

#4주차 고급반

접미사 배열

선수내용: 계수 정렬
T. 장형준

목표

- 부분 접미사들을 빠르게 정렬하는 방법을 알아본다.
- 예) "banana"의 접미사 배열을 구하면 다음과 같다.
 - a
 - ana
 - anana
 - banana
 - na
 - nana

문제 상황

- 원소의 개수가 N 인 배열의 비교기반 정렬은 $O(N \lg N)$ 이다.
- 길이가 최대 N 인 문자열의 비교는 $O(N)$ 이다.
- 문자열을 비교해서 접미사 배열을 만들면 $O(N * N \lg N)$.
- $O(N \lg N)$ 으로 줄여보자.

용어 정의

접미사 배열

정의 - 배열과 위치

- 입력으로 주어지는 문자열을 '원본'이라고 하자.
- 부분 접미사들의 배열을 'sa'이라고 하자.
 - Suffix Array의 Acronym(두문자어).

정의 - 접미사의 이름

- i 번 문자를 처음으로 하는 접미사를 '접미사- i '라고 하자.
 - 편의상 0번 문자부터 있다고 하자.
 - 예) "banana"의 접미사-3은 "ana"다.
- 접미사- i 가 있는 sa 의 인덱스를 위치- i 라고 하자.

정의 - 요약

sa	
인덱스	값
0	a
1	anana
2	ana (접미사-3)
3	banana
4	nana
5	na

위치-3 == 2
: 접미사-3이 sa[2]에 있다.

잠깐 휴식

핵심 아이디어

접미사 배열

감 잡고 넘어가주세요.

비교는 이미 끝났다.

- 일단 모든 문자가 다르다고 생각하자.
- 부분 접미사들을 첫번째 문자를 기준으로 정렬했다고 하자.
- i 번째 글자와 j 번째 글자를 비교하고자 하면
 - 접미사- i 와 접미사- j 가 어디에 있는지가 곧 비교결과이다.
 - $(\text{위치-}i > \text{위치-}j)$ 가 i 번째 글자와 j 번째 글자를 비교한 값이다.

비교는 이미 끝났다.

- 일단 모든 문자가 다르다고 생각하자.
- 부분 접미사들을 앞의 n 문자를 기준으로 정렬했다고 하자.
- 원본 $[i, i+n)$, 원본 $[j, j+n)$ 를 비교하고자 하면
 - 접미사- i 와 접미사- j 가 어디에 있는지가 곧 비교결과이다.
 - (위치- $i >$ 위치- j)가 원본 $[i, i+n)$, 원본 $[j, j+n)$ 를 비교한 값이다.

예시

- 원본: "aaabcd"

sa	
인덱스	값
0	aa bc d
1	aaab c d
2	abcd
3	bcd
4	cd
5	d

앞의 두 글자를 기준으로 정렬했다.

모든 문자가 다른 예시는 아니지만 보자.

예시

- 원본: "aaabcd"

sa	
인덱스	값
0	aa bc d
1	aa ab cd
2	abcd
3	bcd
4	cd
5	d

"bc"와 "ab"를 비교하고싶다.
(파란색)

예시

- 원본: "aaabcd"

sa	
인덱스	값
0	aa b cd
1	aa a bcd
2	a bcd
3	b cd
4	cd
5	d

결과는 이미 있다. (주황색)
따라서 다시 비교할 필요 없다.

예시

- 원본: "aaabcd"

sa	
인덱스	값
0	aa b cd
1	aa a bcd
2	a bcd
3	b cd
4	cd
5	d

접1
접0
접2
접3

접미사-0 ("aaabcd")에서
앞의 두 자리를 빼면
접미사-2 ("abcd")이기 때문이다.

접미사-0[2,4) > 접미사-1[2,4)
==
접미사-2[0,2) > 접미사-3[0, 2)
==
위치-2 > 위치-3

아이디어 정리

- '부분 접미사' 라는 특성을 이용해 비교를 줄일 수 있다.
 - '이러한 특성이 있으면 어떤 계산을 안 할 수 있을까?'에 집중.
- 한번의 비교로 범위를 두배 씩 늘어나게 할 수 있다.
 - 한번의 위치 비교
 - 비교된 범위를 두배 씩

휴식

개념 설명

접미사 배열

정의 - 등수

- 위치를 쓰면 같은 글자(문자열)일때 곤란한 상황이 생긴다.
 - 따라서 위치 대신 등수를 사용하기로 한다.
 - 등수- i 는 접미사- i 의 등수이다.
- 등수는 다음과 같이 매겨진다.
 - sa 가 앞의 n 문자를 기준으로 정렬된 상태에서
 - 첫번째 원소의 등수는 0이다.
 - 이전 원소와 앞의 n 글자가 같으면 등수는 같다.
 - 다르면 +1
 - 다음 슬라이드에 예시가 있다.

내용 - 첫 문자 기준 정렬

- 첫번째 문자를 기준으로 어떻게든 정렬한다. ($O(N)$ ~ $O(N \lg N)$)

sa	
등수	값
	a
	anana
	ana
	banana
	nana
	na

내용 - 등수 매기기

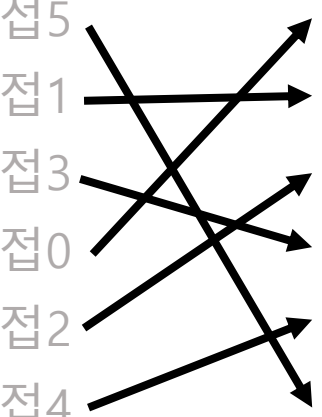
- 첫번째 문자만으로 등수를 매긴다.

sa	
등수	값
0	a
0	anana
0	ana
1	banana
2	nana
2	na

내용 - 현재 상황 정리

sa	
등수	값
0	a
0	anana
0	ana
1	banana
2	nana
2	na

점5 등수-0 == 1
점1 등수-1 == 0
점3 등수-2 == 2
점0 등수-3 == 0
점2 등수-4 == 2
점4 등수-5 == 0

A diagram showing six points (점0 to 점5) on the left and six rank difference calculations (등수-0 to 등수-5) on the right. Arrows connect each point to a calculation: 점5 to 등수-0, 점1 to 등수-1, 점3 to 등수-2, 점0 to 등수-3, 점2 to 등수-4, and 점4 to 등수-5. The arrows for 점1 and 점3 cross each other, and the arrows for 점0 and 점2 cross each other.

등수 배열	
i	등수-i
0	1
1	0
2	2
3	0
4	2
5	0

휴식

정의 - 정렬하기

- 앞에서부터 n 글자를 기준으로 정렬되어있다고 하자.
- 빈 문자열의 등수는 -1 으로 생각하자.
- 계수정렬을 하면 순서가 변하지 않는다.
 - 접미사- i 의 등수를 등수- $(i+n)$ 이라고 생각하고 정렬한다.
 - 접미사- i 의 등수를 등수- i 라고 생각하고 정렬한다.
 - 위 두 과정을 거치면 앞에서부터 $2n$ 글자를 기준으로 정렬되어있을 것이다.
 - 추가 설명이 있으니 이해가 정 안되면 넘기자.

추가 설명1 - 정렬하기

- 두차례의 정렬을 통해 앞에서부터 $2n$ 글자를 기준으로 정렬한다.
 - 앞에서부터 $2n$ 글자를 정렬했다는 것은 다음을 만족하면 된다.
 - $[0, n)$ 의 등수로 정렬되어있다.
 - $[0, n)$ 의 등수가 같다면 $[n, 2n)$ 의 등수로 정렬되어있다.
 - '등수가 같다면', '정렬되어있다'에 집중하자.
 - 따라서, $[n, 2n)$ 의 등수로 정렬한다.
 - '정렬되어있게' 한다.
 - 그 다음, $[0, n)$ 의 등수로 정렬한다.
 - 정렬하되, 같으면 '정렬된 상태를 깨지 않는다'.
 - 따라서 pair($[0, n)$ 의 등수, $[n, 2n)$ 의 등수) 순으로 정렬되어있다.

추가 설명 2 - 정렬하기

- '접미사- i 의 등수를 등수- $(i+n)$ 이라고 생각하고 정렬'의 뜻
 - 이미 앞에서 n 글자를 기준으로 등수가 매겨져있다.
 - 등수- $(i+n)$ 은 접미사의 $[n, 2n)$ 의 등수다.
 - 따라서 등수- $(i+n)$ 이라 생각하면 접미사의 $[n, 2n)$ 순으로 정렬된다.
- '접미사- i 의 등수를 등수- i 라고 생각하고 정렬'의 뜻
 - 등수- i 의 순서대로 다시 정렬하는 과정이다.
 - 등수- i 가 같아도 이전 스텝에서 정렬해봤기 때문에 문제 없다.
 - 계수정렬은 순서가 깨지지 않기 때문이다.

내용 - 새 등수 매기기

- 정렬을 마친 sa 로 새 등수를 매긴다.
- 새등수- i 는 접미사- i 의 새로 매겨진 등수이다.
- 새 등수는 다음과 같이 매긴다.
 - sa 의 인덱스를 따라서
 - 첫번째 원소의 등수는 0
 - 이전 원소와 등수- i , 등수- $(i+n)$ 이 같으면 같은 새등수
 - 아니라면 +1

내용 - 현재 상황 정리

sa	
새등수	값
0	a
1	anana
1	ana
2	banana
3	nana
3	na

접5

접1

접3

접0

접2

접4

등수 배열			
i	등수-i	등수-(i+1)	새등수-i
0	1	0	2 (1, 0)
1	0	2	1 (0, 2)
2	2	0	3 (2, 0)
3	0	2	1 (0, 2)
4	2	0	3 (2, 0)
5	0	-1	0 (0, -1)

내용 - 새 등수 매기기

- 새 등수를 등수로 생각하고 정렬하기로 돌아간다.
- 이렇게 $\lg N$ 번 반복하면 된다.

휴식

기술적인 부분

접미사 배열

등수 함수

- 등수 함수를 만드는 것이 여러모로 편하다.
- `return (i < len) ? 등수[i] : -1`

sa

- sa 배열에는 부분 접미사를 모두 저장할 필요가 없다.
- 접미사-i를 저장하는 대신 i를 저장하자.

ord

- 등수는 보통 `rnk(rank)`로 쓰지만, 여기에서는 `ord(order)`로 쓴다.
- 등수에서 새 등수로 한번에 바뀌는 것이 중요하다.
 - 2차원 배열 `ord[2][N]`으로 만들어서 토글링하자.
 - `ord[현재 스텝%2][i]`

예시코드

- 이렇게 짤 수 있으면 좋습니다.

```
#include <vector>
#include <algorithm>
using namespace std;

#define MAXN 500005

int N, SA[MAXN];
char S[MAXN];

void SuffixArray()
{
    int i, j, k;
    int m = 26; // 처음 알파벳 개수
    vector<int> cnt(max(N, m)+1, 0), x(N+1, 0), y(N+1, 0);
    for (i=1; i<=N; i++) cnt[x[i] = S[i]-'a'+1]++;
    for (i=1; i<=m; i++) cnt[i] += cnt[i-1];
    for (i=N; i; i--) SA[cnt[x[i]]--] = i;
    for (int len=1, p=1; p<N; len<=1, m=p){
        for (p=0, i=N-len; ++i<=N; ) y[++p] = i;
        for (i=1; i<=N; i++) if (SA[i] > len) y[++p] = SA[i]-len;
        for (i=0; i<=m; i++) cnt[i] = 0;
        for (i=1; i<=N; i++) cnt[x[y[i]]]++;
        for (i=1; i<=m; i++) cnt[i] += cnt[i-1];
        for (i=N; i; i--) SA[cnt[x[y[i]]]--] = y[i];
        swap(x, y); p = 1; x[SA[1]] = 1;
        for (i=1; i<N; i++)
            x[SA[i+1]] = SA[i]+len <= N && SA[i+1]+len <= N && y[SA[i]] == y[SA[i+1]] && y[SA[i]+len] == y[SA[i+1]+len] ? p : ++p;
    }
}
```

출처: <https://blog.myungwoo.kr/57> [PS 이야기]

예시코드

- 이렇게 짜면 쉽습니다.

```
#include <bits/stdc++.h>
using namespace std;

const int N = 5e5 + 5;

string str;
int len;
int sa[N];
int ord[N];

int getOrd( int x ) { return x >= len ? -1 : ord[x]; }

int main()
{
    len = str.size();
    sort(sa, sa + len, [](int a, int b){ return str[a] < str[b]; } );
    for( int i = 1; i != sz; i++ ) ord[sa[i]] = ord[sa[i-1]] + str[sa[i-1]] == str[sa[i]];
    for( int cmlen = 1; cmlen <= len; cmlen <= 1 ) {
        sort( sa, sa + len, [](int a, int b){ return getOrd(a + cmlen) < getOrd(b + cmlen); } );
        sort( sa, sa + len, [](int a, int b){ return getOrd(a) < getOrd(b); } );
        ord[sa[0]] = 0;
        for( int i = 1; i != len; i++ ) {
            int conA = getOrd(sa[i]+cmlen) != getOrd(sa[i-1]+cmlen);
            int conB = getOrd(sa[i]) != getOrd(sa[i-1]);
            ord[sa[i]] = ord[sa[i-1]] + (conA || conB);
        }
    }
}
```