{
        int i;
        i = ++(h->heap_size);
        while ((i != 1) && (item.key < h->heap[i / 2].key)) {
                h->heap[i] = h->heap[i / 2];
                i /= 2;
        }
        h->heap[i] = item;
}
// 히프 h에 element item을 삽입하는 함수 정의

element delete_min_heap(HeapType* h)
{
```

```
        int parent, child;
        element item, temp;
        item = h->heap[1];
        temp = h->heap[(h->heap_size)--];
        parent = 1;
        child = 2;
        while (child <= h->heap_size) {
                if ((child > h->heap_size) &&
                        (h->heap[child].key) > h->heap[child + 1].key)
                        child++;
                if (temp.key < h->heap[child].key) break;
                h->heap[parent] = h->heap[child];
                parent = child;
                child *= 2;
        }
        h->heap[parent] = temp;
        return item;
}
// 삭제 함수 정의

TreeNode* make_tree(TreeNode* left, TreeNode* right) {
        TreeNode* node =
                (TreeNode*)malloc(sizeof(TreeNode));
        node->left = left;
        node->right = right;
        return node;
}
// 이진 트리 생성 함수 정의

void destroy_tree(TreeNode* root) {
        if (root == NULL) return;
        destroy_tree(root->left);
        destroy_tree(root->right);
        free(root);
}
// 이진 트리 제거 함수 정의

int is_leaf(TreeNode* root) {
        return !(root->left) && !(root->right);
}

void print_array(int codes[], int n) {
        for (int i = 0; i < n; i++)
                printf("%d", codes[i]);
        printf("₩n");
}

void print_codes(TreeNode* root, int codes[], int top)
{
        if (root->left) {
                codes[top] = 1;
                print_codes(root->left, codes, top + 1);
        }
        if (root->right) {
                codes[top] = 0;
                print_codes(root->right, codes, top + 1);
        }
        if (is_leaf(root)) {
                printf("%s: ", root->ch);
                print_array(codes, top);
        }
}
// 단말노드이면 코드를 출력하는 함수 정의

void huffman_tree(int freq[], char* ch_list[], int n)
{
        int i;
        TreeNode* node, * x;
        HeapType* heap;
        element e, e1, e2;
        int codes[100];
        int top = 0;
        heap = create();
        init(heap);
        for (i = 0; i < n; i++) {
                node = make_tree(NULL, NULL);
                e.ch = node->ch = &ch_list[i];
                e.key = node->weight = freq[i];
                e.ptree = node;
                insert_min_heap(heap, e);
        }
        for (i = 1; i < n; i++) {
                e1 = delete_min_heap(heap);
                e2 = delete_min_heap(heap);
                x = make_tree(e1.ptree, e2.ptree);
                e.key = x->weight = e1.key + e2.key;
                e.ptree = x;
                insert_min_heap(heap, e);
        }
        e = delete_min_heap(heap); // 최종 트리
        print_codes(e.ptree, codes, top);
        destroy_tree(e.ptree);
        free(heap);
}
// 허프만 코드 생성 함수 정의

int main(void)
{
        char* ch_list[] = {
        "이", "다", "는", "의", "에", "을", "고", "가", "하", "지",
        "로", "한", "그", "은", "서", "기", "어", "도", "나", "를",
        "사", "아", "리", "있", "자", "들", "대", "으", "인", "시",
        "해", "라", "것", "수", "니", "게", "정", "보", "일", "적",
        "만", "부", "주", "과", "제", "면", "었", "전", "여", "상",
        "장", "요", "구", "문", "내", "우", "성", "거", "동", "생",
        "되", "마", "러", "국", "소", "화", "신", "무", "했", "원",
        "스", "오", "야", "위", "경", "말", "와", "조", "없", "모",
        "데", "비", "할", "계", "안", "까", "중", "세", "때", "미",
        "려", "회", "학", "관", "선", "간", "치", "진", "공", "실" };
```

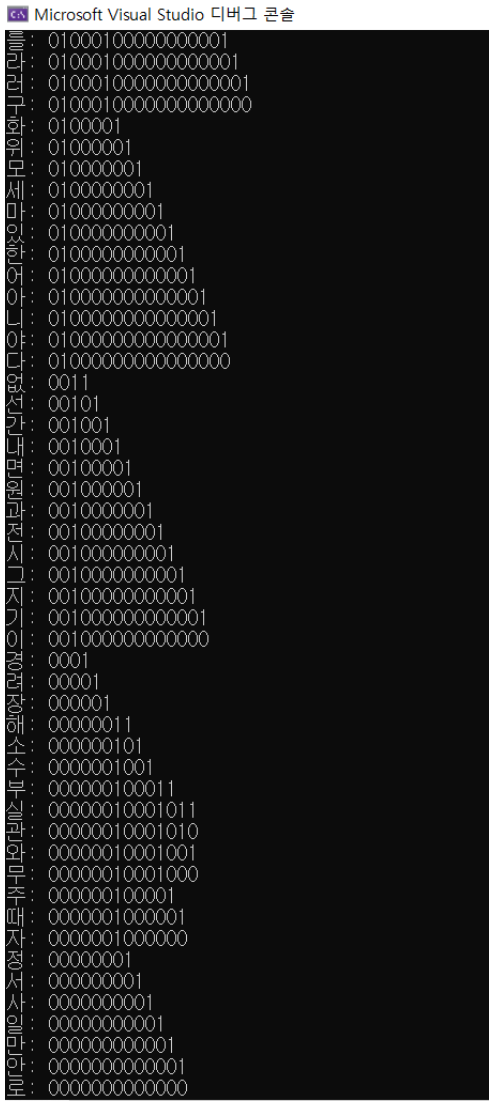
```
        int freq[] = {
        314869, 265071, 232384, 172126, 171493, 162579, 152862, 150918, 144051, 133691,
        113920, 113150, 108164, 106129, 103121, 102709, 99943, 88314, 86764, 84774,
        84247, 80043, 78532, 77858, 75346, 69423, 67855, 66390, 64261, 62338, 60136,
        58485, 58386, 58165, 57152, 56945, 54407, 49019, 45932, 45475, 44672,
        43880, 43879, 43685, 43631, 42568, 40355, 39553, 39529, 39345, 37885,
        37391, 36967, 34109, 33964, 33145, 31692, 31282, 31041, 30364, 30317,
        30184, 30051, 29241, 29105, 29068, 28918, 28660, 28176, 27582, 26804,
        26341, 26009, 25964, 25932, 25618, 25168, 25055, 24975, 24764, 24512,
        24266, 23706, 23653, 23152, 23016, 22916, 22764, 22618, 22373, 22040,
        22033, 21990, 21976, 21973, 21939, 21910, 21719, 21307, 21280 };

        huffman_tree(freq, *ch_list, 100);
        return 0;
}
```

3. 실행 화면 (다음장에 계속)

Microsoft Visual Studio 디버그 콘솔

```
공: 11
제: 1011
지: 10101
우: 101001
신: 1010001
활: 10100001
인: 101000001
분: 1010000001
학: 10100000001
언: 101000000001
조: 1010000000001
까: 10100000000001
에: 101000000000001
문: 1010000000000001
고: 10100000000000001
정: 101000000000000001
하: 1010000000000000001
나: 10100000000000000001
임: 10100000000000000000
을: 1001
회: 10001
요: 100001
쟁: 1000001
시: 10000001
당: 100000001
여: 1000000001
정: 10000000001
게: 100000000001
을: 1000000000001
들: 10000000000001
대: 100000000000001
희: 1000000000000001
했: 10000000000000001
도: 100000000000000001
보: 1000000000000000001
더: 10000000000000000001
가: 100000000000000000000
학: 011
진: 0101
문: 01001
적: 0100011
비: 01000101
미: 010001001
었: 0100010001
리: 01000100001
상: 010001000001
이: 0100010000001
거: 01000100000001
는: 010001000000001
제: 0100010000000001
```


Microsoft Visual Studio 디버그 콘솔
를라 : 0100010000000001
라라 : 01000100000000001
러라 : 010001000000000001
구화 : 01000100000000000
화 : 0100001
워 : 01000001
모 : 010000001
세 : 0100000001
마 : 01000000001
있안 : 010000000001
한어 : 0100000000001
아 : 01000000000001
니야 : 010000000000001
야다 : 0100000000000001
다없 : 01000000000000000
선 : 00101
간 : 001001
내 : 0010001
면원 : 00100001
과전 : 001000001
전시 : 0010000001
그지 : 00100000001
지기 : 001000000001
이 : 001000000000000
정관 : 0001
장해 : 00001
소수 : 000001
부 : 00000011
관학 : 000000101
무주 : 0000001001
태자 : 00000100011
정서 : 00000001
사일 : 0000001000011
만안 : 0000001000101
로 : 0000001000100
화 : 0000001000010
주태 : 0000001000001
자정 : 0000001000000
서사 : 000000001
일만 : 0000000001
안로 : 0000000000001
화 : 0000000000000