Trie & KMP

문자열 알고리즘이란?

- 문자열 (`string`)
  - 문자를 순차 나열한 수열
  - 문자는 **대소비교(<, >, =)** 만 가능하다! (사칙연산은 불가능)

- 문자열로 할 수 있는 것들
  - 매칭(일치/불일치 판단)
  - 사전 순(lexicographical) 비교
  - 문자열 결합(concatenate), 부분 문자열(substring), ...

문자열 알고리즘이란?

- 다른 알고리즘과의 차이점
  - 문자는 비교 연산 외에 다른 연산이 불가능하다.
  - 입력을 받은 값을 수정하지 않고 비교만 진행한다면 높은 확률로 문자열 알고리즘으로 해결을 시도하자.

→ 배열의 값이 숫자더라도 사칙연산을 사용하지 않으면 문자열 알고리즘을 의심하자!

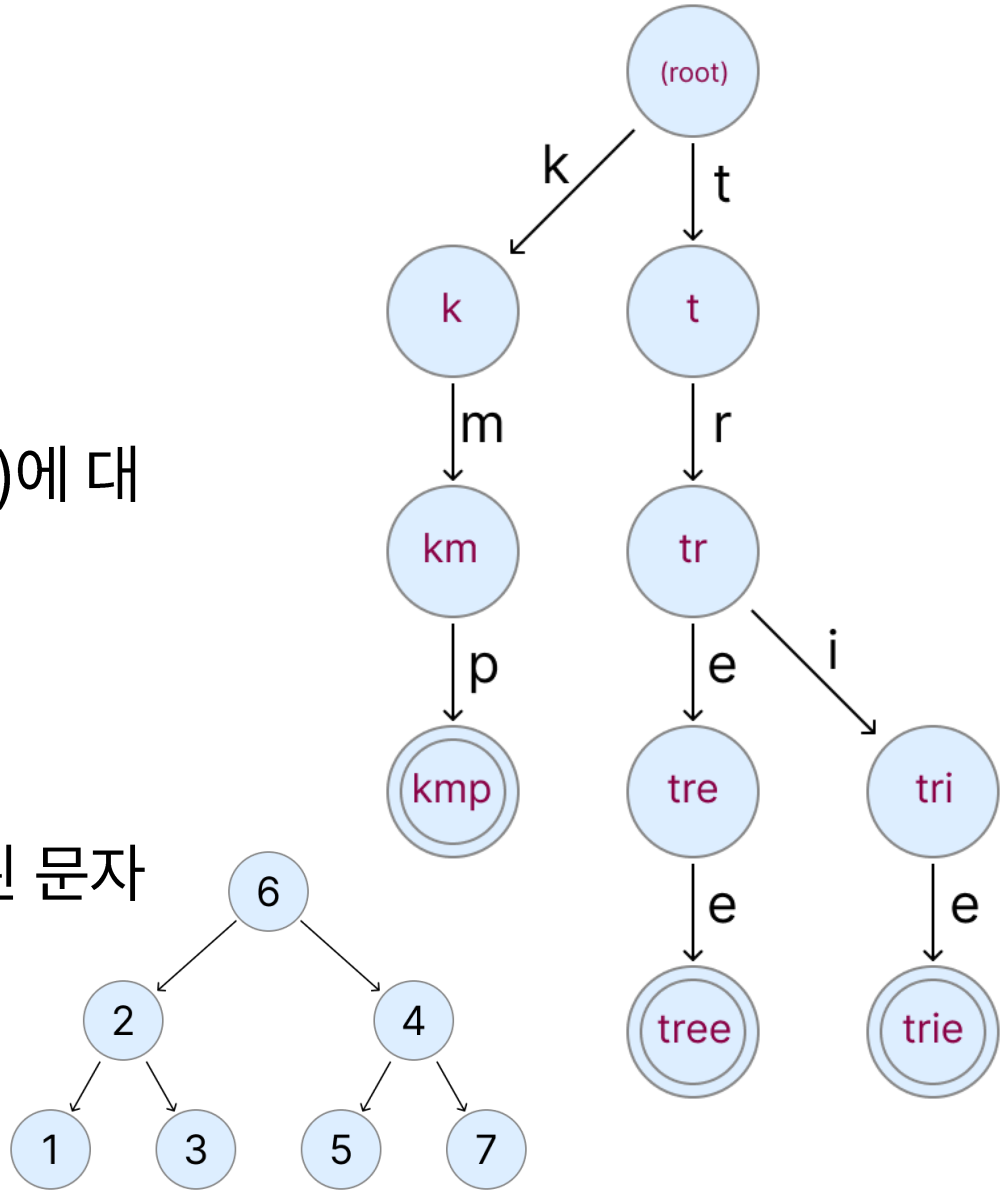
Trie에 들어가기 앞서서...

- 건국대학교 전교생의 데이터를 가지고 있다고 한다면...
- 이름이 건구스인 사람은 몇 명일까? 김으로 시작하는 사람은 몇 명일까?
- ... 여러가지 방법이 있지만, 가장 안정적이고 빠른 방법은?

<출처 : 건대신문>

Trie (트라이)

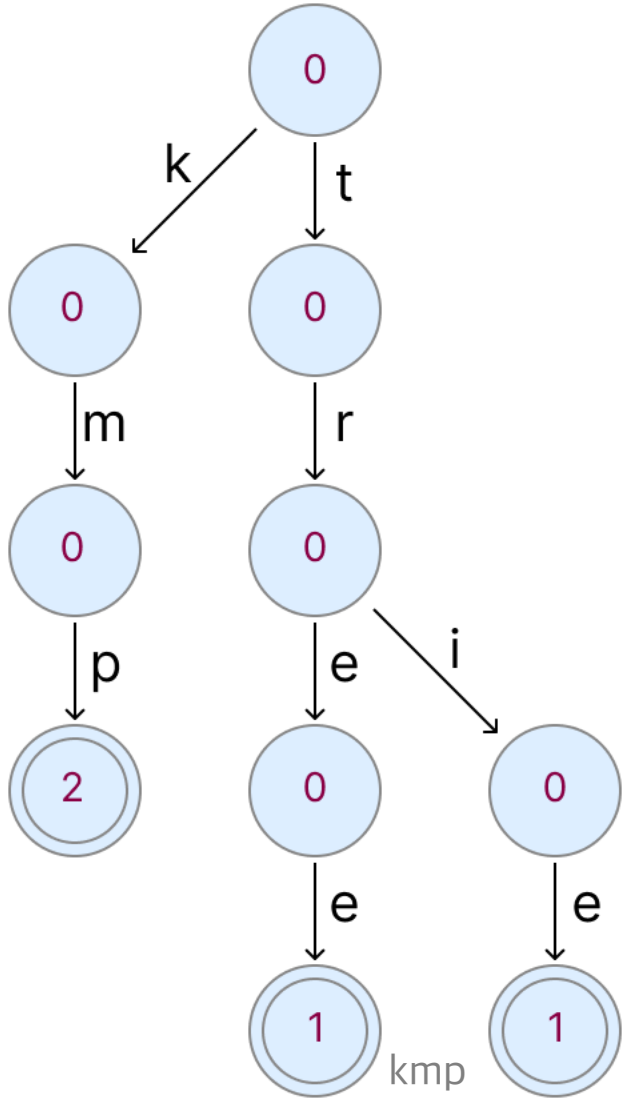
- Trie는 문자열을 트리로 나타낸 **자료구조**
- 일반적인 탐색 트리와 다르게, 노드가 문자열(key)에 대한 데이터를 가지고 있지 않다.
- 노드의 위치가 데이터 그 자체
- 완전히 일치하는 문자열이 존재하는 노드에 매칭된 문자열의 개수를 저장한다.



일반적인 탐색 트리

Trie (트라이)

- Search
  - 트라이를 탐색하면서 노드가 없을 경우, 매칭되는 문자열이 없는 것이다.
  - 문자열을 모두 탐색한 뒤에, 마지막 노드에 매칭된 문자열의 유무를 확인
- Update
  - 트라이를 탐색하면서 노드가 없을 경우, 노드를 생성한 뒤에 탐색한다.
  - 문자열을 모두 탐색한 뒤에, 마지막 노드에 매칭된 문자열의 수를 늘린다.

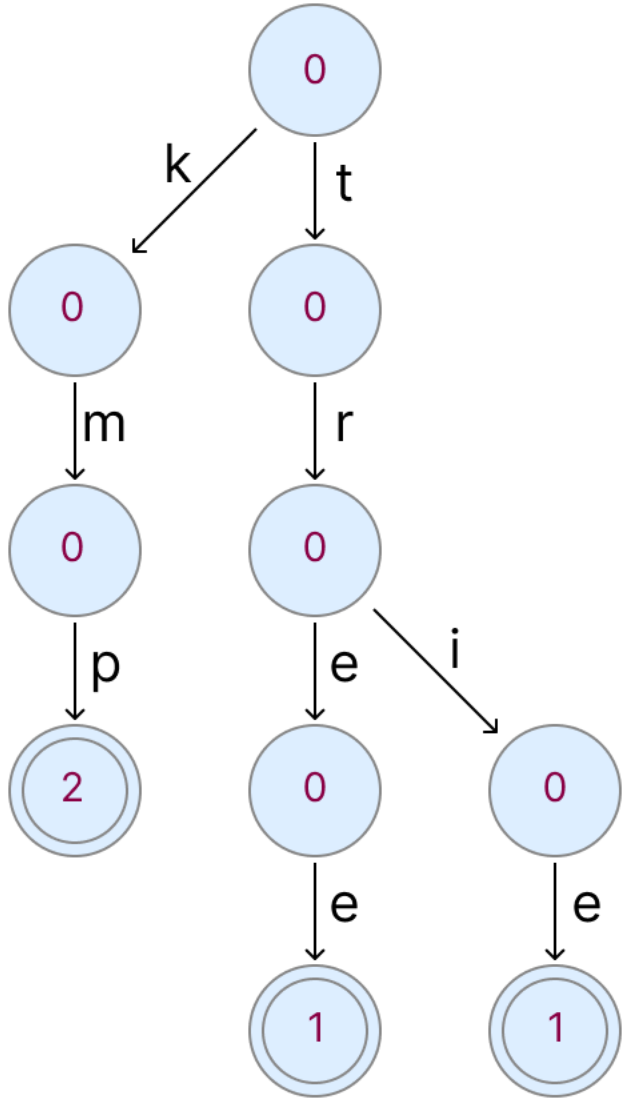


대응되는 key 값 : kmp kmp tree trie

Trie (트라이)

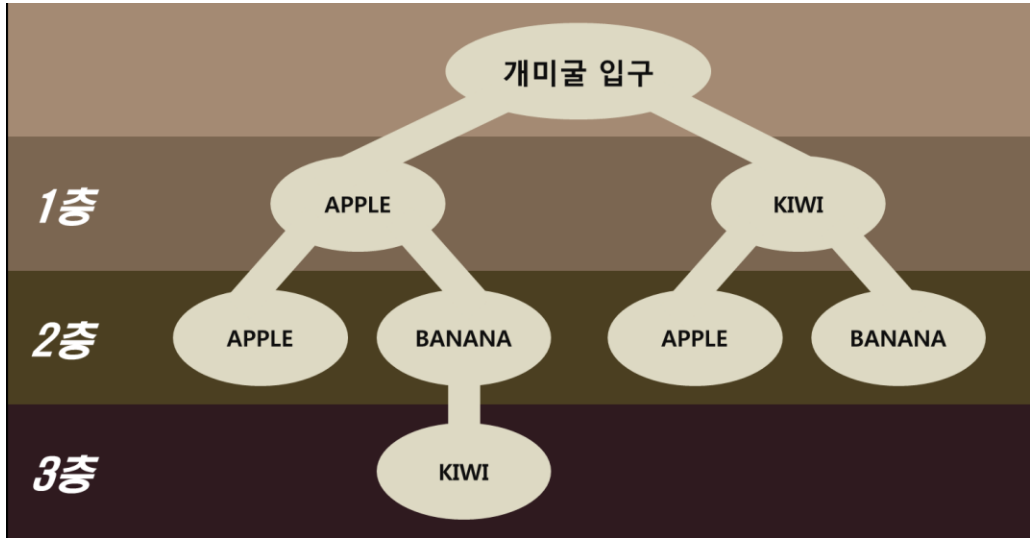
```
struct Trie {
    Trie *children[26];
    int cnt;
    void update(const string &str, int idx = 0) {
        if (idx >= str.length()) {
            cnt++;
            return;
        }
    if (!children[str[idx] - 'a’])
        children[str[idx] - 'a'] = new Trie();
    children[str[idx] - 'a']->update(str, idx + 1);

    // further search / query function below..
}
```



개미굴 BOJ 14725

- 로봇 개미는 트리 모양의 개미굴을 탐색한다.
- 로봇 개미는 개미굴의 저장소를 지날 때 마다 지나온 저장소를 기록한다.
- 전체 개미굴의 구조를 출력하시오.



개미굴 BOJ 14725

- 각 먹이마다 Trie를 업데이트 하면서 이동한다.
- 단, edge를 구분하는 값이 string이므로, 적절한 자료구조에 저장한다.
- unordered_map, map 등을 사용할 수 있다.

수열과 쿼리 20 BOJ 16903

문제

0이 하나 포함되어 있는 배열 A가 있다. 이때, 다음 쿼리를 수행해야 한다.

- 1 x : A에 x를 추가한다.
- 2 x : A에서 x를 제거한다. A에 x가 두 개 이상 있는 경우에는 하나만 삭제한다. 항상 A에 x가 있는 쿼리만 주어진다.
- 3 x : A에 포함된 각각의 원소와 x를 XOR 연산을 한 다음, 가장 큰 값을 출력한다.

입력

첫째 줄에 쿼리의 개수 M($1 \le M \le 200,000$)이 주어진다. 둘째 줄부터 M개의 줄에 쿼리가 주어진다. 입력으로 주어지는 x의 범위는 10^9보다 작거나 같은 자연수이다.

3번 쿼리는 하나 이상 주어진다.

수열과 쿼리 20 BOJ 16903

- 배열에 추가 순서는 관계가 없으므로, 추가한 값과 그 개수만을 저장하면 된다.
- XOR은 비트 연산이므로, 각 2진수 자릿수는 서로 독립이다.
- 특정 값 기준으로 XOR한 값이 큰 순서는 무엇일까?

수열과 쿼리 20 BOJ 16903

- 수는 높은 자릿수가 크면 큰 값이다. 높은 자리부터 x와 다른 수를 찾아야 한다.
- 높은 자리부터 한 자리씩 확인하면서 이동하므로, Trie를 사용하여 해결할 수 있다.
- 다만, 각 노드 아래에 몇 개의 값이 있는지 저장할 필요가 있다.

- 해당 자리 수를 1로 만들 수 있는 수가 1개 이상 있다면, 그 수를 선택한다.
- 아니라면, 0으로 만드는 수 중에서 선택한다.

문자열 검색 알고리즘

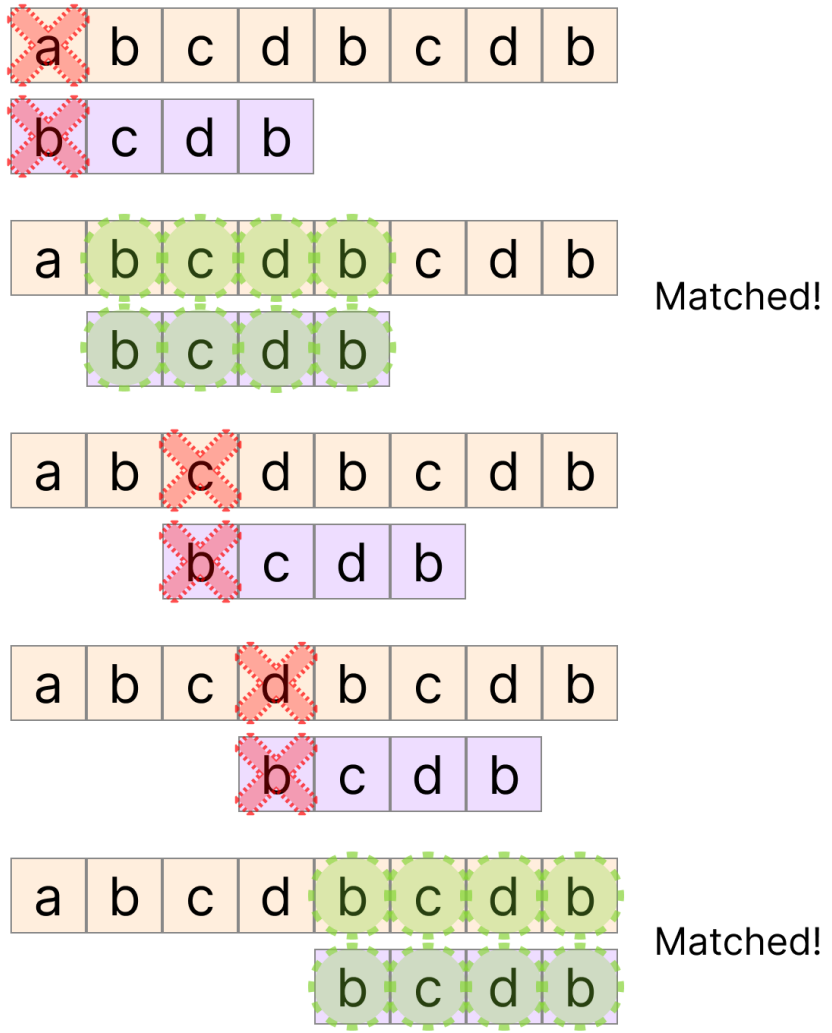
- "문자열 s 내에서 특정 문자열 p가 어디서 등장하는가?"

- **A B C D B C D B C D**
- **D B C D**

- $search(s, p) = [3, 6]$ → 3, 6번째 문자부터 p가 등장한다. (0-based)

문자열 검색 알고리즘 (Naïve)

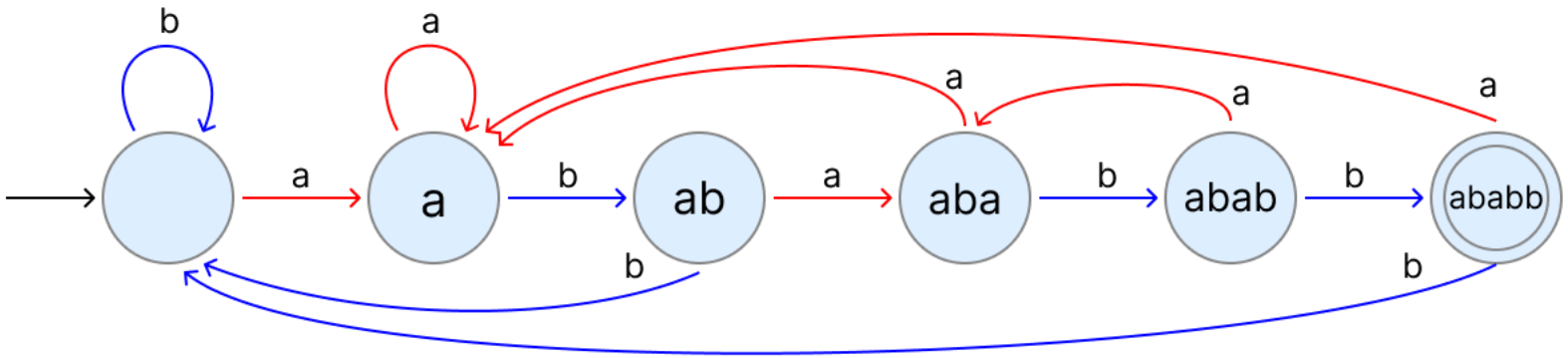
- "문자열 s 내에서 특정 문자열 p가 어디서 등장하는가?"
- **A B C D B C D B C D**
- **D B C D**
- s 의 같은 문자를 여러 번 확인할 수 있다
- 시간 복잡도 $O(|s| \times |p|)$
- 시간을 줄일 수 있는 방법은 없을까?



Knuth-Morris-Pratt

- 생각할 수 있는 좋은 알고리즘의 시간 복잡도 : $O(|s| + |p|)$
- 문자열 검색에 실패할 때, 지금까지의 매칭 정보를 활용해서 이미 알고 있는 부분은 비교하지 않는다.
- 특정 $pattern$이 포함되는지 확인할 수 있는 **기계**를 만든다!

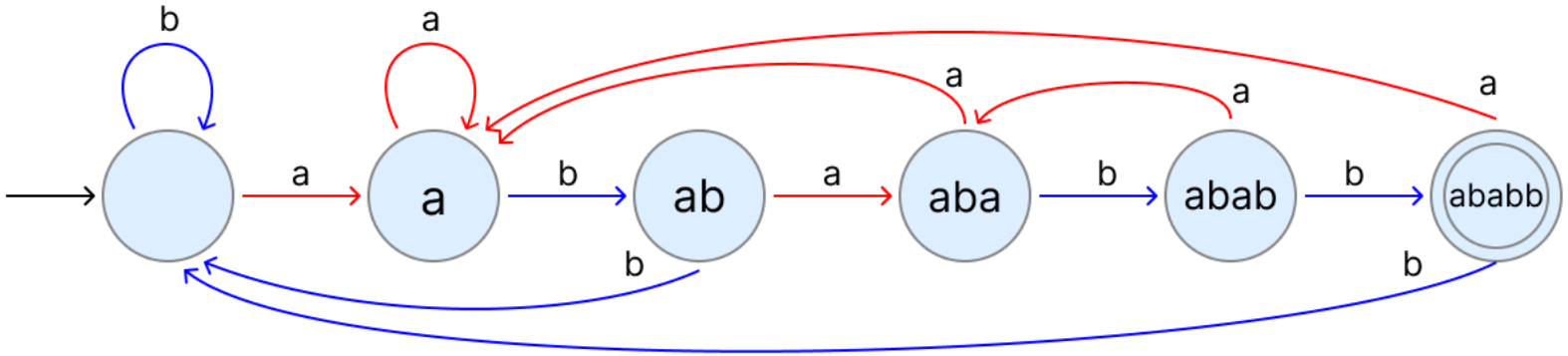
KMP Algorithm



※ a, b 이외의 문자는 전부 첫 노드로 이어짐.

- $s = \text{“}ababbabb\text{”}$ 를 이용하여 매칭을 확인해보자.
- 이 **기계**를 빠르게 만들 수 있다면, 빠르게 문자열 검색을 할 수 있다!

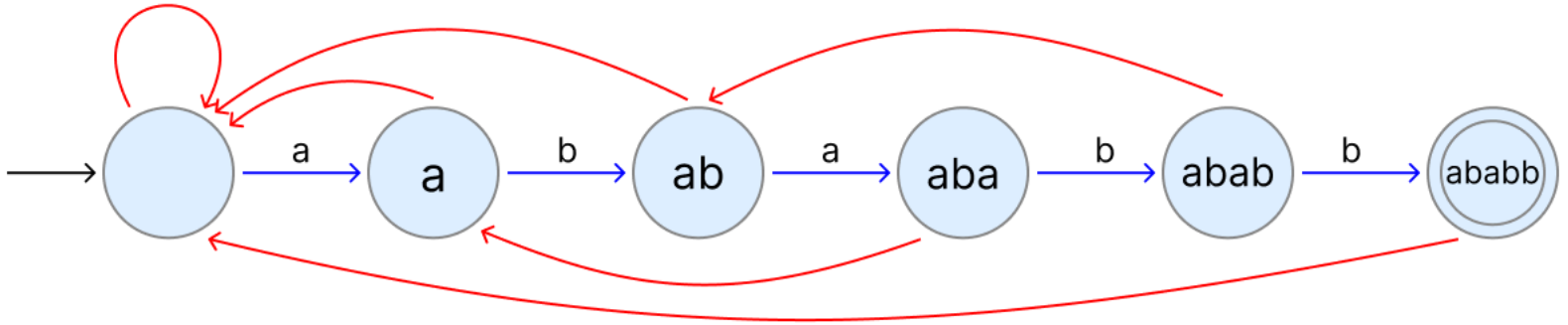
KMP Algorithm



※ a, b 이외의 문자는 전부 첫 노드로 이어짐.

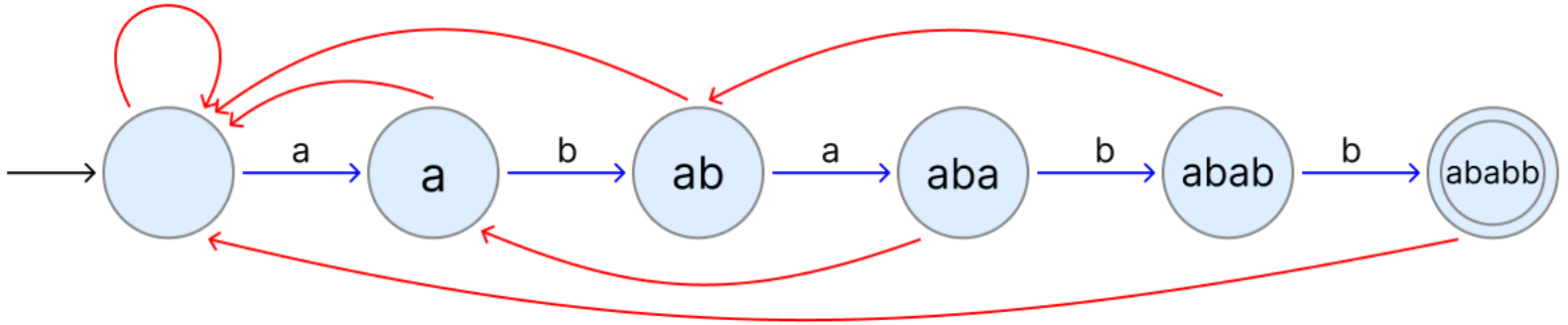
- 모든 노드가 문자의 종류만큼 $edge$를 가지고 있다.
- 이 간선들은 대부분 비슷한 노드로 향한다. $O(|s| + |p| \times |\Sigma|)$ * |Σ| : 모든 가능한 문자의 개수
- 이를 최적화할 수는 없을까?

KMP Algorithm



- $s = \text{“}ababababb\text{”}$ 를 이용하여 매칭을 확인해보자.
- Failure function : 매칭에 실패했을 경우 이동한다.
- 이동하는 과정을 매칭에 성공하거나, 첫 노드에 도달할 때까지 반복한다.

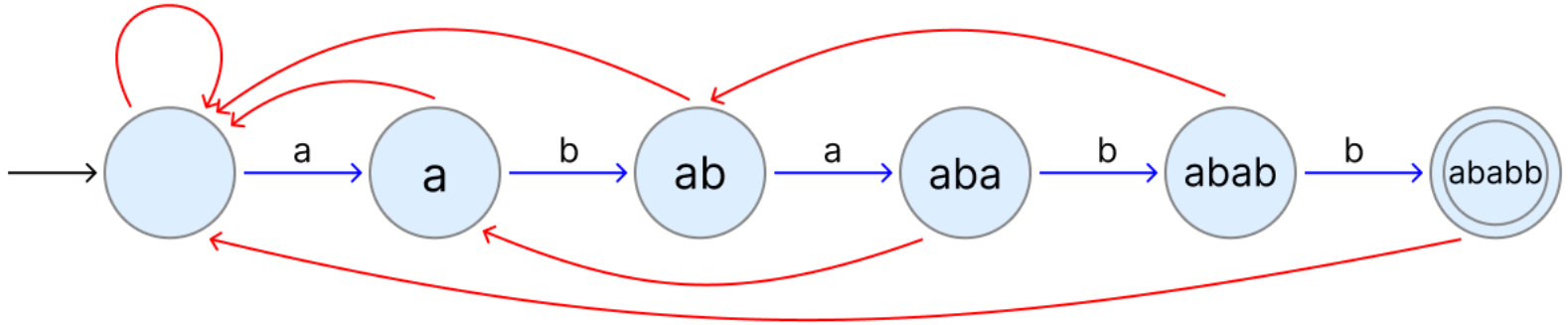
KMP Algorithm



- Failure function을 활용해서 모든 문자에 대한 $edge$를 만드는 것이 아니라, 1개의 $edge$로만 모든 동작을 하도록 한다.
- 시간 복잡도 : $O(|s| + |p|)$
- 이제 이 **기계**를 $O(|p|)$ 시간 내에 만들면 된다!

기계 사용하기

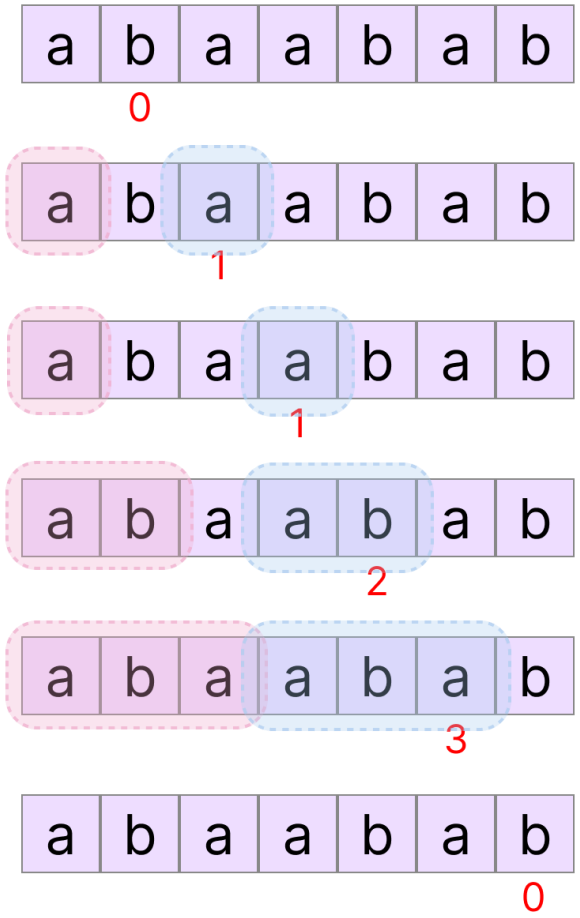
- 이 기계를 저장하는 자료구조를 생각해보자.
- 모든 간선은 다음 노드를 향하고, 노드의 개수도 패턴의 길이와 비슷하다.
- 따라서 다음과 같이 **Failure function이 가리키는 노드 번호만 저장**해도 된다.



$\rightarrow \{0, 0, 1, 2, 0\}$

Failure Function (pi)

- Failure function은 매칭에 실패했을 때, 가장 많은 패턴을 포함하여 문자열과 일치하는 노드 번호를 가리킨다.
- 어떤 문자열의 접두사와 접미사가 같은 길이 중, 가장 긴 길이

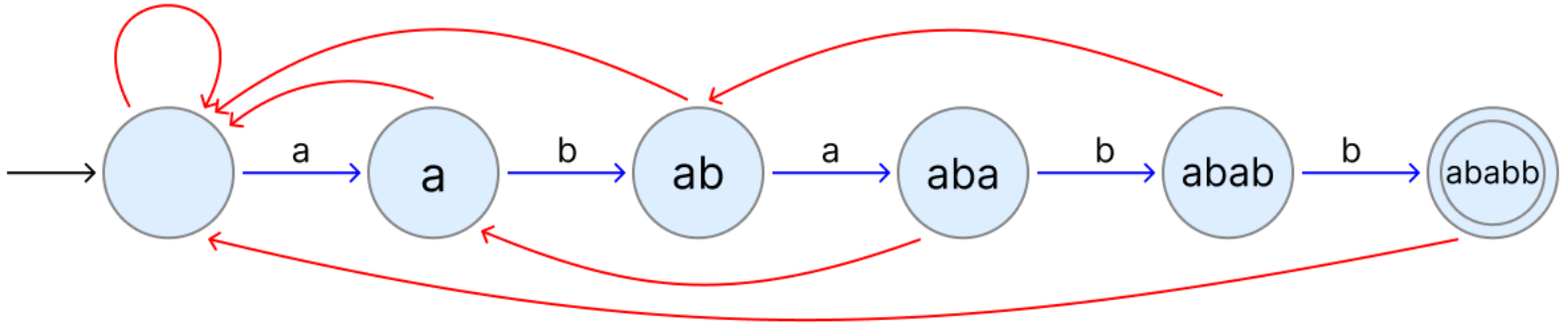


Failure Function (pi)

$s = abaC$, $p = abaD$ 라고 하자.

- aba까지 매칭된 이후 C와 D가 다르기 때문에, Failure function을 이용한다.
- 여기까지 기계는 문자열이 aba로 시작한다는 것을 알 수 있다.
- 다음으로 문자열 ba …가 패턴과 매칭되는지 확인해야 한다. (일치하지 않음)
- 다음으로 a …가 매칭되는지 확인해야 한다.
- $pattern$과 일치하기 때문에 Failure function의 값은 1이다.

기계 만들기 – Failure function



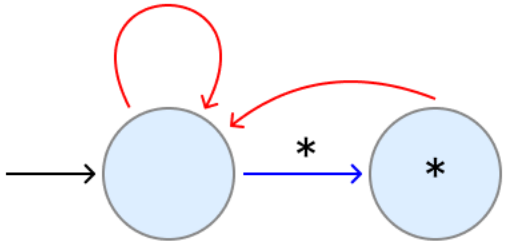
- 매칭에 성공할 경우, 반드시 다음 노드를 가리킨다.
- 노드의 개수는 반드시 $|p| + 1$ 개이다.
- Failure function은 기계를 만들면서 활용한다 (???)

기계 만들기 – failure function

- 수학적 귀납법을 사용해 보자

1. Base case

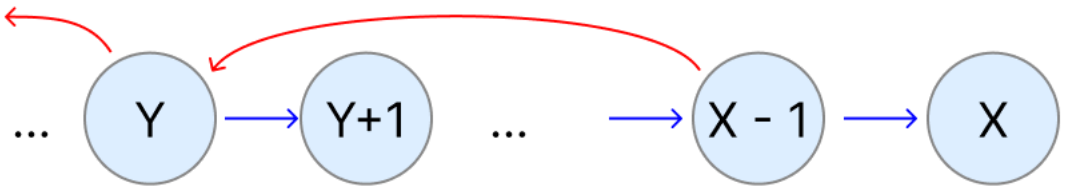
- 첫 두 노드는 반드시 첫 번째 노드가 Failure function의 값이다.



기계 만들기 – failure function

2. Step

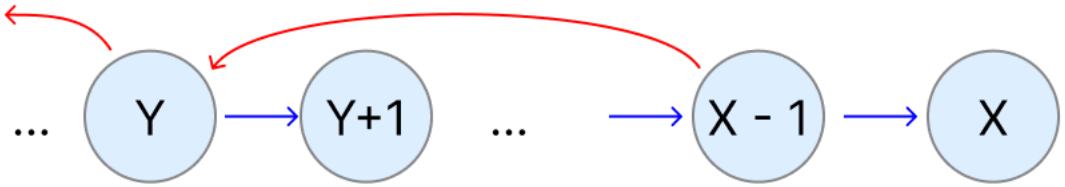
- 임의의 노드 X 전까지 모든 노드의 Failure function이 완성되었다고 가정하자.
- 그렇다면, X 이전 노드의 Failure function이 가리키는 노드를 Y라고 하자.



기계 만들기 – failure function

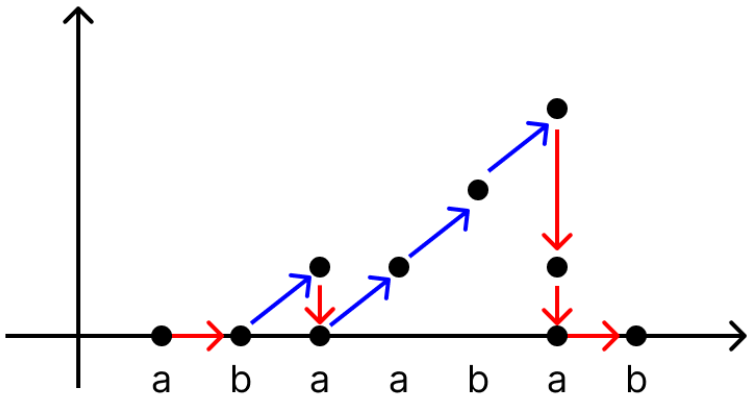
2. Step

- 만약 X의 문자와 $Y+1$의 문자가 같다면, X의 Failure function 값은 $Y+1$이다.
- 다르다면, Y의 Failure function을 따라가서 반복한다.
- 만약 첫 노드를 만난다면 첫 노드를 Failure function 값으로 설정한다.
- Y와 X사이에는 왜 Failure function이 존재할 수 없을까???



Failure function의 성질

- pi 함수는 매칭되는 경우 최대 1 증가할 수 있다. 즉, 최대 $|p|$까지 증가한다.
- 감소되는 경우 1 이상 감소하므로, 최대 $|p|$ 만큼 failure function을 이용한다.
- 즉, 총 $O(|p|)$ 이내에 failure function을 완성할 수 있다.



Failure function의 성질

- $|pattern|$ 에서 (failure function의 마지막 값)을 뺀 값은 문자열의 주기를 의미
  - $abaabaab$의 failure function은 $\{0, 0, 1, 1, 2, 3, 4, 5\}$이다.
  - $|abaabaab| - 5 = 3$이므로,
  - 문자열 $abaabaab$는 aba를 반복하여 만들 수 있다.

- 대부분 문제에서 사용되는 failure function의 성질은

  접두사와 접미사의 길이가 같다는 것을 이용한다!

KMP Algorithm

```
int kmp(const string &str, const string &pattern) {
    …
    int idx = 0;
    for (int i = 0; i < str.length(); i++)
    {
        while (idx && str[i] != pattern[idx])
            idx = pi[idx - 1];
        if (str[i] == pattern[idx])
        {
            if (++idx == pattern.length())
            {
                retval++;
                idx = pi[idx - 1];
            }
        }
    }

    return retval;
}
```

```
vector<int> get_pi(const string &pattern)
{
    …
    int idx = 0;
    for (int i = 1; i < pattern.length(); i++)
    {
        while (idx && pattern[i] != pattern[idx])
            idx = pi[idx - 1];
        if (pattern[i] == pattern[idx])
            pi[i] = ++idx;
    }

    return pi;
}
```

KMP Algorithm 정리

- 대부분 문제에서 사용되는 failure function의 성질은 접두사와 접미사의 길이가 같다는 것을 이용한다.
- $string$과 $pattern$이 같거나 비슷한 문자열에 대해서는 pi함수만 이용하면 되는 경우가 많다.
- failure function의 주기에 대해서는 반드시 알아두자.

찾기 BOJ 1786

- ...Well-Known
- KMP를 짜보고 디버깅 용도로 사용하자

Cubeditor BOJ 1701

- 입력에서 주어진 문자열의 두 번이상 나오는 부분문자열 중에서 가장 긴 길이를 출력한다.

Cubeditor BOJ 1701

- pi 함수의 최댓값으로 부분 문자열 중에 접두사와 일치하는 가장 긴 문자열을 찾을 수 있다.
- KMP를 모든 접미사에 대해 실행한다면, 부분 문자열 중에서 가장 긴 문자열을 찾을 수 있다.

순환 수열 BOJ 12104

문제

두 바이너리 문자열 A와 B가 주어졌을 때, A와 XOR 했을 때, 0이 나오는 B의 순환 순열의 개수를 구하는 프로그램을 작성하시오.

순환 순열이란 순열 $P = P_0, P_1, ..., P_{N-1}$이 있을 때, 왼쪽으로 k번 순환 이동시킨 순열이다. 즉, P를 k번 순환 이동 시키면, P_i -> $P_{i+k \bmod n}$ 이 된다.

입력

첫째 줄에 A, 둘째 줄에 B가 주어진다. A와 B의 길이는 10^5를 넘지 않는 자연수이며, 두 문자열의 길이는 같다.

출력

첫째 줄에 A와 XOR했을 때, 0이 나오는 B의 순환 순열의 개수를 출력한다.

예제 입력 1 복사

```
101
101
```

예제 출력 1 복사

```
1
```

순환 수열 BOJ 12104

- XOR 했을 때 0이 나온다는 것은 같다는 것이다.
- $B + B[:-1]$을 문자열로 KMP 알고리즘을 활용한다.

라디오 전송 BOJ 3356

문제

라디오 방송국은 메시지를 여러 청취자에게 전송한다. 모든 청취자가 메시지를 확실히 받게 하기 위해서 메시지를 계속해서 반복 전송한다.

한 청취자가 받은 메시지가 주어진다. 항상 청취자가 받은 메시지의 길이는 방송국에서 보낸 메시지의 길이보다 크거나 같다. 이때, 라디오 방송국에서 보낸 메시지를 구하는 프로그램을 작성하시오.

즉, 입력으로 S가 주어졌을 때, S가 S' + S' + ... + S'의 부분 문자열이 되는 가장 짧은 부분수열 S'를 구하는 프로그램을 작성하시오.

입력

첫째 줄에 S의 길이 L이 주어진다. 둘째 줄에는 길이가 L인 S가 주어진다. 메시지는 알파벳 소문자로만 이루어져 있다. (1 ≤ L ≤ 1,000,000)

출력

첫째 줄에 S'의 길이 L'을 출력한다.

예제 입력 1 복사

```
8
cabcabca
```

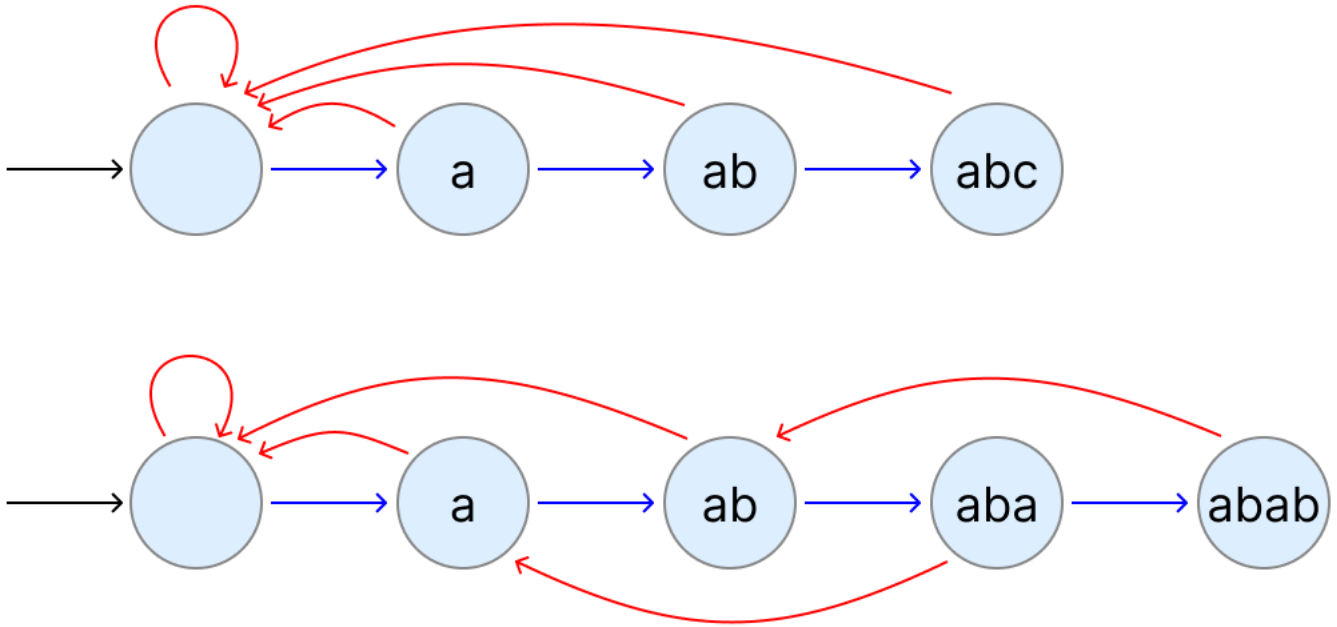
예제 출력 1 복사

```
3
```

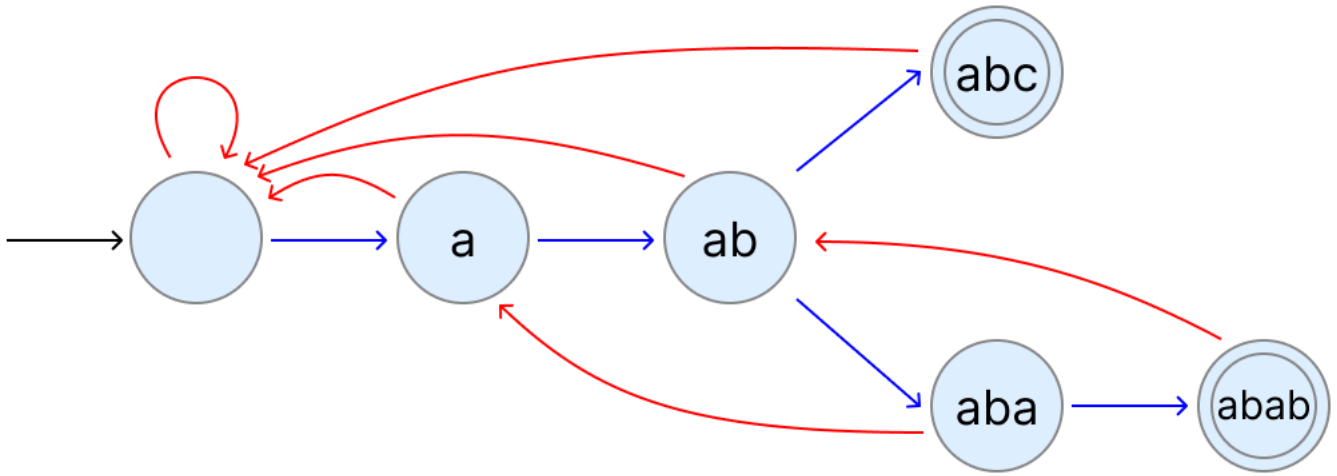
라디오 전송 BOJ 3356

- pi함수의 주기를 이용한다.
- $|p|$- (pi함수의 마지막 값)

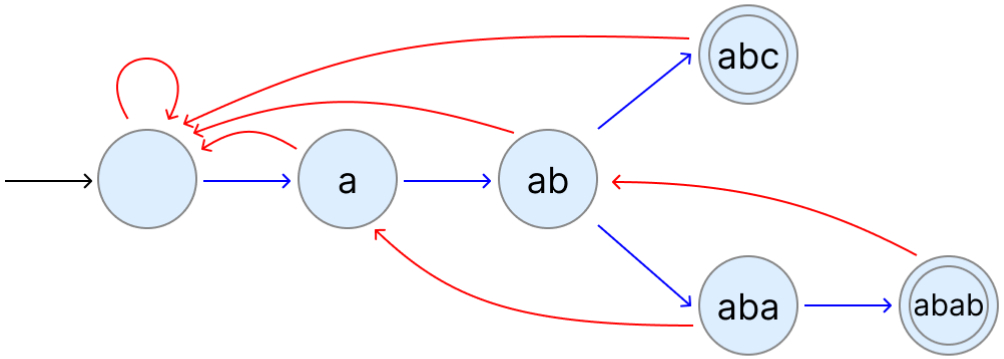
Trie + KMP = ???



Trie + KMP = ???



Aho-Corasick



- 일대다 문자열 패턴 매칭 알고리즘
- 문자열 s, 패턴 $p_1, \ldots, p_n$ 에 대해 시간 복잡도 $O(|s| + \sum |p_i|)$