JooHyun – Lee (comkiwer)

TS문자열검색

Hancom Education Co. Ltd.

Problem

Problem

TS웹페이지에 단어를 추가, 삭제하고, 검색하는 시스템을 설계하고 있다.

추가되는 단어는 모두 영어 알파벳 소문자로만 구성되어 있다고 가정한다.

삭제하거나 검색할 때 주어지는 단어는 와일드카드 문자 '?'가 최대 3개 포함될 수 있다.

와일드카드 문자 '?' 하나는 임의의 한개 문자로 치환될 수 있는 특별한 문자이다.

예를 들어, "ad??t"는 "admit", "adult", "adapt"가 될 수 있지만,
"about", "adios", "adjust"는 될 수 없다.

그리고 "?oat"는 ""boat", "coat", "goat"는 될 수 있지만,
"beat", "float", "throat"는 될 수 없다.

아래 API 설명을 참조하여 각 함수를 구현하시오.

Problem

1. `void init()` :

   각 테스트 케이스의 처음에 호출된다.

   시스템에 단어가 없는 상태가 된다.

2. `int addStr(char str[])` :

   시스템에 단어 `str`을 추가한 다음에, 시스템에 존재하는 단어 `str`의 개수를 반환한다.

   `str`은 길이가 5이상 30이하의 영어 알파벳 소문자로 구성되어 있고,
   '\0' 문자로 끝나는 문자열이다. 시스템에 이미 존재하는 단어도 추가해야 한다.

   Parameters

   `str`: 추가되는 단어 (5 ≤ 단어의 길이 ≤ 30)

   Returns

   시스템에 존재하는 단어 `str`의 개수를 반환한다.

Problem

3. `int deleteStr(char str[]) :`

시스템에서 `str`과 일치하는 단어를 모두 삭제하고, 삭제한 단어의 개수를 반환한다.

`str`은 길이가 `5`이상 `30`이하의 영어 알파벳 소문자와
와일드카드 문자 '`?`'로 구성되어 있고, '`\0`' 문자로 끝나는 문자열이다.

`str`에 포함될 수 있는 와일드카드 문자 '`?`'는 최대 `3`개이다.

삭제한 단어가 없을 경우, `0`을 반환한다.

Parameters

`str:` **삭제하는 단어** `( 5 ≤` **단어의 길이** `≤ 30, 0 ≤ '?'`**의 개수** `≤ 3 )`

Returns

시스템에서 삭제한 단어의 개수를 반환한다.

Problem

4. `int searchStr(char str[]) :`

시스템에서 `str`과 일치하는 단어를 검색하고, 검색된 단어의 개수를 반환한다.

`str`은 길이가 `5`이상 `30`이하의 영어 알파벳 소문자와
와일드카드 문자 '`?`'로 구성되어 있고, '`\0`' 문자로 끝나는 문자열이다.

`str`에 포함될 수 있는 와일드카드 문자 '`?`'는 최대 `3`개이다.

검색된 단어가 없을 경우, `0`을 반환한다.

Parameters

`str:` **검색하는 단어** `( 5 ≤` **단어의 길이** `≤ 30, 0 ≤ '?'`**의 개수** `≤ 3 )`

Returns

시스템에서 검색된 단어의 개수를 반환한다.

Problem

[제약사항]

1. 각 테스트 케이스 시작 시 init() 함수가 호출된다.
2. 삭제하거나 검색할 때 주어지는 문자열은 와일드카드 문자 '?'가 최대 3개 포함될 수 있다.
3. 각 테스트 케이스에서 addStr() 함수의 호출 횟수는 20,000 이하이다.
4. 각 테스트 케이스에서 모든 함수의 호출 횟수 총합은 40,000 이하이다.

Problem

TS문자열관리

[예제]

순서	함수	반환
1	init()	
2	addStr (“about”)	1
3	addStr (“about”)	2
4	addStr (“above”)	1
5	searchStr (“abo??”)	3
6	addStr (“admit”)	1
7	addStr (“adult”)	1
8	searchStr (“ad??t”)	2
9	addStr (“amount”)	1
10	searchStr (“a???t”)	4
11	deleteStr (“a?o?n?”)	1
12	addStr (“avoid”)	1
13	searchStr (“a?o??”)	4
14	deleteStr (“about”)	2
15	searchStr (“a?o??”)	2
16	addStr (“visit”)	1
17	searchStr (“???it”)	2
18	searchStr (“?c?d?”)	0

Problem analysis

Problem analysis : 예제분석

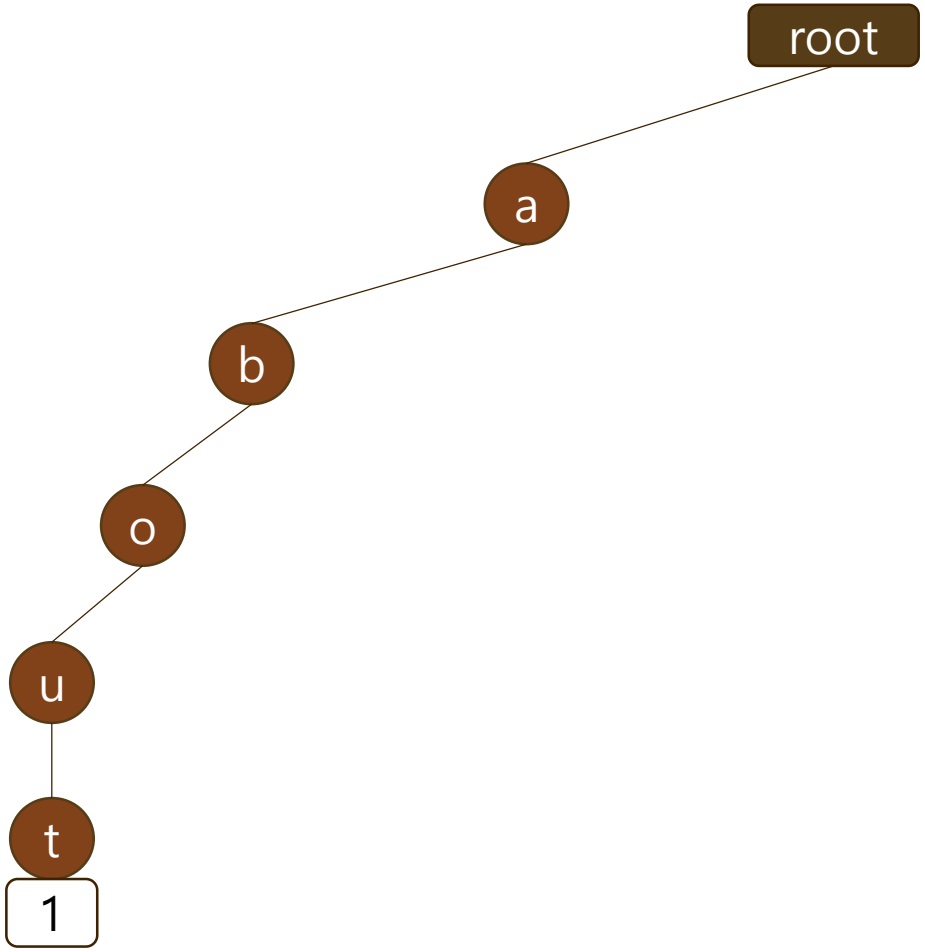
1

root

2 about 1
2 about 2
2 above 1
4 abo?? 3
2 admit 1
2 adult 1
4 ad??t 2
2 amount 1
4 a???t 4
3 a?o?n? 1
2 avoid 1
4 a?o?? 4
3 about 2
4 a?o?? 2
2 visit 1
4 ???it 2
4 ?c?d? 0

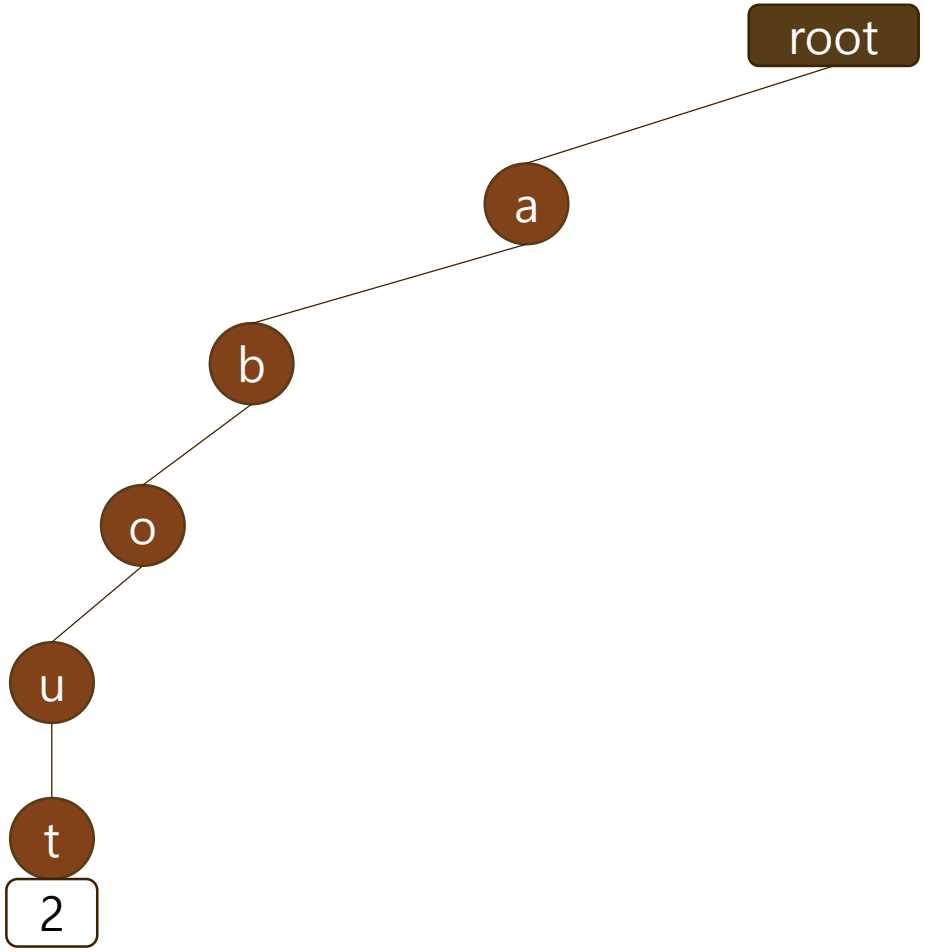
Problem analysis : 예제분석

1
2 about 1
2 about 2
2 above 1
4 abo?? 3
2 admit 1
2 adult 1
4 ad??t 2
2 amount 1
4 a???t 4
3 a?o?n? 1
2 avoid 1
4 a?o?? 4
3 about 2
4 a?o?? 2
2 visit 1
4 ???it 2
4 ?c?d? 0



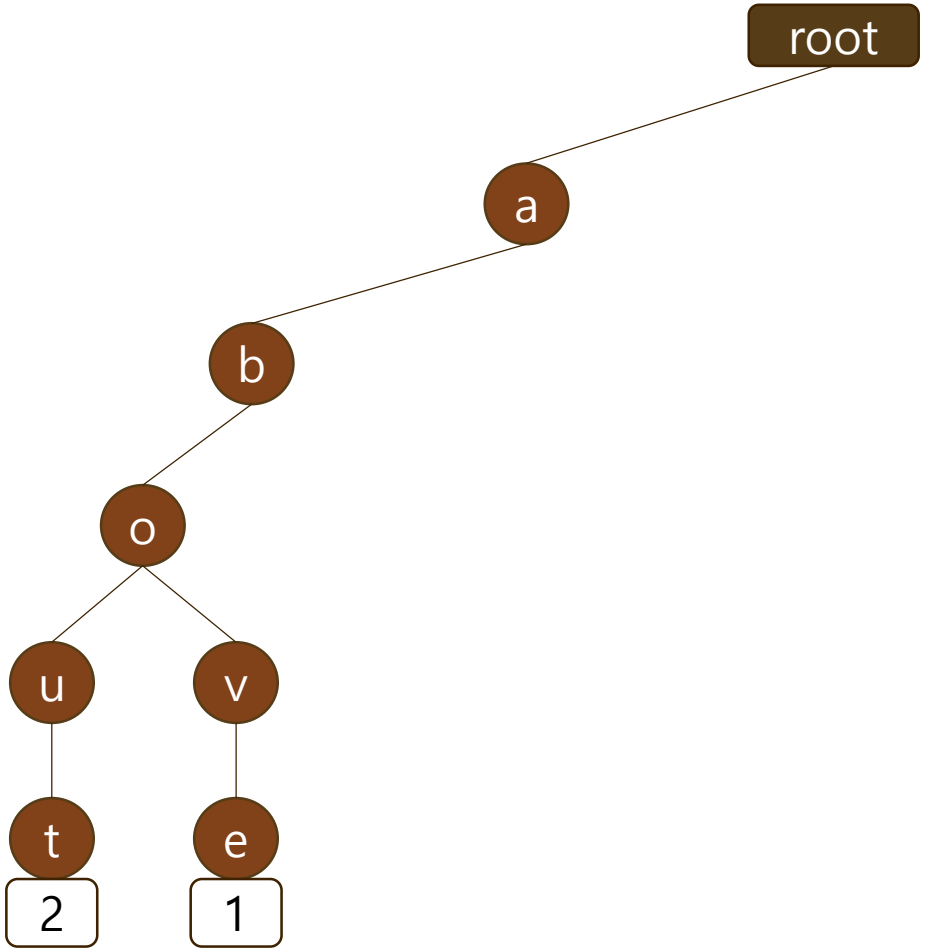
Problem analysis : 예제분석

1
2 about 1
2 about 2
2 above 1
4 abo?? 3
2 admit 1
2 adult 1
4 ad??t 2
2 amount 1
4 a???t 4
3 a?o?n? 1
2 avoid 1
4 a?o?? 4
3 about 2
4 a?o?? 2
2 visit 1
4 ???it 2
4 ?c?d? 0



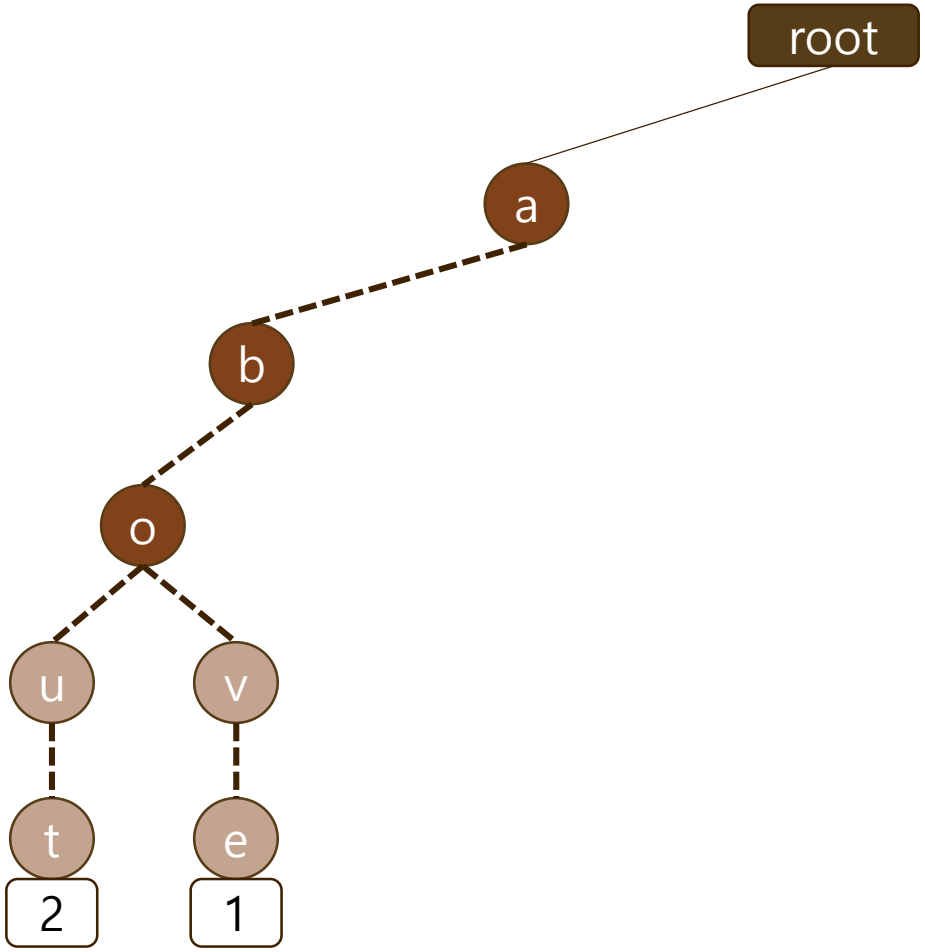
Problem analysis : 예제분석

1
2 about 1
2 about 2
2 above 1
4 abo?? 3
2 admit 1
2 adult 1
4 ad??t 2
2 amount 1
4 a???t 4
3 a?o?n? 1
2 avoid 1
4 a?o?? 4
3 about 2
4 a?o?? 2
2 visit 1
4 ???it 2
4 ?c?d? 0



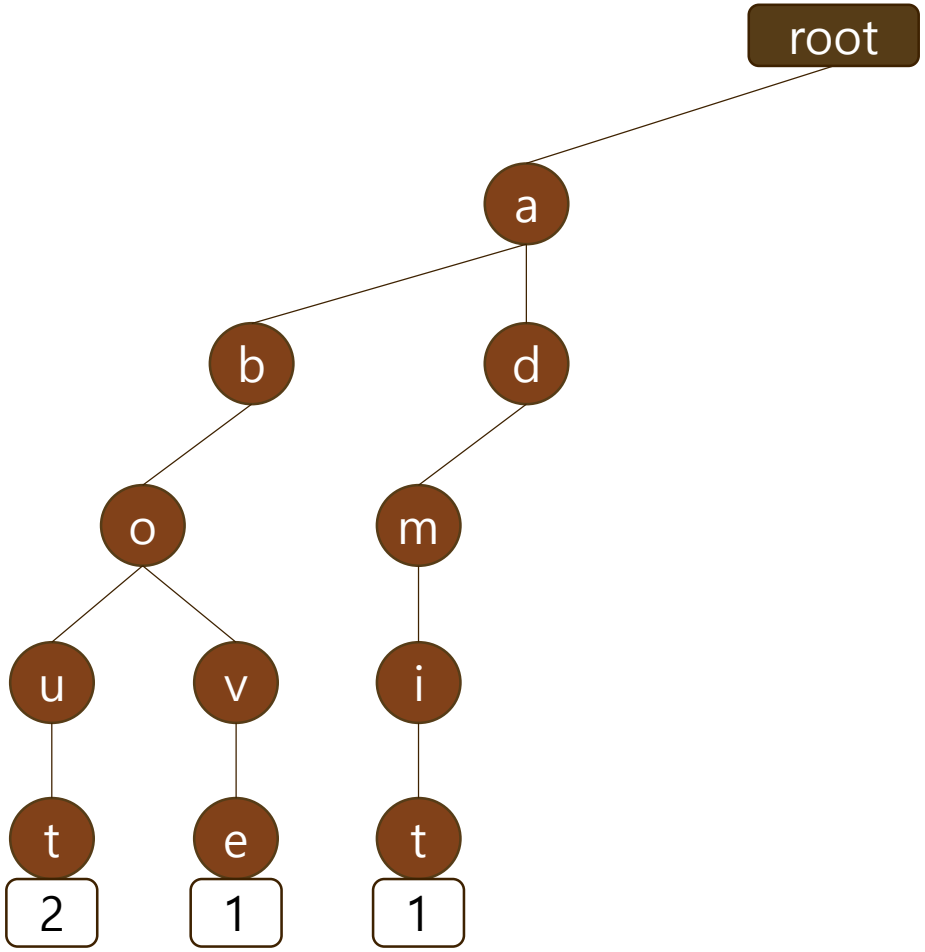
Problem analysis : 예제분석

1
2 about 1
2 about 2
2 above 1
4 abo?? 3
2 admit 1
2 adult 1
4 ad??t 2
2 amount 1
4 a???t 4
3 a?o?n? 1
2 avoid 1
4 a?o?? 4
3 about 2
4 a?o?? 2
2 visit 1
4 ???it 2
4 ?c?d? 0



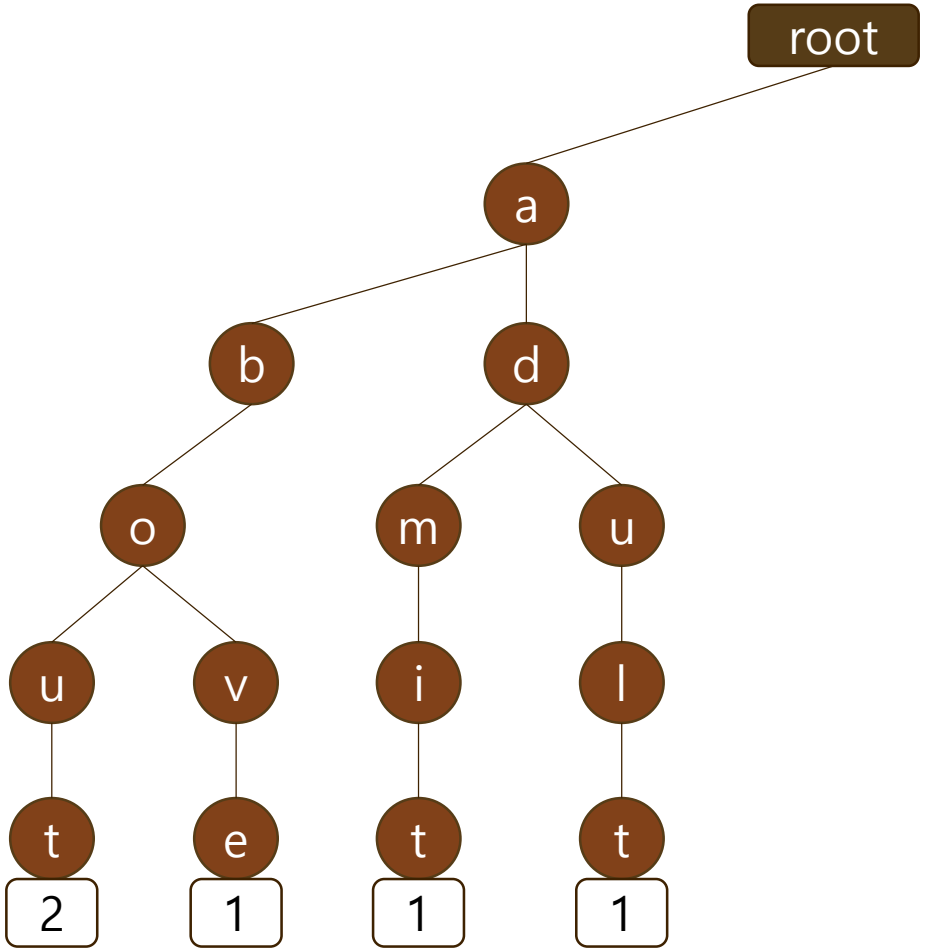
Problem analysis : 예제분석

1
2 about 1
2 about 2
2 above 1
4 abo?? 3
2 admit 1
2 adult 1
4 ad??t 2
2 amount 1
4 a???t 4
3 a?o?n? 1
2 avoid 1
4 a?o?? 4
3 about 2
4 a?o?? 2
2 visit 1
4 ???it 2
4 ?c?d? 0



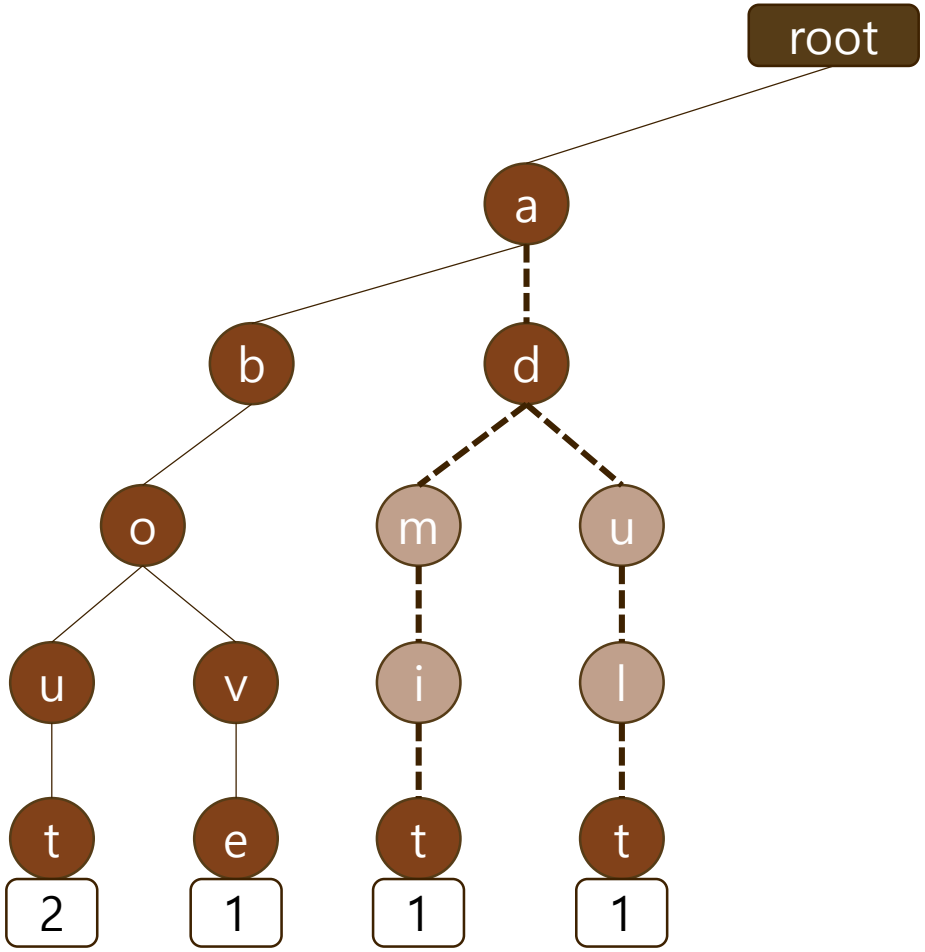
Problem analysis : 예제분석


```
1
2 about 1
2 about 2
2 above 1
4 abo?? 3
2 admit 1
2 adult 1
4 ad??t 2
2 amount 1
4 a???t 4
3 a?o?n? 1
2 avoid 1
4 a?o?? 4
3 about 2
4 a?o?? 2
2 visit 1
4 ???it 2
4 ?c?d? 0
```



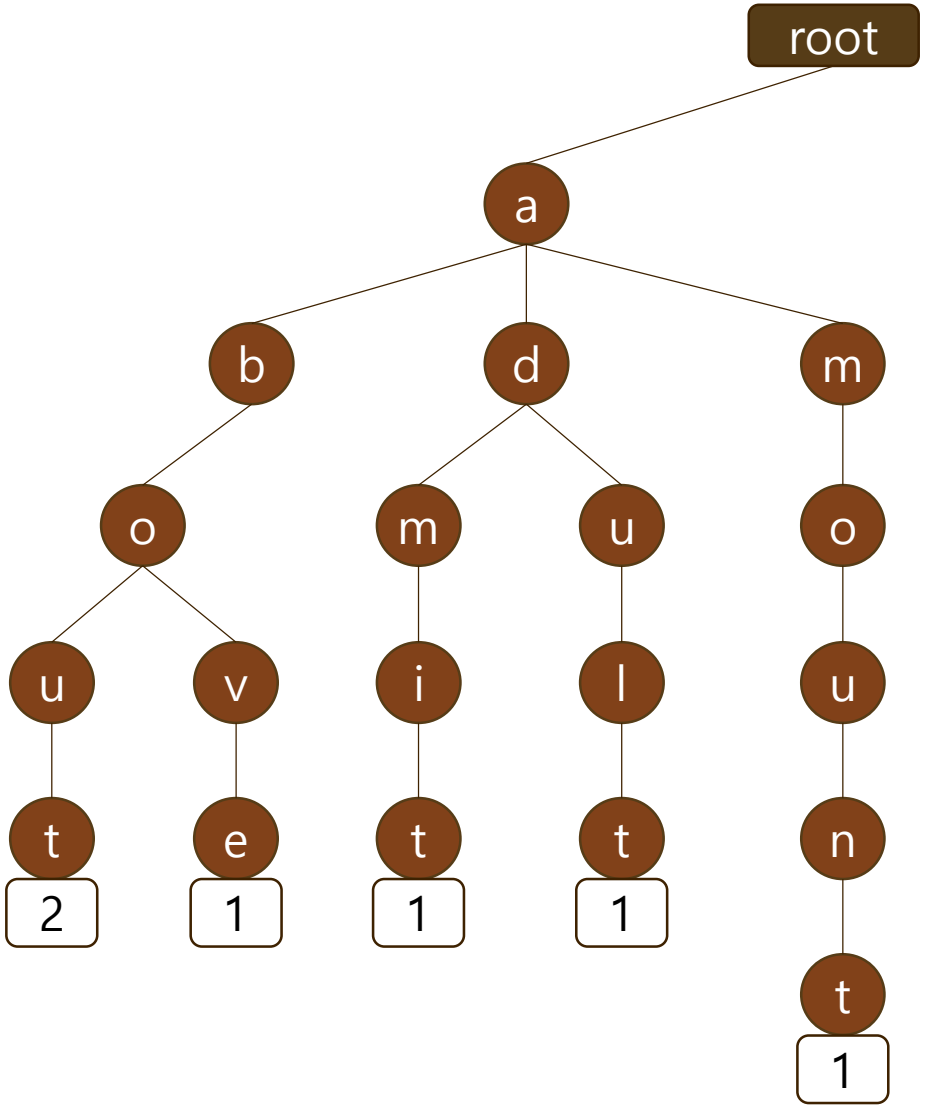
Problem analysis : 예제분석

1
2 about 1
2 about 2
2 above 1
4 abo?? 3
2 admit 1
2 adult 1
4 ad??t 2
2 amount 1
4 a???t 4
3 a?o?n? 1
2 avoid 1
4 a?o?? 4
3 about 2
4 a?o?? 2
2 visit 1
4 ???it 2
4 ?c?d? 0



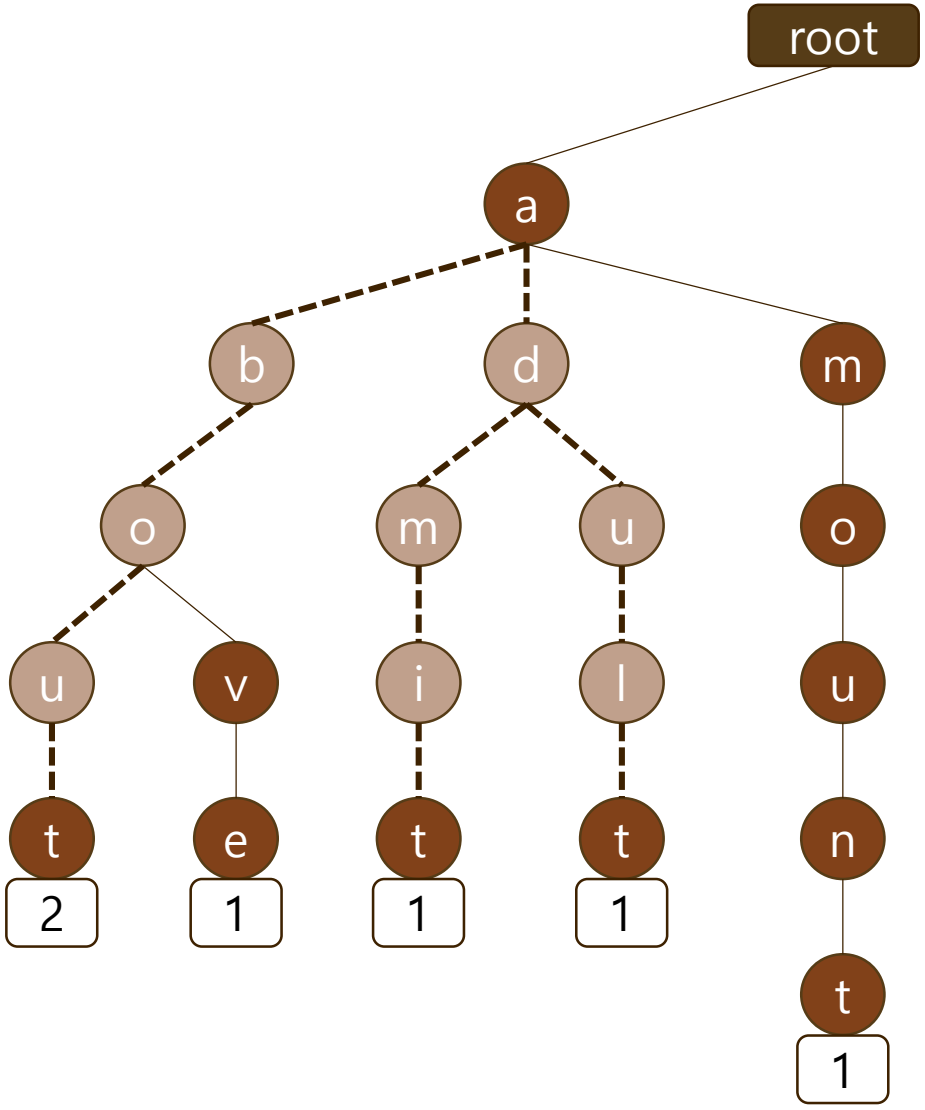
Problem analysis : 예제분석

1
2 about 1
2 about 2
2 above 1
4 abo?? 3
2 admit 1
2 adult 1
4 ad??t 2
2 amount 1
4 a???t 4
3 a?o?n? 1
2 avoid 1
4 a?o?? 4
3 about 2
4 a?o?? 2
2 visit 1
4 ???it 2
4 ?c?d? 0



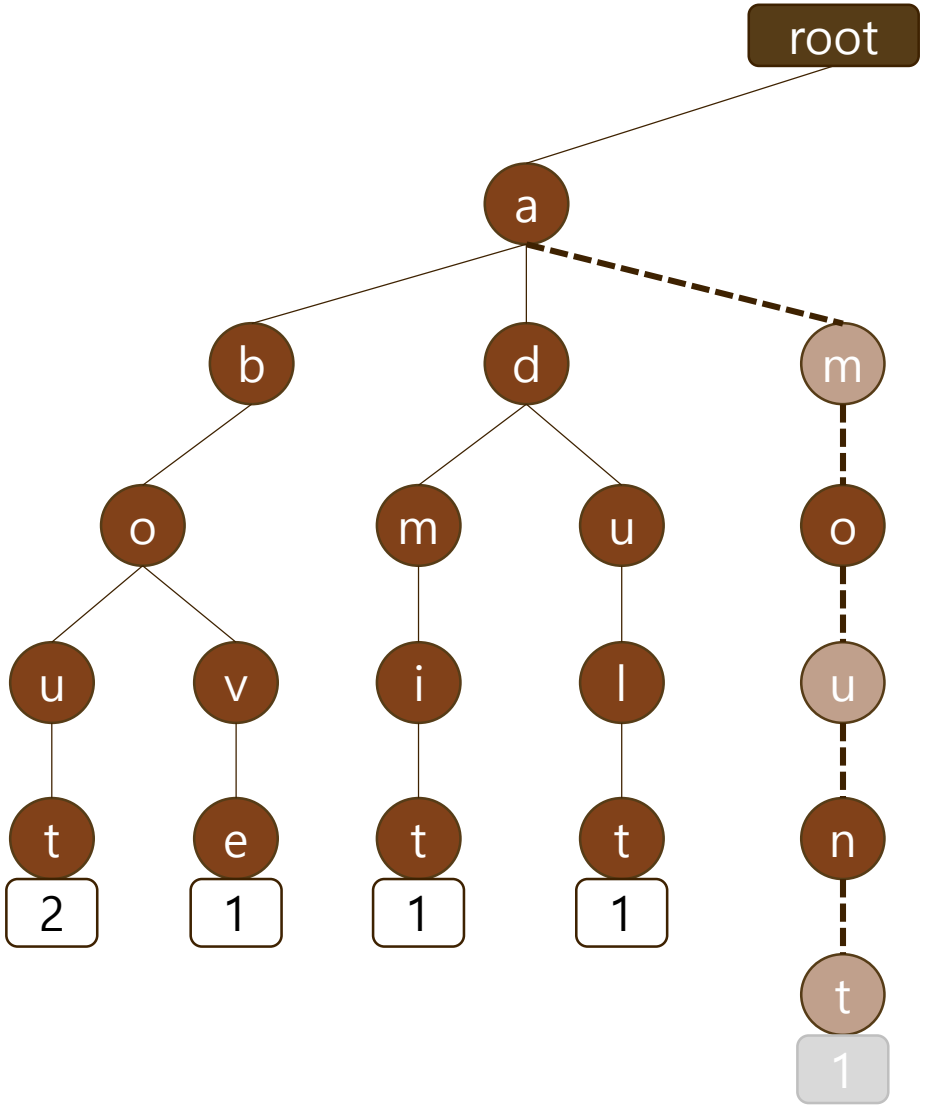
Problem analysis : 예제분석

1
2 about 1
2 about 2
2 above 1
4 abo?? 3
2 admit 1
2 adult 1
4 ad??t 2
2 amount 1
4 a???t 4
3 a?o?n? 1
2 avoid 1
4 a?o?? 4
3 about 2
4 a?o?? 2
2 visit 1
4 ???it 2
4 ?c?d? 0



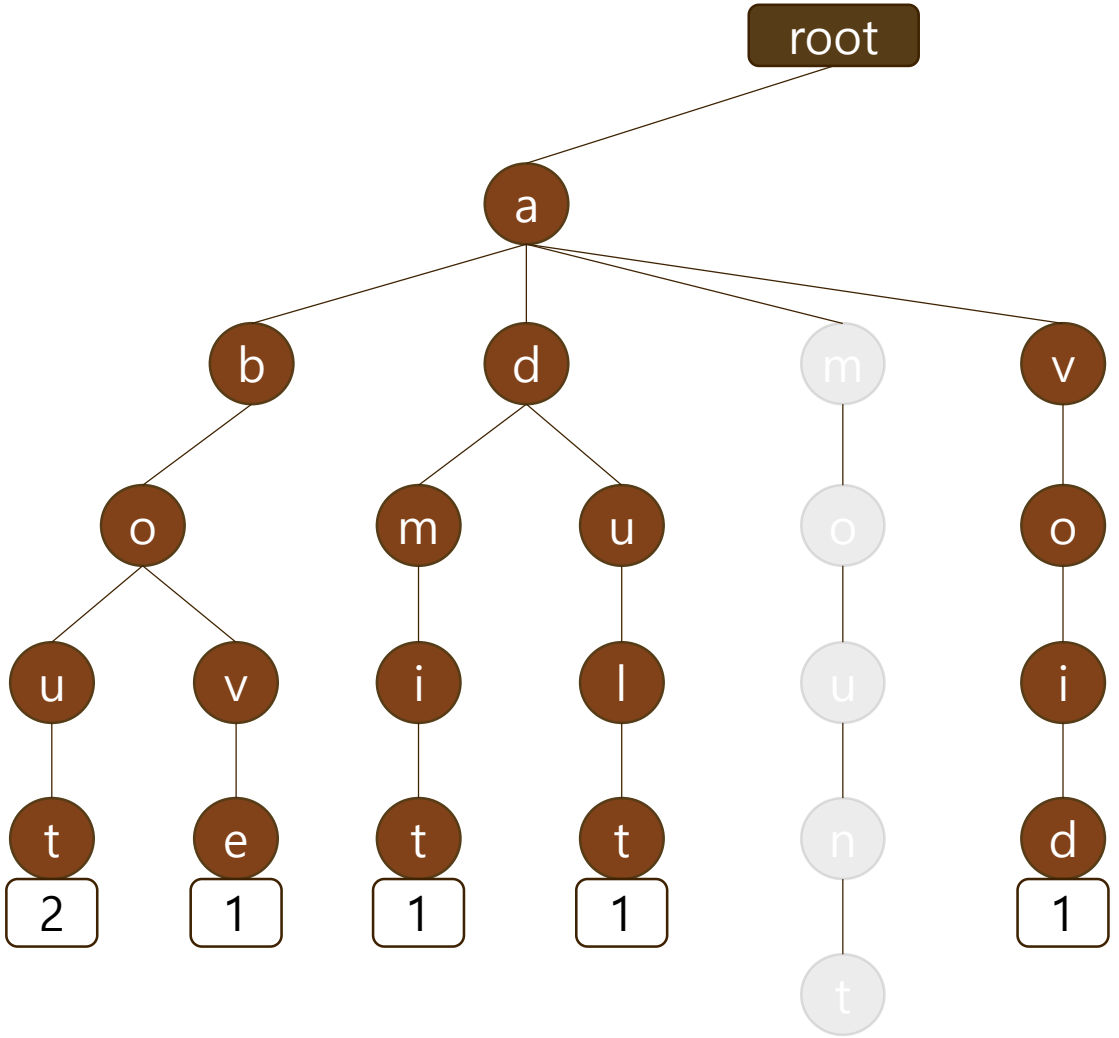
Problem analysis : 예제분석

1
2 about 1
2 about 2
2 above 1
4 abo?? 3
2 admit 1
2 adult 1
4 ad??t 2
2 amount 1
4 a???t 4
3 a?o?n? 1
2 avoid 1
4 a?o?? 4
3 about 2
4 a?o?? 2
2 visit 1
4 ???it 2
4 ?c?d? 0



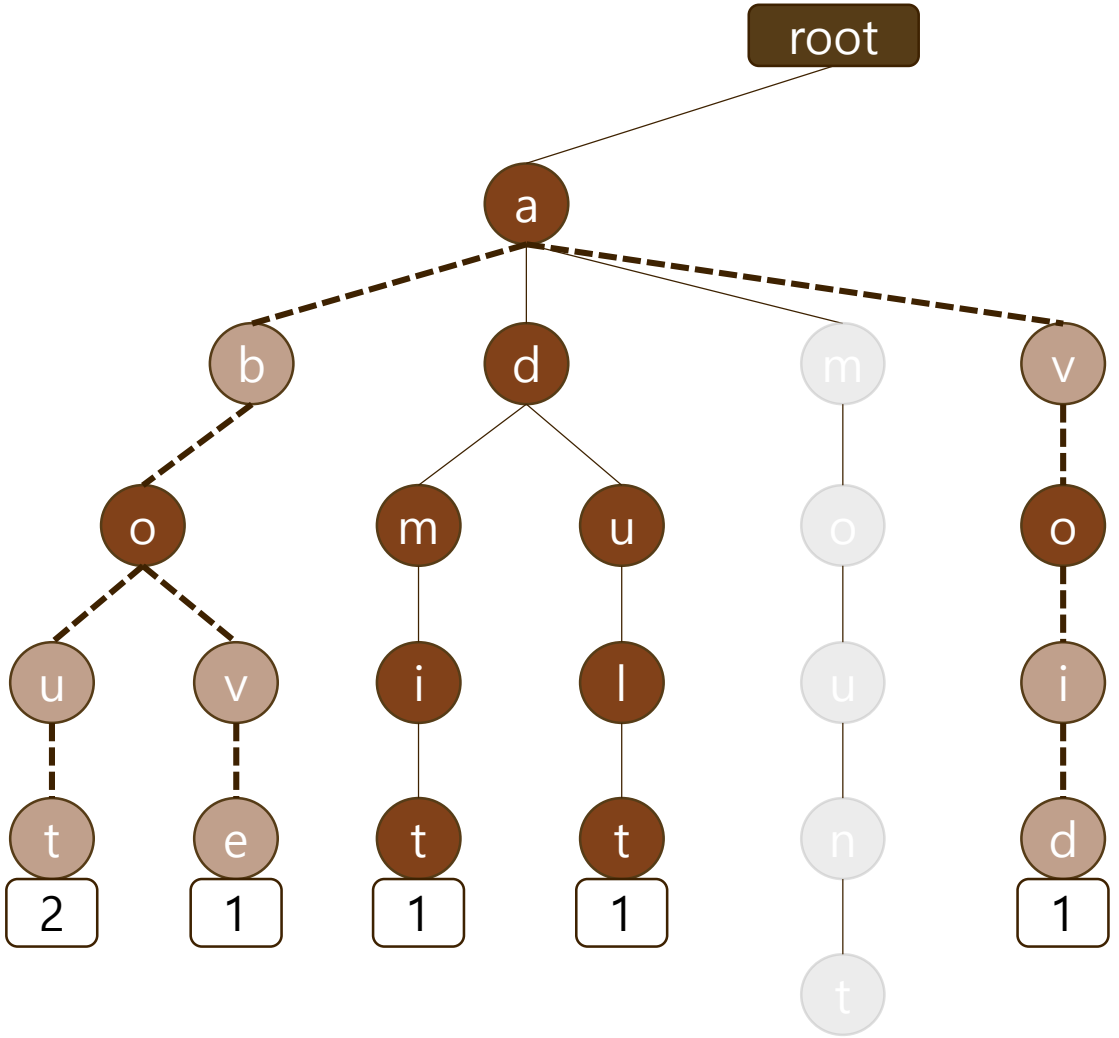
Problem analysis : 예제분석

1
2 about 1
2 about 2
2 above 1
4 abo?? 3
2 admit 1
2 adult 1
4 ad??t 2
2 amount 1
4 a???t 4
3 a?o?n? 1
2 avoid 1
4 a?o?? 4
3 about 2
4 a?o?? 2
2 visit 1
4 ???it 2
4 ?c?d? 0



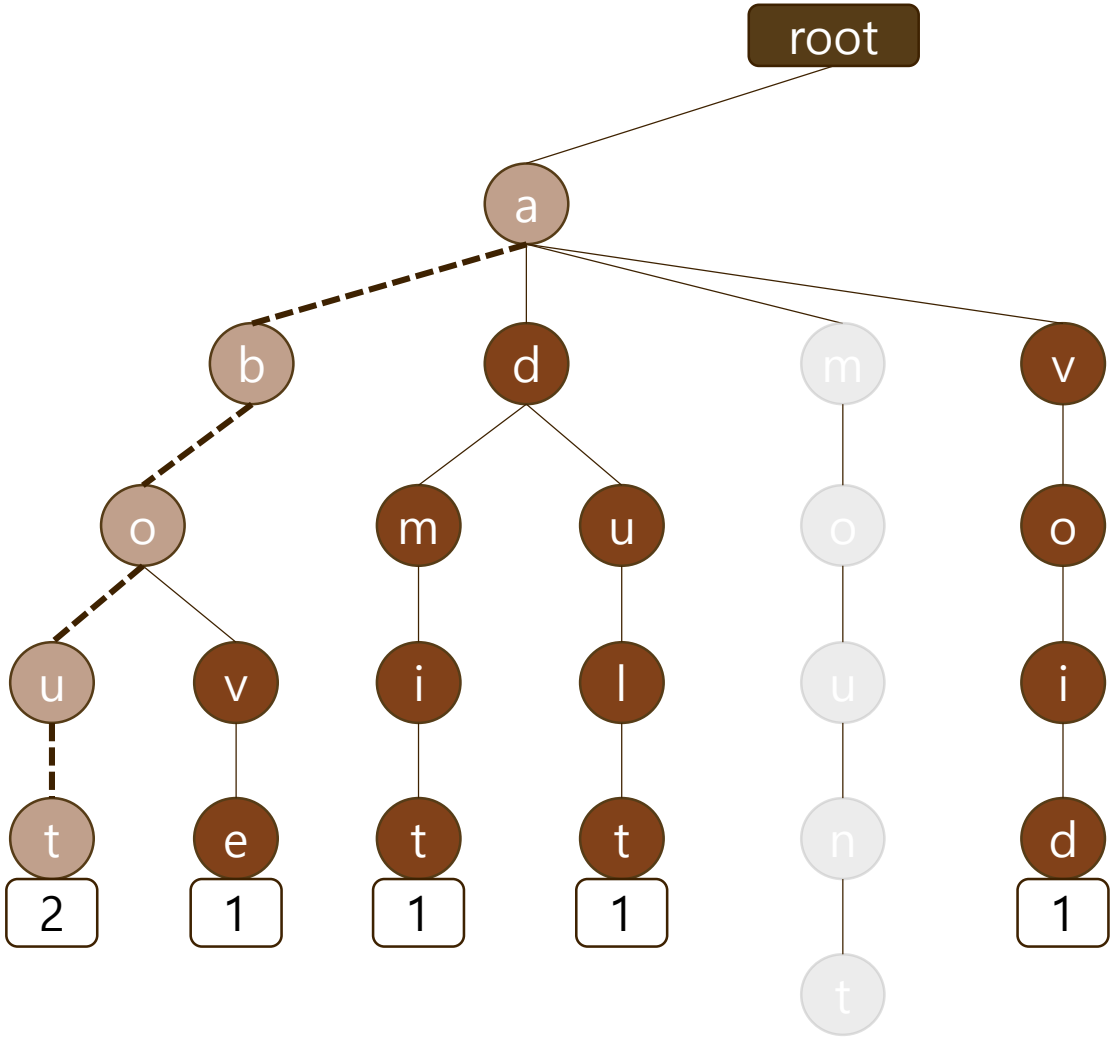
Problem analysis : 예제분석

1
2 about 1
2 about 2
2 above 1
4 abo?? 3
2 admit 1
2 adult 1
4 ad??t 2
2 amount 1
4 a???t 4
3 a?o?n? 1
2 avoid 1
4 a?o?? 4
3 about 2
4 a?o?? 2
2 visit 1
4 ???it 2
4 ?c?d? 0



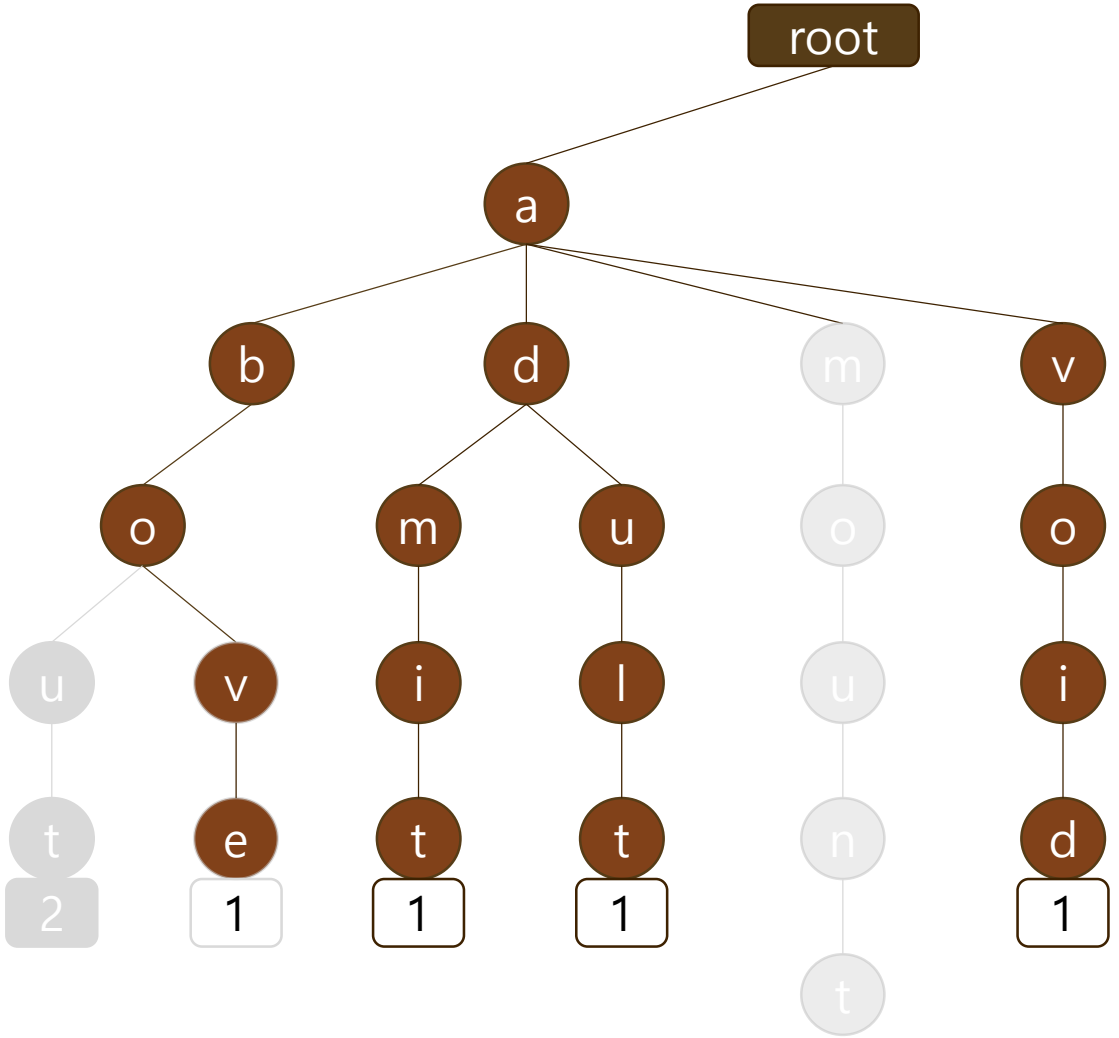
Problem analysis : 예제분석

1
2 about 1
2 about 2
2 above 1
4 abo?? 3
2 admit 1
2 adult 1
4 ad??t 2
2 amount 1
4 a???t 4
3 a?o?n? 1
2 avoid 1
4 a?o?? 4
3 about 2
4 a?o?? 2
2 visit 1
4 ???it 2
4 ?c?d? 0



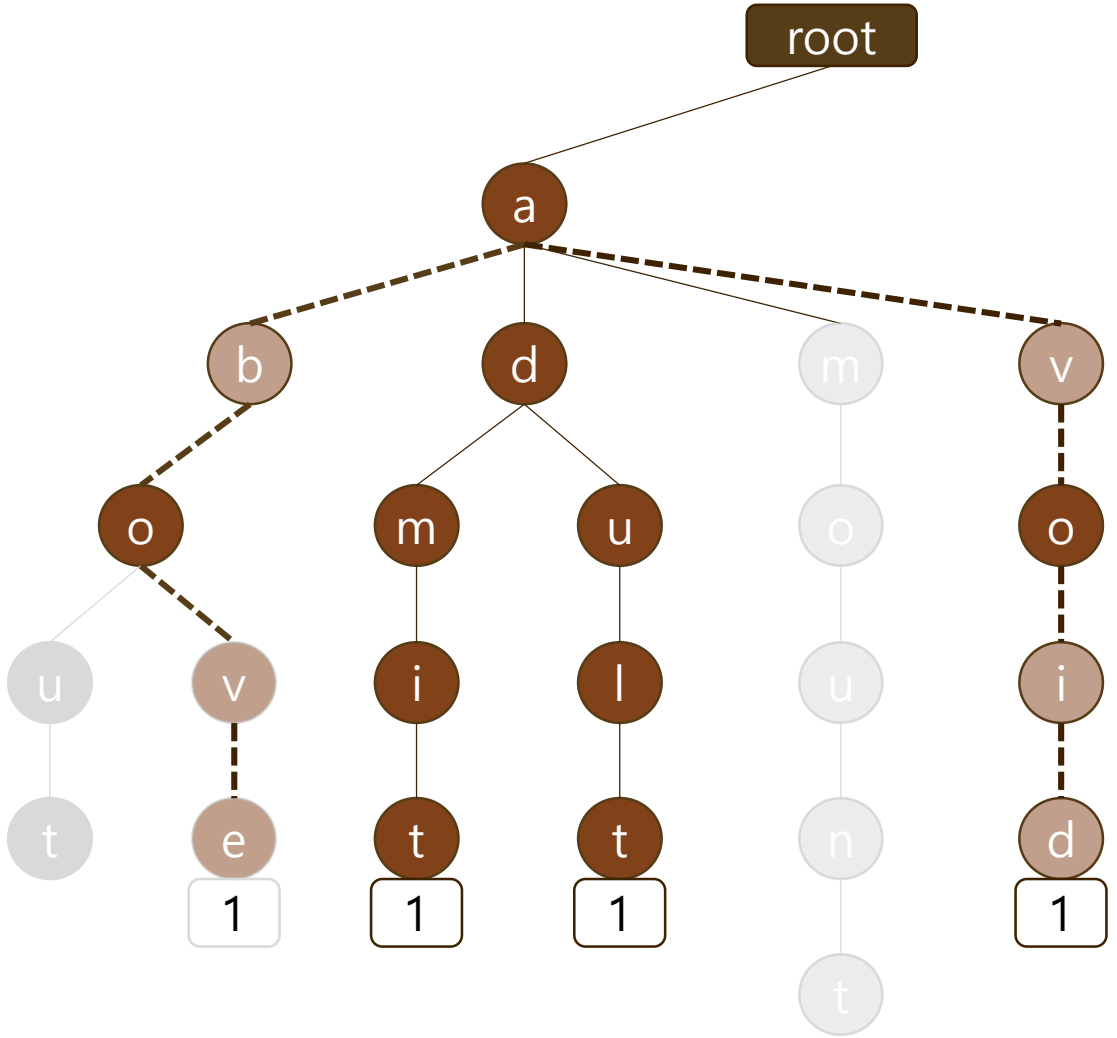
Problem analysis : 예제분석

1
2 about 1
2 about 2
2 above 1
4 abo?? 3
2 admit 1
2 adult 1
4 ad??t 2
2 amount 1
4 a???t 4
3 a?o?n? 1
2 avoid 1
4 a?o?? 4
3 about 2
4 a?o?? 2
2 visit 1
4 ???it 2
4 ?c?d? 0



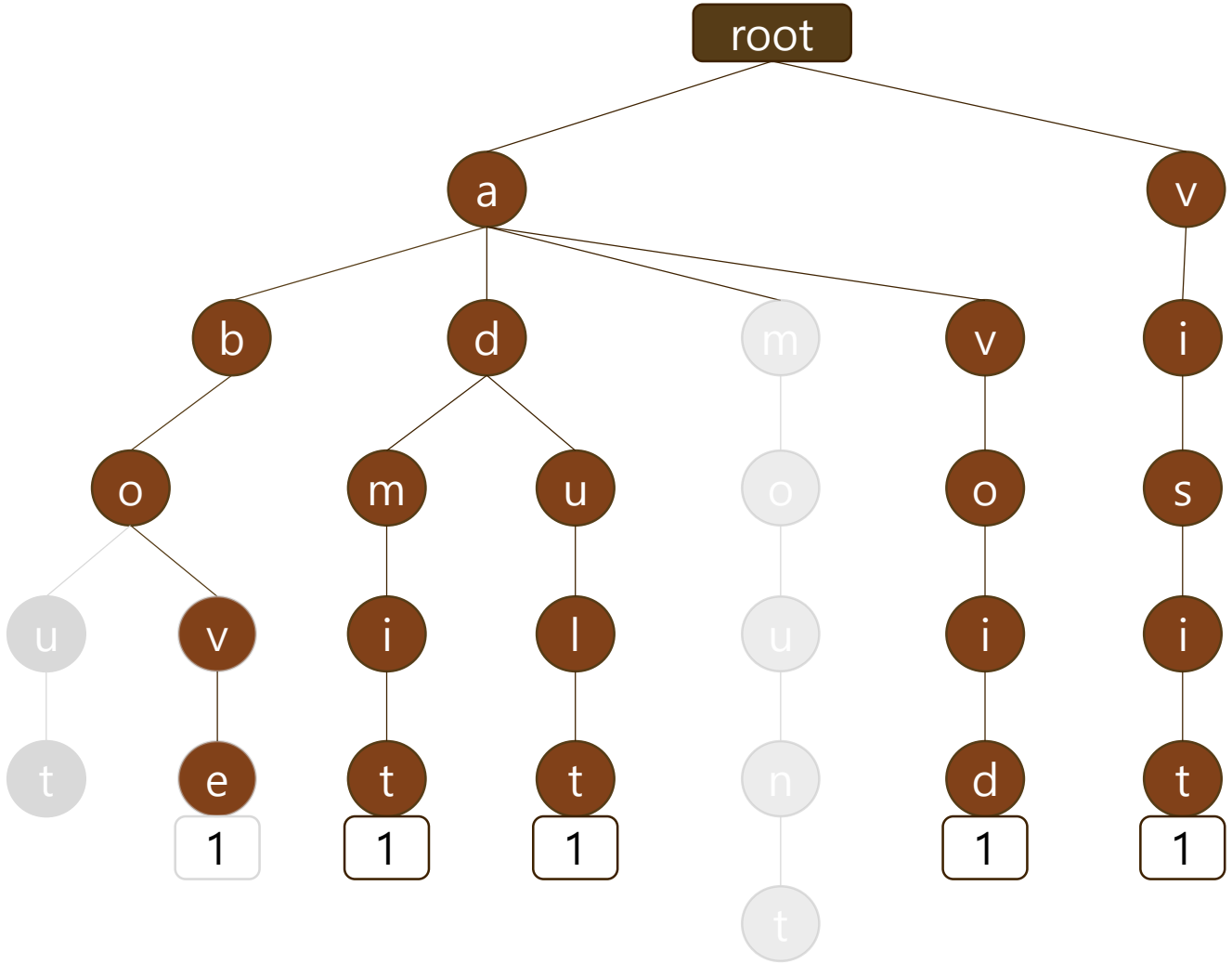
Problem analysis : 예제분석

1
2 about 1
2 about 2
2 above 1
4 abo?? 3
2 admit 1
2 adult 1
4 ad??t 2
2 amount 1
4 a???t 4
3 a?o?n? 1
2 avoid 1
4 a?o?? 4
3 about 2
4 a?o?? 2
2 visit 1
4 ???it 2
4 ?c?d? 0



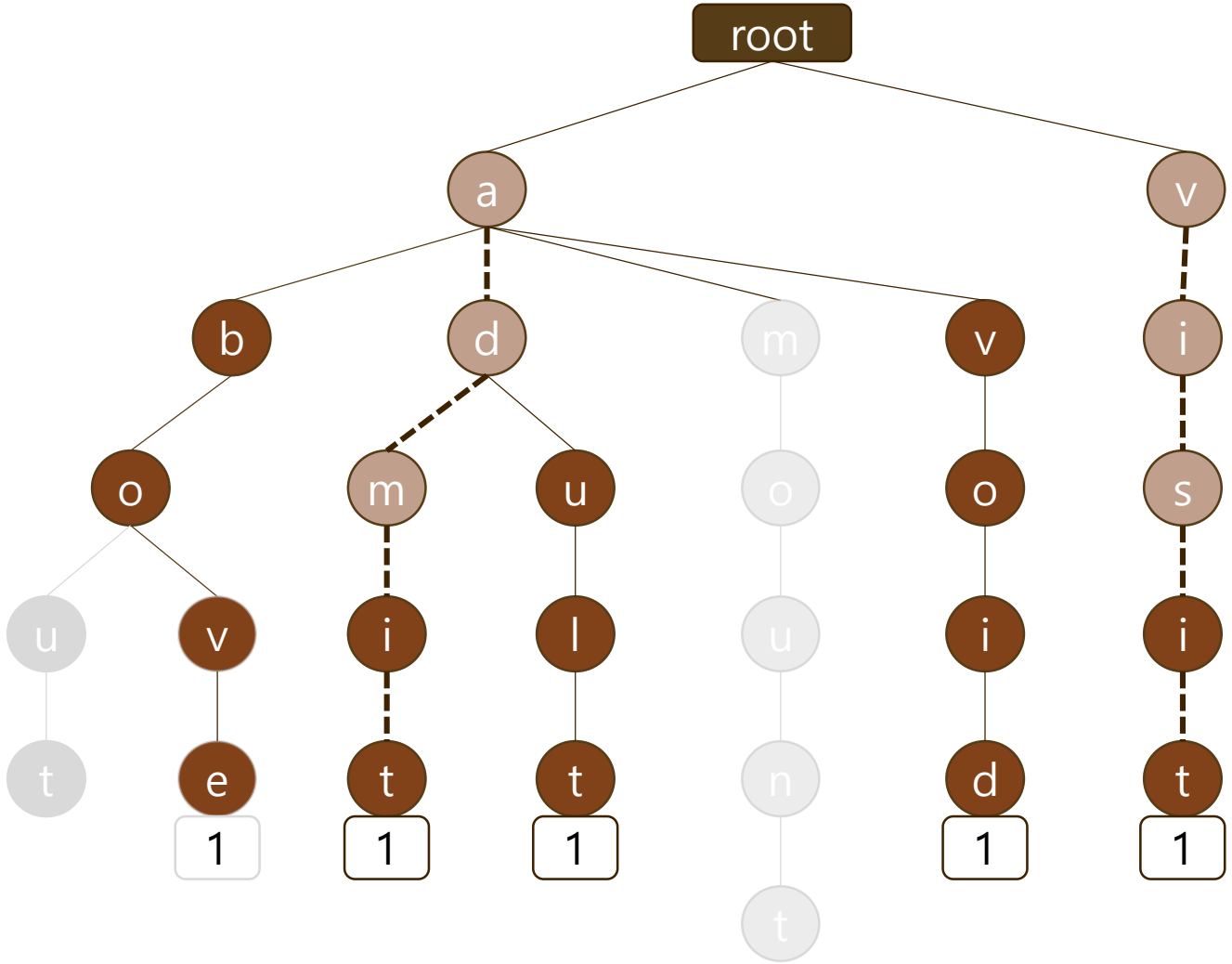
Problem analysis : 예제분석

1
2 about 1
2 about 2
2 above 1
4 abo?? 3
2 admit 1
2 adult 1
4 ad??t 2
2 amount 1
4 a???t 4
3 a?o?n? 1
2 avoid 1
4 a?o?? 4
3 about 2
4 a?o?? 2
2 visit 1
4 ???it 2
4 ?c?d? 0



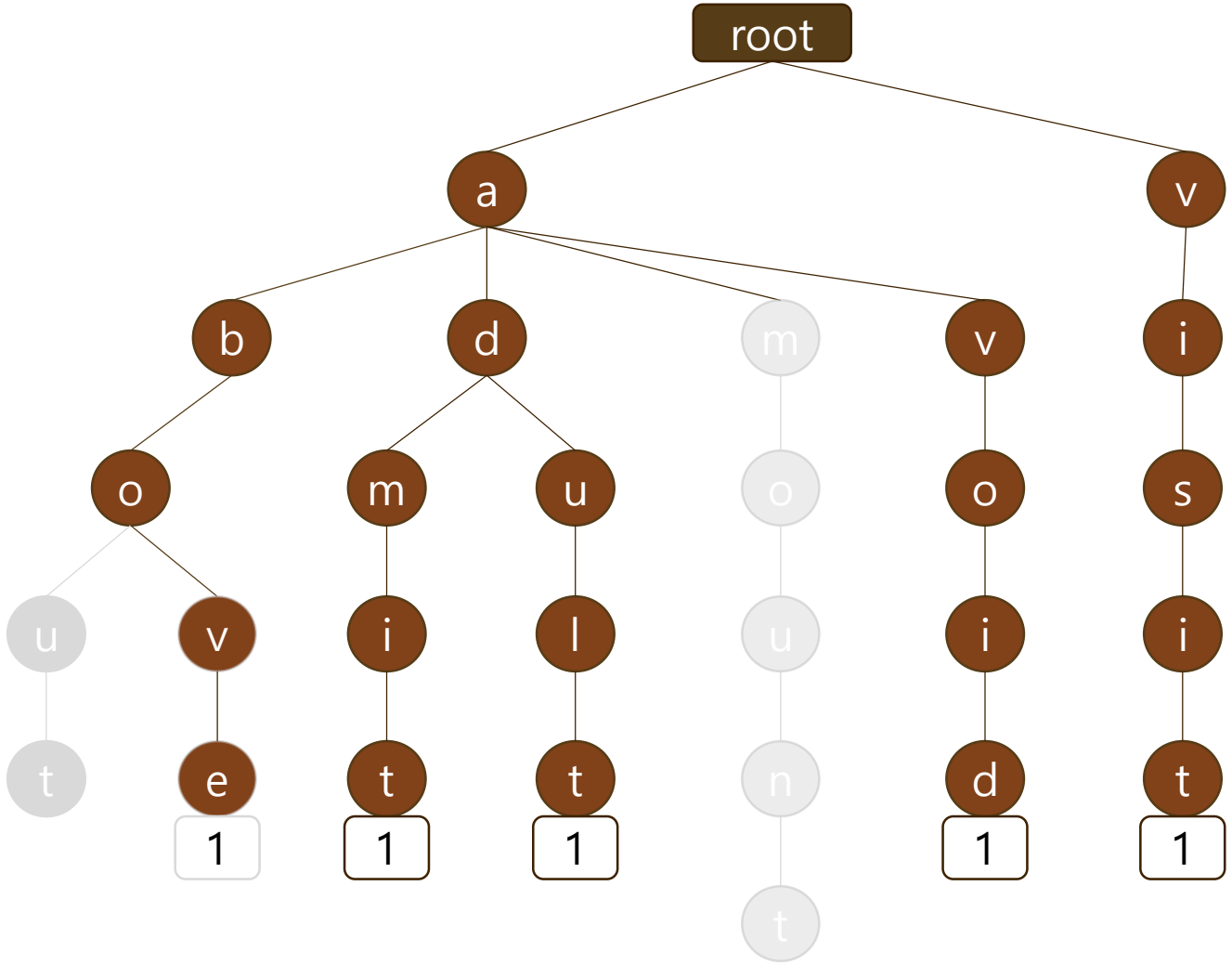
Problem analysis : 예제분석

1
2 about 1
2 about 2
2 above 1
4 abo?? 3
2 admit 1
2 adult 1
4 ad??t 2
2 amount 1
4 a???t 4
3 a?o?n? 1
2 avoid 1
4 a?o?? 4
3 about 2
4 a?o?? 2
2 visit 1
4 ???it 2
4 ?c?d? 0



Problem analysis : 예제분석

1
2 about 1
2 about 2
2 above 1
4 abo?? 3
2 admit 1
2 adult 1
4 ad??t 2
2 amount 1
4 a???t 4
3 a?o?n? 1
2 avoid 1
4 a?o?? 4
3 about 2
4 a?o?? 2
2 visit 1
4 ???it 2
4 ?c?d? 0



Problem analysis : 제약조건 및 API

- addStr()는 최대 2만번 호출되며 추가되는 문자열의 길이는 5~30이다.
- 문자열 탐색과 삭제에는 하나의 문자열에
  최대 3개의 와일드 카드 문자('?') 가 포함될 수 있다.
- 모든 함수 호출은 최대 4만번이다.

- 와일드 카드 문자 '?'를 고려하여 문자열을 탐색할 수 있어야 한다.
- 와일드 카드 문자 '?'를 고려하여 문자열을 탐색할 수 있도록 문자열이 추가되어야 한다.
- 어떤 자료구조를 생각 할 수 있을까?

Solution sketch

Solution sketch

- 문자열을 추가, 탐색, 삭제에 사용될 수 있는 자료구조는
  hash, trie 등이 있다.

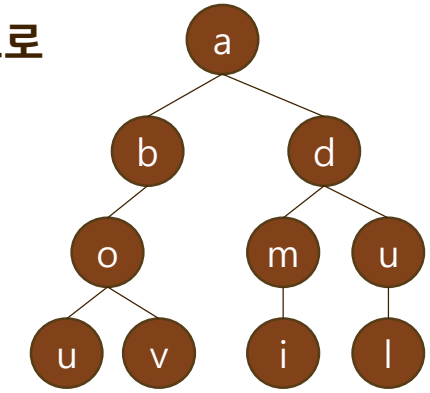
- 이 문제에 있어 hash를 사용하는 상황을 가정해 보자.
  1. addStr()이 호출될 때 와일드카드를 생각하지 않고 hash에 저장하는 경우
     - ✓ 탐색과 삭제에서 주어지는 문자열에 와일드 카드 문자가 포함된 경우
       최대 26 * 26 * 26 = 17,576 개의 후보 문자열을 만들어 확인해야 한다.

  2. addStr()이 호출될 때 와일드카드를 고려하여 hash에 저장하는 경우
     - ✓ 탐색과 삭제에서 주어지는 문자열에 와일드 카드 문자가 1개일때를 대비하여
       최대 30개의 문자열을 추가로 hash테이블에 저장할 수 있다.
     - ✓ 탐색과 삭제에서 주어지는 문자열에 와일드 카드 문자가 2개일때를 대비하여
       최대 30*29/2 = 435개의 문자열을 추가로 hash테이블에 저장할 수 있다.
     - ✓ 탐색과 삭제에서 주어지는 문자열에 와일드 카드 문자가 2개일때를 대비하여
       최대 30*29*28/6 = 4060개의 문자열을 추가로 hash테이블에 저장할 수 있다.

Solution sketch

- 위와 같이 해시를 사용하는 경우 시간복잡도가 비교적 높은 편이다.
  또한 구현에 따른 오버헤드가 문제 해결에 있어 효율성을 떨어뜨린다.

- 다음 trie를 사용하는 경우를 생각해 보자.
  - ✓ 접두 문자열이 같은 문자열은 하나의 계통으로 관리할 수 있다.
  - ✓ 와일드 카드 문자열을 고려하여 탐색하는데 어려움이 없다.
  - ✓ 빠른 탐색을 위하여 child 노드 인덱스(포인터)를 미리 확보해야 함으로
    많은 공간을 필요로 한다. (예 : 소문자라면 26개)
  - ✓ 이 문제에서는 단어수 * 문자종류 * 단어길이 만큼 필요하므로
    최대 20,000 * 26 * 30 = 15,600,000 개의 공간이
    사용될 수 있다.
  - ✓ 포인터를 사용한다면 여기에 8을
    정수 인덱스를 사용한다면 여기에 4를 곱해야 한다.



Solution sketch

✓ 사용가능한 메모리의 크기가 256MB이므로
포인터를 사용하든(약 120MB) 정수 인덱스를 사용하든(약 60MB) 문제되지 않는다.

[summary]

- 트라이는 예제 분석에서 살펴본 바와 같이
이 문제에 최적화된 자료구조로 볼 수 있다.

- 저장된 문자열에 대해서만 탐색하면서 와일드 카드를 함께 고려할 수 있다.

- 트라이 자료구조는 문자열 추가, 검색, 삭제에 유용하다.
- 준비해 둔다면 유용하게 사용할 수 있다.

Code example1

: Trie

int array based

Code example1


```
#define rint register int
const int LM = 20000 * 30; // 600,000

struct Trie {
    struct Node { int cnt, child[26]; }buf[LM];
    int bcnt;
    void init() { initBuf(bcnt = 1); }
    void initBuf(int idx) { buf[idx] = { 0 }; }
    int add(char*s, int cur = 1) {
        for (; *s; ++s) {
            int idx = *s - 'a';
            if (!buf[cur].child[idx]) initBuf(++bcnt), buf[cur].child[idx] = bcnt;
            cur = buf[cur].child[idx];
        }
        return ++buf[cur].cnt;
    }
```

Code example1


```
    int search(char*s, int flag = 0, int cur = 1) {
        if (!cur) return 0;
        int ret = 0;
        if (!*s) {
            ret = buf[cur].cnt;
            if (flag) buf[cur].cnt = 0;
            return ret;
        }
        if (*s == '?') {
            for (rint i = 0; i < 26; ++i) if (buf[cur].child[i])
                ret += search(s + 1, flag, buf[cur].child[i]);
        }
        else ret += search(s + 1, flag, buf[cur].child[*s - 'a']);
        return ret;
    }
}trie;
```

Code example1


```
void init() {
    trie.init();
}
int addStr(char str[]) {    // 20'000
    return trie.add(str);
}
int deleteStr(char str[]) {
    return trie.search(str, 1);
}
int searchStr(char str[]) {
    return trie.search(str);
}
```

Code example2

: Trie

pointer based

Code example 2

TS문자열관리

```
#define rint register int
const int LM = 20000 * 30; // 600,000

struct Trie {
    struct Node {
        int cnt;
        Node*child[26];
    }buf[LM];
    int bcnt;
    void init() { initBuf(bcnt = 1); }
    void initBuf(int idx) { buf[idx] = { 0 }; }
    int add(char*s) {
        Node*cur = buf + 1;
        for (; *s; ++s) {
            int idx = *s - 'a';
            if (!cur->child[idx]) initBuf(++bcnt), cur->child[idx] = buf + bcnt;
            cur = cur->child[idx];
        }
        return ++cur->cnt;
    }
```

Code example 2


```
    int search(char*s, int flag, Node*cur) {
        if (!cur) return 0;
        int ret = 0;
        if (!*s) {
            ret = cur->cnt;
            if (flag) cur->cnt = 0;
            return ret;
        }
        if (*s == '?') {
            for (rint i = 0; i < 26; ++i) if (cur->child[i])
                ret += search(s + 1, flag, cur->child[i]);
        }
        else ret += search(s + 1, flag, cur->child[*s - 'a']);
        return ret;
    }
}trie;
```

Code example 2


```
void init() {
    trie.init();
}
int addStr(char str[]) {    // 20'000
    return trie.add(str);
}
int deleteStr(char str[]) {
    return trie.search(str, 1, trie.buf + 1);
}
int searchStr(char str[]) {
    return trie.search(str, 0, trie.buf+1);
}
```

Thank you.