7. String

2017010698
수학과 오서영

문자열 (String)

: information retrieval(정보검색), bioinformatics(생물정보학)에서 유용

- **부분문자열**(substring) : S = "avada", S[0...2] = ava
- **접두사**(prefix) : S[...3] = avad
- **접미사**(suffix) : S[2...] = ava

Palindrome (회문)

: 문자열을 거꾸로 해도 원래의 문자열과 같은 문자열

주어진 문자열이 palindrome(회문)인지 확인하는 코드

```
def is_palindrome(s):
    return s == s[::-1]


s1 = "abccba"
s2 = “hello;"
print(is_palindrome(s1))   % True
print(is_palindrome(s2))   % False
```

O(n)

Manacher's Algorithm

: 문자열 내에서 팔린드롬(palindrome,회문)을 찾는것과 관련된 알고리즘

문자열 S 입력 -> 반환 :
문자열 S와 길이가 같은 정수 배열 A,
각 A의 원소 A[i]는 i번째 문자열을 중심으로 하는
가장 긴 팔린드롬의 반지름의 길이.

Ex) S='banana' -> A = '001210'

```
R = -1
p = -1
for i = 0 to n-1:
    if i <= R:
        A[i] = min(A[2*p - i], R-i)
    else:
        A[i] = 0
    while S[i-A[i]-1] == S[i+A[i]+1]:
        A[i] = A[i] + 1
    if i+A[i] > R:
        R = i+A[i], p = i
```

O(n)

1. i는 0부터 n−1(n=|S|)까지 진행된다
2. j<i인 모든 j에 대해 R=max(j+A[j])이라하고, 또한 그러한 j를 p라 하자. 즉, R=p+A[p]
3. i≤R인지 여부에 따라 A[i]의 **초기값**이 정해진다
i>R이라면, A[i]의 초기값은 0이다.
i≤R이라면, i는 p를 중심으로 한 팔린드롬에 속한다는 이야기이다. 이때 p를 중심으로 i의 대칭점 i′을 구한다. (즉, i′=2*p−i) A[i]의 초기값은 min(R−i,A[i′])으로 둔다.
4. A[i]의 초기값에서부터, S[i−A[i]]과 S[i+A[i]]가 같을 때까지 A[i]를 증가시키고, 그 다음 i로 넘어간다.

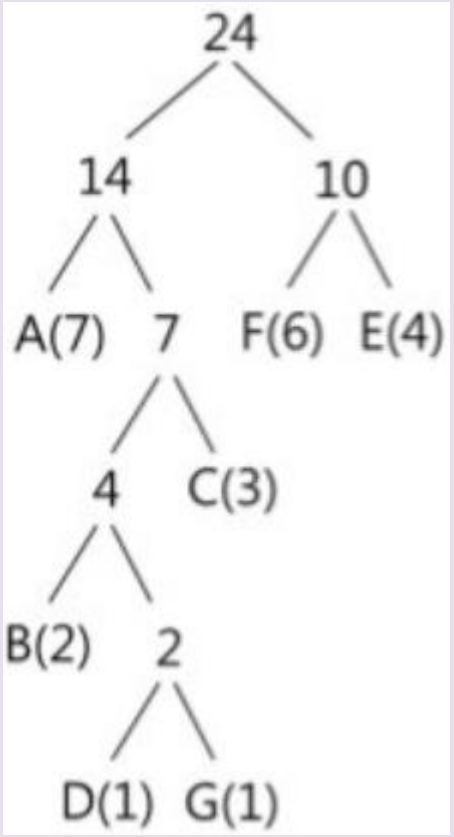
Huffman coding

: 텍스트 압축을 위해 널리 사용되는 방법, 원본 데이터에서 자주 출현하는 문자는 적은 비트의 코드로 변환, 빈도가 낮은 문자는 많은 비트의 코드로 변환하여 표현 -> 전체 데이터를 표현하는데 필요한 비트 수를 줄이는 방식

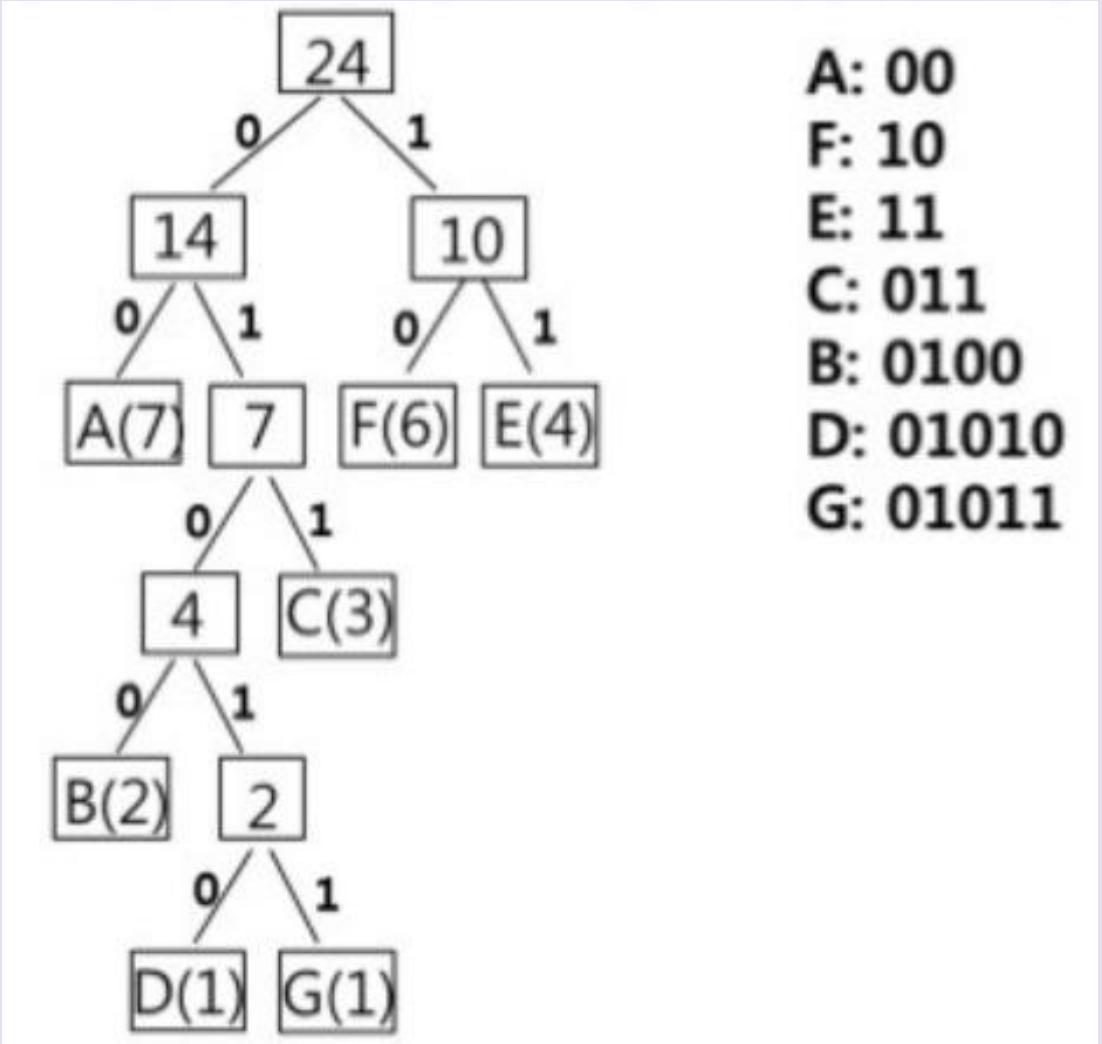
AAAAAAABBCCCDEEEEFFFFFFG

1. 출현 빈도수를 내림차순 정렬
: A(7) F(6) E(4) C(3) B(2) D(1) G(1)

2. 출현 빈도가 가장 작은 D,G를 묶어
이진트리를 구성하고 루트노드에 합인 2를 부여
-> 계속 반복

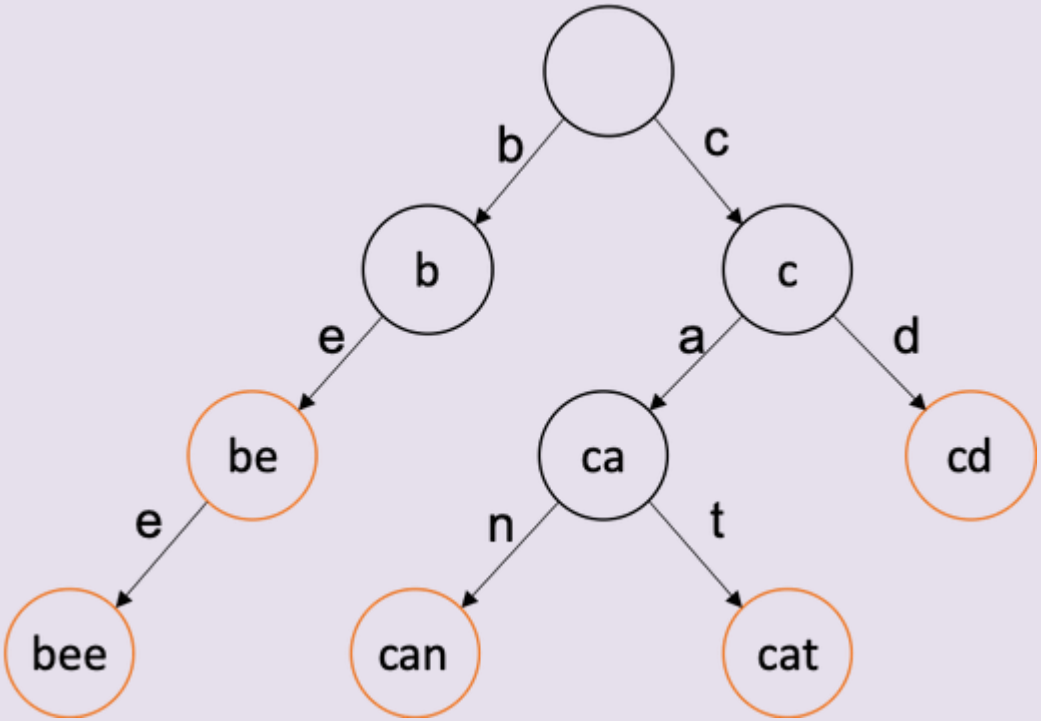


3. 전체 트리에 대해 각가지 왼쪽은 0,
오른쪽은 1을 기입하여 허프만 트리를 완성



Trie

: 문자열을 저장하고 효율적으로 탐색하기 위한 트리 형태의 자료구조



목적

빠르게 탐색이 가능하다는 장점
각 노드에서 자식들에 대한 포인터들을 배열로 모두 저장하고 있다는 점에서
저장 공간의 크기가 크다는 단점
-> 검색어 자동완성, 사전에서 찾기, 문자열 검사

시간 복잡도

제일 긴 문자열의 길이 : L, 총 문자열들의 수 : M

생성시 시간복잡도 : O(M*L

탐색시 시간복잡도 : O(L).

Suffix Tree (접미사 배열)

: 문자열 S의 모든 접미사를 사전순으로 정렬해 놓은 배열

Ex) baekjoon의 접미사는 baekjoon, aekjoon, ekjoon, kjoon, joon, oon, on, n 으로 총 8가지.
이를 사전순으로 정렬하면, aekjoon, baekjoon, ekjoon, joon, kjoon, n, on, oon이 된다.

문자열 S가 주어졌을 때, 모든 접미사를 사전순으로 정렬한 다음 출력하는 프로그램을 작성하시오.

```
#include <iostream>
#include <string>
#include <algorithm>
using namespace std;

int main(){
    string str[1000];
    cin >> str[0];
 // substr을 사용하여 하나씩 늘려가면서 접미사 저장
    int len = str[0].size();
    for(int i=1; i<len; i++)
        str[i] = str[0].substr(i);
 // sort() 함수의 첫번째 파라미터 = 시작점 포인터지점(주소),
두번째 파라미터 = 도착지 포인터지점(주소)+문자열개수(str.size)
    sort(str, str+len);
 // 저장된 접미사값 sort된값을 사용하여 정렬
    for(int i=0; i<len; i++)
        cout << str[i] << endl;
}
```

O(n)

Reference

[1] [ALGOSPOT] Manacher's algorithm,
https://algospot.com/wiki/read/Manacher%27s_algorithm
[2] 허프만 트리를 이용한 텍스트 압축,
https://m.blog.naver.com/ndb796/220829142548
[3] [알고리즘] 백준알고리즘 11656번 접미사 배열,
https://www.acmicpc.net/problem/11656
[4] 구종만.『프로그래밍 대회에서 배우는 알고리즘 문제해결전략』.
인사이트(2012)