Huffman Coding

주어진 문자열에 대해 허프만 코드를 적용한 후의 전체 문자열의 bit수를 출력하세요.

입력/출력

aaaabbb 7 해설 : a와 b가 각각 1bit 코드로 변환되며, a가 4글자, b가 3글자이므로 전체 문자열은 7bit가 됩니다.
aaaaaabbbc 14 해설 : a는 1bit, b와 c는 각각 2bit 코드로 변환되며, 전체 문자열은 14bit가 됩니다.
abracadabra 23
visionzquestionzonionzcaptionzgraduationzeducation 185

조건

- 자료구조 라이브러리는 사용할 수 없으며, 단순 배열은 사용해도 무방합니다. 또한 직접 구현한 자료구조는 사용할 수 있습니다.
- 문자열 관련 라이브러리 함수와 클래스, 객체 함수는 사용해도 무방합니다.
- 문자열은 'a'부터 'z'까지 26개의 알파벳 소문자로만 이루어져 있으며, 최대길이는 100입니다.
- 각 test case별 시간제한은 1초입니다. 조건을 만족할 수 있는 프로그래밍 언어를 선택하세요.
- 주어진 문자열의 빈도수를 계산할 때, 최소 빈도수를 사용할 때, stable하게 동작하도록 하세요.
- 공백 문자가 없으므로 문자열 입력을 받을 때 다음과 같은 예제를 따르면 정상 작동합니다.

C char array[100]; scanf("%s", array);
C++ char array[100]; cin >> array;
Java Scanner s; String str = s.nextLine(); str.charAt(n); // (n번째 위치의 문자를 가져오는 함수)
Python inputs = input()

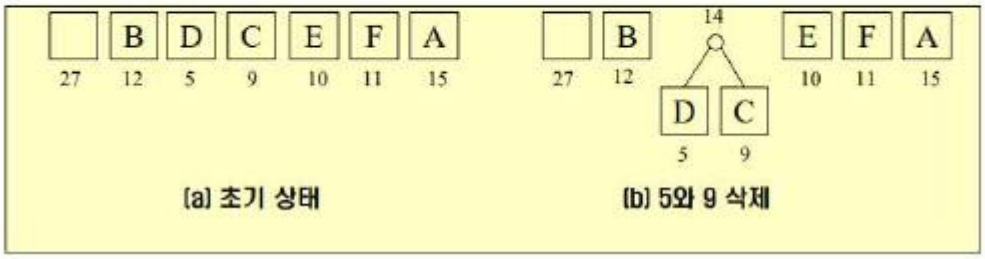
- 허프만 인코딩에 대해서는 다음 페이지에 설명되어 있습니다.

허프만 트리 구축 과정(1)

- 최소 히프
  - 허프만 트리의 자료구조에 가장 적합
- 주어진 텍스트의 빈도수를 계산

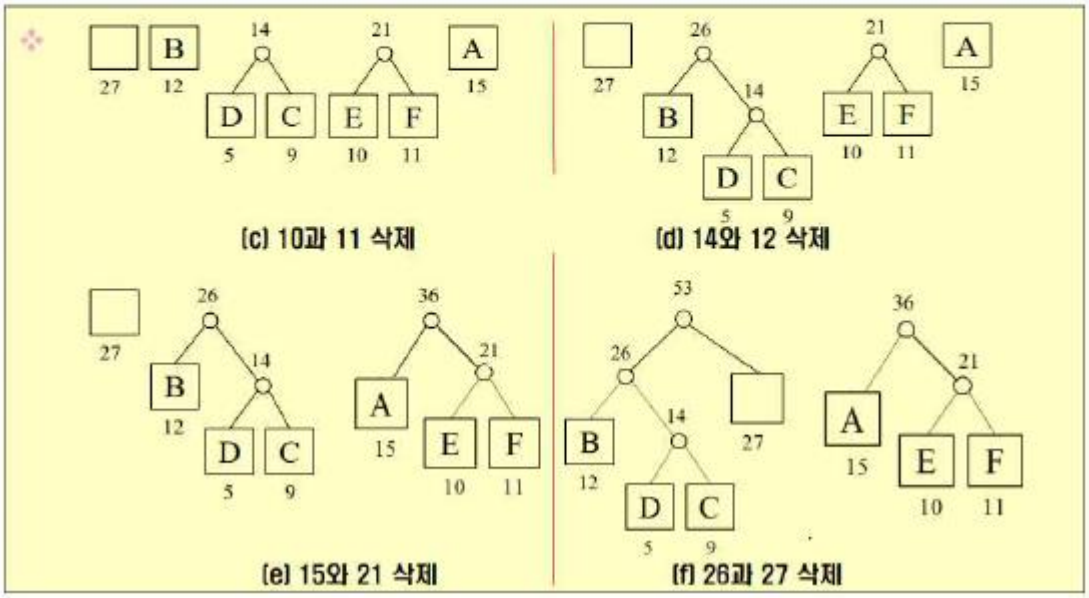
		A	B	C	D	E	F
k	0	1	2	3	4	5	6
count[k]	27	15	12	9	5	10	11

- 허프만 트리 구축



(a) 초기 상태 (b) 5와 9 삭제

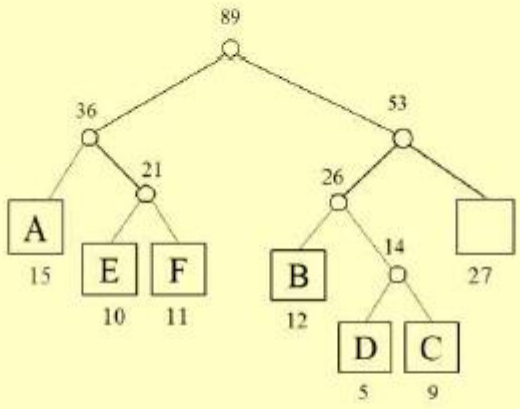
허프만 트리 구축 과정(2)



(c) 10과 11 삭제 (d) 14와 12 삭제

(e) 15와 21 삭제 (f) 26과 27 삭제

허프만 트리 구축 과정(3)



(g) 36과 53 삭제

		A	B	C	D	E	F
k	0	1	2	3	4	5	6
	11	00	100	1011	1010	010	011

모범답안

```
#include<iostream>
#include<stdio.h>
#include<string.h>
#include<string>
using namespace std;

struct ini {
    char cha;
    int freq;
    ini* left;
    ini* right;
    ini* par;
    string text;
    int len;
};

ini heap[5000];

void Heapifymin(ini arr[], int h, int m) {
    ini v = arr[h];
    int i;
    for (i = 2 * h; i <= m; i = i * 2) {
        if (i < m && arr[i].freq < arr[i + 1].freq) {
            i = i + 1;
        }
        if (v.freq >= arr[i].freq) {
            break;
        }
        else {
            arr[i / 2] = arr[i];
        }
    }
    arr[i / 2] = v;
}
void HeapSortmin(ini arr[], int n) {
    for (int i = n / 2; i >= 1; i--) {
        Heapifymin(arr, i, n);
    }
    for (int i = n - 1; i >= 1; i--) {
        ini temp = arr[1];
        arr[1] = arr[i + 1];
        arr[i + 1] = temp;
        Heapifymin(arr, 1, i);
    }
}

ini *table[26];
```

```
void buildEnconde(ini *r, string str) {
    r->text = str + r->text;
    r->len = r->text.size();
    if (r->left == NULL && r->right == NULL) {
        table[(int)r->cha-97] = r;
    }
    if (r->left != NULL) {
        buildEnconde(r->left, str + "0");
    }
    if (r->right != NULL) {
        buildEnconde(r->right, str + "1");
    }
}


ini build(ini arr[], int n) {
    ini r;
    for (int i = 1; i < n; i++) {
        ini p = arr[1];
        ini q = arr[2];
        r.freq = p.freq + q.freq;
        r.cha = p.cha + q.cha;
        arr[n + i * 2 - 1] = p;
        arr[n + i * 2] = q;
        r.left = &arr[n + i * 2 - 1];
        r.right = &arr[n + i * 2];
        arr[1] = r;
        for (int j = 2; j < n; j++) {
            arr[j] = arr[j + 1];
        }
        arr[n] = arr[0];
        HeapSortmin(arr, n - i);
    }
    return r;
}
```

```
char input[200];
int no1[26];
int main() {
    cin >> input;
    int inputlen = strlen(input);
    for (int i = 0; i < inputlen; i++) {
        no1[(int)input[i] - 97]++;
    }
    int size=0;
    for (int i = 0; i < 26;i++) {
        if (no1[i] == 0)    continue;
        else {
            size++;
            heap[size].freq = no1[i];
            heap[size].cha = i + 97;
            heap[size].right = NULL;
            heap[size].left = NULL;
            heap[size].par = NULL;
        }
    }
    HeapSortmin(heap, size);
    ini root = build(heap, size);

    root.len = 0;
    buildEnconde(&root, "");

    int finallen = 0;

    for (int i=0;i< inputlen;i++) {
        finallen = finallen + table[(int)input[i] - 97]->len;
    }
    cout << finallen << endl;

    return 0;
}
```