JooHyun – Lee (comkiwer)

TS염기서열

Hancom Education Co. Ltd.

Problem

Problem

TS 염기서열은 'A', 'G', 'C', 'T'로 구성된 길이가 3의 배수인 문자열이다.

"GCCCAC", "GGTCCTCCA", "CTA"가 염기서열의 예이다.

또한, 염기서열은 고유한 식별 번호를 가진다.

이러한 염기서열을 등록하고 관리하는 시스템을 개발하고자 한다.

각 기능의 세부 동작은 다음의 API 요구사항 및 예제를 참고하고 각 함수를 구현하라.

아래는 User Code 부분에 작성해야 하는 API 의 설명이다.

Problem

void init()

각 테스트 케이스의 맨 처음에 1회 호출된다.

초기에 등록된 염기서열은 없다.

Problem

TS염기서열

int addSeq(int ID, int Len, char Seq[])

염기서열을 등록하고 등록이 성공한 경우 1을 반환하고 실패한 경우 0을 반환한다.

ID는 염기 서열의 식별 번호이다. Len는 염기서열의 길이이고 Len은 3의 배수이다.

염기서열은 Seq의 배열로 주어지고, Seq는 Len개의 'A', 'G', 'C', 'T' 문자로 이루어진 문자열이다.

등록된 식별 번호는 유일해야 한다.

삭제되지 않은 이미 등록된 식별 번호가 있는 경우 등록에 실패한다.

등록된 염기서열도 유일해야 한다.

삭제되지 않은 똑같은 염기서열이 존재하는 경우 등록에 실패한다.

Parameters

ID : 염기서열의 식별 번호 ($1 \leq ID \leq 1{,}000{,}000{,}000$)

Len: 염기서열의 길이 ($3 \leq Len \leq 60$, 3의 배수)

Seq : 등록하고자하는 염기서열

Returns

등록 성공 여부. 성공한경우 1을 반환하고 실패한 경우 0을 반환하다.

Problem

TS염기서열

int searchSeq(int Len, char prefix[])

등록된 염기서열 중 prefix으로 시작하는 염기서열들을 찾고 그 결과 값을 반환한다.

Len은 prefix의 길이이고 3의 배수이다.
prefix은 Len개의 'A', 'G', 'C', 'T' 문자로 이루어진 문자열이다.

반환하는 결과 값은 다음과 같다.

ⓐ prefix으로 시작하는 염기서열이 2개 이상인 경우 개수를 반환한다.

ⓑ prefix으로 시작하는 염기서열이 1개인 경우는 해당 염기서열의 식별 번호를 반환한다.

ⓒ prefix으로 시작하는 염기서열이 없는 경우 -1을 반환한다.

예를 들어 등록된 염기서열이 (111,"ACGTTC"), (222, "ACG"), (333, "TAACGT")인 경우를 생각하자. 여기서 (a, b)의 a는 식별 번호이고 b는 염기서열이다.

만약, prefix = "ACG"로 주어졌다면,
(111, "ACGTTC")와 (222,"ACG")가 "ACG"으로 시작하는 염기서열이므로 2를 반환한다.

Problem

int searchSeq(int Len, char prefix[]) : *계속*

…

만약, prefix = "TAA"로주어졌다면,
(333, "TAACGT")만 "TAA"으로 시작하는 염기서열이므로 333을 반환한다.

만약, prefix = "AAC"로 주어졌다면 "AAC"로 시작하는 염기서열이 없으므로 -1을 반환한다.

Parameters

Len : 시작 문자열의 길이.($3 \leq Len \leq 60$, 3의 배수)

prefix: 찾고자 하는 염기 서열들의 시작 문자열

Returns

찾은 염기서열이 2개 이상인 경우 그 개수를 반환하고 1개인 경우 식별 번호를 반환하고

해당하는 염기서열이 없는 경우 -1을 반환한다.

Problem

int eraseSeq(int ID)

등록된 염기서열 중 식별 번호가 ID인 염기서열을 삭제하고
삭제에 성공한 경우 1을 반환하고 실패한 경우 0을 반환한다.

삭제된 염기서열은 더 이상 나타나지 않는다.

식별 번호가 ID인 염기서열이없는 경우 삭제에 실패한다.

Parameters

ID : 삭제하고자 하는 염기서열의 식별 번호 ($1 \leq ID \leq 1{,}000{,}000{,}000$)

Returns

삭제 성공 여부. 성공한경우 1을 반환하고 실패한 경우 0을 반환하다.

Problem

int changeBase(int ID, int Pos, char Base)

식별 번호가 ID인 염기서열에서 Pos번째 위치에 있는 문자를 Base로 변경하고
변경에 성공하면 1을 반환하고 실패하면 0을 반환한다.

염기서열의 위치는 0부터 시작한다.

다음과 같은 경우는 변경에 실패한다.

① 식별 번호 ID인 염기 서열이 없는 경우

② 식별 번호 ID인 염기 서열에서 Pos번째 위치가 없는 경우

③ 식별 번호 ID인 염기 서열의 Pos번째 문자가 Base인 경우

④ 식별 번호 ID인 염기 서열의 Pos번째 문자가 Base로 변경될 경우
이미 똑같은 염기서열이 다른 식별 번호로 등록된 경우,

Problem

int changeBase(int ID, int Pos, char Base) : *계속*

…

예를 들면 (111,"ACGTTC")가 있다고 하자.

Pos = 2이고 Base = 'T'이라고 하면 (111, "ACTTTC")로 변경된다.

Parameters

ID : 변경하고자 하는 염기서열의 식별 번호 ($1 \leq ID \leq 1{,}000{,}000{,}000$)

Pos : 변경되어야 할 문자의 위치 ($0 \leq Pos \leq 59$)

Base : 변경하고자 하는 문자 (Base = 'A', 'G', 'C', or 'T')

Returns

염기 값 변경의 성공 여부. 성공한 경우 1을 반환하고 실패한 경우 0을 반환한다

Problem

TS염기서열

[제약사항]

1. 각 테스트 케이스 시작 시 init() 함수가 한 번 호출된다.
2. 염기서열의 식별 번호는 1 이상 1,000,000,000 이하이다.
3. addSeq(), searchSeq() 함수 호출 시 전달되는 mLen는 3의 배수이고
   3 이상 60 이하의 정수임을 보장한다.
4. 각 테스트 케이스에서 addSeq() 함수가 최대 10,000번 호출된다.
5. 각 테스트 케이스에서 searchSeq() 함수가 최대 20,000번 호출된다.
6. 각 테스트 케이스에서 eraseSeq() 함수가 최대 5,000번 호출된다.
7. 각 테스트 케이스에서 changeBase() 함수는 최대 10,000 호출된다.

Problem analysis

Problem analysis : 예제

TS염기서열

Order	Function	Return	Description
1	init()		초기화
2	addSeq(111, 6, "ACGTTC")	1	(111, "ACGTTC")가 등록된다.
3	addSeq(222, 3, "ACG")	1	(222, "ACG")가 등록된다.
4	addSeq(333, 6, "TAACGT")	1	(333, "TAACGT")가 등록된다.

seqID *: ordered*

Seq	ID
ACG	222
ACGTTC	111
TAACGT	333

idSit *: unordered*

ID	Seq :: iterator
222	it::ACG
111	it::ACGTTC
333	it::TAACGT

Problem analysis : 예제

Order	Function	Return	Description
5	addSeq(111, 6, "GTATCA")	0	이미 식별 번호가 111인 염기서열이 등록되어 있어 실패한다.
6	addSeq(444, 6, " TAACGT")	0	이미 "TAACGT"인 염기서열이 등록되어 있어 실패한다.

seqID *: ordered*

Seq	ID
ACG	222
ACGTTC	111
TAACGT	333

idSit *: unordered*

ID	Seq :: iterator
222	it::ACG
111	it::ACGTTC
333	it::TAACGT

Problem analysis : 예제

TS염기서열

Order	Function	Return	Description
7	searchSeq(3, "ACG")	2	"ACG"로 시작하는 염기서열은 (111, "ACGTTC"), (222, "ACG")이다. 2개이므로 2를 반환한다.

seqID *: ordered*

Seq	ID
ACG	222
ACGTTC	111
TAACGT	333

idSit *: unordered*

ID	Seq :: iterator
222	it::ACG
111	it::ACGTTC
333	it::TAACGT

Problem analysis : 예제

TS염기서열

Order	Function	Return	Description
8	searchSeq(3, "TAA")	333	"TAA"로 시작하는 염기서열은 (333, "TAACGT")이다. 1개이므로 식별 번호인 333을 반환한다.

seqID *: ordered*

Seq	ID
ACG	222
ACGTTC	111
TAACGT	333

idSit *: unordered*

ID	Seq :: iterator
222	it::ACG
111	it::ACGTTC
333	it::TAACGT

Problem analysis : 예제

Order	Function	Return	Description
9	searchSeq(3, "AAC")	-1	"ACC"로 시작하는 염기서열은 없다. -1을 반환한다.

seqID *: ordered*

Seq	ID
ACG	222
ACGTTC	111
TAACGT	333

idSit *: unordered*

ID	Seq :: iterator
222	it::ACG
111	it::ACGTTC
333	it::TAACGT

Problem analysis : 예제

TS염기서열

Order	Function	Return	Description
10	changeBase(111, 2, 'T')	1	(111, "ACGTTC")가 (111, "ACTTTC")로 변경된다.

seqID *: ordered*

Seq	ID
ACG	222
ACTTTC	111
TAACGT	333

idSit *: unordered*

ID	Seq :: iterator
222	it::ACG
111	it::ACTTTC
333	it::TAACGT

Problem analysis : 예제

Order	Function	Return	Description
11	changeBase(111, 2, 'T')	0	식별 번호 111인 염기서열의 2번째 문자가 'T'이므로 변경에 실패한다.
12	changeBase(111, 7, 'C')	0	식별 번호 111인 염기서열에서 7번째 문자가 존재하지 않으므로 변경에 실패한다.

seqID *: ordered*

Seq	ID
ACG	222
ACTTTC	111
TAACGT	333

idSit *: unordered*

ID	Seq :: iterator
222	it::ACG
111	it::ACTTTC
333	it::TAACGT

Problem analysis : 예제

TS염기서열

Order	Function	Return	Description
13	addSeq(444, 3, "ATG")	1	(444, "ATG")가 등록된다.

seqID *: ordered*

Seq	ID
ACG	222
ACTTTC	111
ATG	444
TAACGT	333

idSit *: unordered*

ID	Seq :: iterator
222	it::ACG
111	it::ACTTTC
333	it::TAACGT
444	it::ATG

Problem analysis : 예제

TS염기서열

Order	Function	Return	Description
14	addSeq(555, 9, "GTTCAGTGC")	1	(555, "GTTCAGTGC")가 등록된다.

seqID *: ordered*

Seq	ID
ACG	222
ACTTTC	111
ATG	444
GTTCAGTGC	555
TAACGT	333

idSit *: unordered*

ID	Seq :: iterator
222	it::ACG
111	it::ACTTTC
333	it::TAACGT
444	it::ATG
555	it::GTTCAGTGC

Problem analysis : 예제

TS염기서열

Order	Function	Return	Description
15	addSeq(666, 6, "CGCCAG")	1	(666, "CGCCAG")가 등록된다.

seqID *: ordered*

Seq	ID
ACG	222
ACTTTC	111
ATG	444
CGCCAG	666
GTTCAGTGC	555
TAACGT	333

idSit *: unordered*

IDhttffgfy?]	Seq :: iterator
222	it::ACG
111	it::ACTTTC
333	it::TAACGT
444	it::ATG
555	it::GTTCAGTGC
666	it::CGCCAG

Problem analysis : 예제

TS염기서열

Order	Function	Return	Description
16	changeBase(222, 1, 'T')	0	식별 번호 222의 1번째 문자를 'T'로 변경할 경우 이미 "ATG"가 등록되어 있으므로 실패한다.

seqID *: ordered*

Seq	ID
ACG	222
ACTTTC	111
ATG	444
CGCCAG	666
GTTCAGTGC	555
TAACGT	333

idSit *: unordered*

ID	Seq :: iterator
222	it::ACG
111	it::ACTTTC
333	it::TAACGT
444	it::ATG
555	it::GTTCAGTGC
666	it::CGCCAG

Problem analysis : 예제

TS염기서열

Order	Function	Return	Description
17	searchSeq(6, "ACTTTC")	111	"ACTTTC"로 시작하는 염기서열은 (111, "ACTTTC")이다. 1개이므로 식별 번호인 111을 반환한다.

seqID *: ordered*

Seq	ID
ACG	222
ACTTTC	111
ATG	444
CGCCAG	666
GTTCAGTGC	555
TAACGT	333

idSit *: unordered*

ID	Seq :: iterator
222	it::ACG
111	it::ACTTTC
333	it::TAACGT
444	it::ATG
555	it::GTTCAGTGC
666	it::CGCCAG

Problem analysis : 예제

Order	Function	Return	Description
18	eraseSeq(777)	0	식별 번호가 777인 염기서열은 없다.

seqID *: ordered*

Seq	ID
ACG	222
ACTTTC	111
ATG	444
CGCCAG	666
GTTCAGTGC	555
TAACGT	333

idSit *: unordered*

ID	Seq :: iterator
222	it::ACG
111	it::ACTTTC
333	it::TAACGT
444	it::ATG
555	it::GTTCAGTGC
666	it::CGCCAG

Problem analysis : 예제

Order	Function	Return	Description
19	eraseSeq(222)	1	식별 번호가 222인 염기서열을 삭제한다.

seqID *: ordered*

Seq	ID
ACG	*222*
ACTTTC	111
ATG	444
CGCCAG	666
GTTCAGTGC	555
TAACGT	333

idSit *: unordered*

ID	Seq :: iterator
222	*it::ACG*
111	it::ACTTTC
333	it::TAACGT
444	it::ATG
555	it::GTTCAGTGC
666	it::CGCCAG

Problem analysis : 예제

TS염기서열

Order	Function	Return	Description
20	addSeq(777, 3, "ACG")	1	(777, "ACG")가 등록된다.

seqID *: ordered*

Seq	ID
ACTTTC	111
ACG	777
ATG	444
CGCCAG	666
GTTCAGTGC	555
TAACGT	333

idSit *: unordered*

ID	Seq :: iterator
111	it::ACTTTC
333	it::TAACGT
444	it::ATG
555	it::GTTCAGTGC
666	it::CGCCAG
777	it::ACG

Problem analysis : 제약 조건 및 API

- 염기서열은 두 가지 속성을 갖는다.
  1. ID : 1 ~ 10억
  2. Seq : 네 문자(“ACGT”)로 구성된 Len(3~60 이고 3의 배수)길이의 문자열

- 아래 API함수들의 제한된 시간에 실행될 수 있어야 한다.
  - addSeq(ID, Len, Seq[]) : 1만 번 호출
    ID와 Seq[]가 이미 등록되어 있는지 검색될 수 있어야 한다.
  - searchSeq(Len, prefix[]) : 2만 번 호출
    prefix[]를 접두사로 갖는 Seq[] (들)를 찾을 수 있어야 한다.
  - eraseSeq(ID) : 5천 번 호출
    ID를 이용하여 Seq[]를 찾고 ID와 Seq[]를 삭제할 수 있어야 한다.
  - changeBase(ID, Pos, Base) : 1만 번 호출
    ID를 이용하여 Seq[]를 찾아 Seq[]를 변경할 수 있어야 한다.

Problem analysis : 제약 조건 및 API

- API 요구사항을 요약해 보면
  다음 기능들을 적절한 시간 내에 수행할 수 있다면
  문제가 해결 될 것으로 보인다.

  - ID와 Seq[] 검색, 삽입, 삭제를 수행할 수 있다.

  - ID를 이용하여 Seq[]를 검색하고
    Seq[]를 이용하여 ID를 검색할 수 있다.

  - 같은 접두문자열을 갖는 문자열을 검색할 수 있다.

- 어떤 자료구조를 선택할 것인가?

Solution sketch

Solution sketch

- ID : 1 ~ 10억이므로 renumbering hash 또는
  `unordered_map<ID, string> idSeq;` 를 사용할 수 있다.

- 문자열 Seq에 대하여 저장, 탐색, 삭제
  그리고 search(Len, prefix) 함수 호출에
  적절히 답하기 위하여 trie를 이용할 수 있다.

- Seq[]를 관리하는 다른 방법으로는
  Seq[]가 3문자 단위로 주어지므로 첫 3문자를 이용하여
  hash code를 생성한 후(4문자이므로 0 ~ 63)
  이를 인덱스로 Seq[]를 분류해서 담는 방법을 생각할 수 있다.
  이 경우 searchSeq(Len, prefix) 함수가 호출되는 경우 해당 슬롯에 저장된
  Seq[]들을 선형으로 탐색하여 찾을 수 있다.
  `unordered_map<string, ID> seqID[64];`

Solution sketch

- 또 다른 방법으로는 stl container를 이용하는 방법 으로
  다음과 같은 자료구조를 설계할 수 있다.

```
// searchSeq()에 응답하기 위하여 map<string, int> 또는 set<string>
// 을 생각할 수 있는데 eraseSeq()에서 1개 뿐일 때,
// ID반환을 위하여 map<string, int>를 사용한다.
map<string, int> seqID; // <Seq, id>

// <id, Seq iterator>  :ID를 이용하여 문자열을 찾기 위한 container
unordered_map<int, map<string, int>::iterator> idSit;
```

- 이를 이용하여 각 API함수에서 해야 할 일을 정리해 보자.

Solution sketch

void init()

- seqID와 idSeq를 초기화 한다.

int addSeq(int ID, int Len, char Seq[])

- ID 또는 Seq[]가 존재한다면 0을 반환한다.

- 새로운 자료라면 seqID에 추가하고
  추가한 위치를 나타내는 iterator를 이용하여
  idSit에 데이터를 추가한다.
  예) `idSit[ID] = seqID.insert({ Seq, ID }).first;`
- 1을 반환한다.

Solution sketch

int searchSeq(int Len, char prefix[])

- seqID에는 Seq[]들이 아스키코드 오름차순으로 정렬되어 관리된다.
- prefix[]로 시작하는 Seq[]의 첫 위치를 seqID에서 찾는다.
- prefix[]보다 큰 첫 위치를 seqID에서 찾는다.

- prefix[]가 없다면 -1을 반환한다.
- prefix[]가 1개 뿐이라면 ID를 반환한다.
- prefix[]가 2개 이상인 경우 개수를 반환한다.

예)

```
string s = prefix;
auto sit = seqID.lower_bound(s);        // prefix[]로 시작하는 첫 위치 찾기
++s.back();
auto eit = seqID.lower_bound(s);        // prefix[]보다 큰 첫 위치 찾기
int ret = (int)distance(sit, eit);
if (ret == 0) return - 1;               // ⓒ 없는 경우

if (ret == 1) return sit->second;       // ⓑ 1개 뿐인 경우
return ret;                             // ⓐ 2개 이상인 경우
```

Solution sketch

int eraseSeq(int ID)

- idSit를 이용하여 ID에 대응하는 Seq[]의 iterator를 찾는다.
  예)
  ```
  auto it = idSit.find(ID);
  ```
- 존재하지 않는다면 0을 반환한다.
  예)
  ```
  if (it == idSit.end()) return 0;
  ```

- 존재한다면 ID와 Seq[]를 삭제하고 1을 반환한다.
  예)
  ```
  auto sit = it->second;
  idSit.erase(it);
  seqID.erase(sit);
  return 1;
  ```

Solution sketch

int changeBase(int ID, int Pos, char Base)

- idSit를 이용하여 ID에 대응하는 Seq[]의 iterator를 찾는다.
- 존재하지 않는다면 0을 반환한다.
- 해당 문자열s를 구한다. s에 Pos인덱스가 없다면 0을 반환한다.
- s에 Pos가 있는데 s[Pos]==Base라면 0을 반환한다.
- s[Pos]를 Base로 바꾼 t문자열이 존재한다면 0을 반환한다.
- s문자열을 지우고 t문자열을 추가한다. 1을 반환한다.

예)

```
auto it = idSit.find(ID);
if (it == idSit.end()) return 0;        // ⓐ 없는 경우
string s = it->second->first, t = s;
if (s.size() <= Pos) return 0;          // ⓑ Pos인덱스가 없는 경우
if (s[Pos] == Base) return 0;           // ⓒ Pos인덱스 문자가 Base인 경우
t[Pos] = Base;
if (seqID.count(t)) return 0;           // ⓓ t(변경된 문자열)가 존재하는 경우
seqID.erase(it->second);
it->second = seqID.insert({ t, ID }).first;
return 1;
```

Solution sketch

[Summary]

- <key, value> 를 한 쌍으로 하는 데이터를 정렬된 상태를 유지하면서 key를 기준으로 삽입, 삭제, 검색을 O(logN)에 수행하는 stl컨테이너는 map이다.
- <key, value> 를 한 쌍으로 하는 데이터를 정렬은 지원하지 않으며 key를 기준으로 삽입, 삭제, 검색을 평균O(1)에 수행하는 stl컨테이너는 unordered_map이다.
- 문제의 성격에 맞도록 STL 컨테이너를 사용할 수 있다.

[One more step]

- trie등 여러가지 방법으로 본 문제를 해결해 볼 수 있다.

Code example 1

map

unordered_map

lower_bound

Code example 1


```cpp
#include <map>
#include <string>
#include <unordered_map>
using namespace std;

// eraseSeq()에서 1개뿐일때 ID반환을 위하여 map<string, int>를 사용한다.
map<string, int> seqID;
unordered_map<int, map<string, int>::iterator> idSit; // <id, sequence iterator>

void init()
{
    seqID.clear();
    idSit.clear();
}

int addSeq(int ID, int Len, char Seq[])
{
    // ID가 존재하거나 Seq가 존재하는 경우
    if (idSit.count(ID) || seqID.count(Seq)) return 0;

    idSit[ID] = seqID.insert({ Seq, ID }).first;
    return 1;
}
```

Code example 1


```
int searchSeq(int Len, char prefix[])
{
    string s = prefix;
    auto sit = seqID.lower_bound(s);
    ++s.back();
    auto eit = seqID.lower_bound(s);
    int ret = (int)distance(sit, eit);
    if (ret == 0) return - 1;   // ⓒ 없는 경우

    if (ret == 1) return sit->second;     // ⓑ 1개 뿐인 경우
    return ret;                           // ⓐ 2개 이상인 경우
}

int eraseSeq(int ID)
{
    auto it = idSit.find(ID);
    if (it == idSit.end()) return 0;
    auto sit = it->second;
    idSit.erase(it);
    seqID.erase(sit);
    return 1;
}
```

Code example 1


```
int changeBase(int ID, int Pos, char Base)
{
    auto it = idSit.find(ID);
    if (it == idSit.end()) return 0;       // ⓐ 없는 경우
    string s = it->second->first, t = s;
    if (s.size() <= Pos) return 0;         // ⓑ Pos인덱스가 없는 경우
    if (s[Pos] == Base) return 0;          // ⓒ Pos인덱스 문자가 Base인 경우
    t[Pos] = Base;
    if (seqID.count(t)) return 0;          // ⓓ t(변경된 문자열)가 존재하는 경우
    seqID.erase(it->second);
    it->second = seqID.insert({ t, ID }).first;
    return 1;
}
```

Code example 2

unordered_map

directed addressing hash

linear search

Code example 2


```cpp
#include <map>
#include <string>
#include <unordered_map>
using namespace std;

const int LM = 64;

// eraseSeq()에서 1개뿐일때 ID반환을 위하여 <string, int>를 사용한다.
unordered_map<string, int> seqID[LM];

// <id, sequence iterator>
unordered_map<int, unordered_map<string, int>::iterator> idSit;

int getCode(const char*s) {
    int code = 0;
    for (int i = 0; i < 3; ++i) {
        int k = s[i] != 'A' ? s[i] != 'C' ? s[i] != 'G' ? 3 : 2 : 1 : 0;
        code = code * 4 + k;
    }
    return code;
}

void init()
{
    for (int i = 0; i < 64; ++i) seqID[i].clear();
    idSit.clear();
}
```

Code example 2


```
int addSeq(int ID, int Len, char Seq[])
{
    // ID가 존재하거나 Seq가 존재하는 경우
    int hidx = getCode(Seq);
    if (idSit.count(ID) || seqID[hidx].count(Seq)) return 0;

    idSit[ID] = seqID[hidx].insert({ Seq, ID }).first;
    return 1;
}

int searchSeq(int Len, char prefix[])
{
    string s = prefix;
    int ret = -1, cnt = 0;
    int hidx = getCode(prefix);
    for (auto&seq : seqID[hidx]) {
        string t = seq.first.substr(0, Len);
        if (s == t) ret = seq.second, cnt++;
    }
    if (cnt == 0) return -1;       // ⓒ 없는 경우
    if (cnt == 1) return ret;      // ⓑ 1개 뿐인 경우
    return cnt;                    // ⓐ 2개 이상인 경우
}
```

Code example 2


```
int eraseSeq(int ID){
    auto it = idSit.find(ID);
    if (it == idSit.end()) return 0;
    auto sit = it->second;
    int hidx = getCode(sit->first.c_str());
    idSit.erase(it);
    seqID[hidx].erase(sit);
    return 1;
}

int changeBase(int ID, int Pos, char Base){
    auto it = idSit.find(ID);
    if (it == idSit.end()) return 0;            // ⓐ 없는 경우
    string s = it->second->first, t = s;
    if (s.size() <= Pos) return 0;              // ⓑ Pos인덱스가 없는 경우
    if (s[Pos] == Base) return 0;               // ⓒ Pos인덱스 문자가 Base인 경우
    t[Pos] = Base;

    int tidx = getCode(t.c_str());
    if (seqID[tidx].count(t)) return 0;         // ⓓ t(변경된 문자열)가 존재하는 경우

    int sidx = getCode(s.c_str());
    seqID[sidx].erase(it->second);
    it->second = seqID[tidx].insert({ t, ID }).first;
    return 1;
}
```

Code example 3

trie

unordered_map

Code example 3

TS염기서열

```
#ifndef _CRT_SECURE_NO_WARNINGS
#define _CRT_SECURE_NO_WARNINGS
#endif

#include <cstring>
#include <string>
#include <unordered_map>
using namespace std;

const int LM = 30005;
const int BLM = 30000 * 60 * 4;


int ridCnt;
struct Data {
    int ID;
    string dna;
}A[LM];

int getIdx(char c) {
    return c != 'A' ? c != 'C' ? c != 'G' ? 3 : 2 : 1 : 0;
}
```

Code example 3


```
struct Trie {
    int bcnt;
    struct Node {
        int rid;       // renumbering id
        int cnt;       // prefix count
        int child[4]; // child node num;
    }buf[BLM];
    void init() { buf[bcnt = 0] = { 0 }; }
    int find(int par) {
        if (buf[par].rid) return A[buf[par].rid].ID;
        for (int i = 0; i < 4; ++i)if(buf[par].child[i]) {
            return find(buf[par].child[i]);
        }
        return -1;
    }
    int searchSeq(char*s, int par = 0) {
        for (int i = 0; s[i]; ++i) {
            int idx = getIdx(s[i]);
            if (buf[par].child[idx] == 0) return -1;
            par = buf[par].child[idx];
        }
        if (buf[par].cnt > 1) return buf[par].cnt;
        return find(par);
    }
```

Code example 3


```
    void insert(char*s, int par = 0) {
        for (int i = 0; s[i]; ++i) {
            buf[par].cnt++;
            int idx = getIdx(s[i]);
            if (buf[par].child[idx] == 0) {
                buf[++bcnt] = {0};
                buf[par].child[idx] = bcnt;
            }
            par = buf[par].child[idx];
        }
        buf[par].rid = ridCnt;
        buf[par].cnt++;
    }
    int erase(const char*s, int par = 0) {
        buf[par].cnt--;
        if (*s == 0) {
            buf[par].rid = 0;
            return buf[par].cnt;
        }
        int idx = getIdx(*s);
        int res = erase(s + 1, buf[par].child[idx]);
        if (res == 0) buf[par].child[idx] = 0;
        return buf[par].cnt;
    }
}trie;
```

Code example 3


```
unordered_map<int, int> ids;
unordered_map<string, int> hmap;

void init()
{
    trie.init();
    ridCnt = 0;
    ids.clear(), hmap.clear();
}

int addSeq(int ID, int Len, char Seq[])
{
    if (ids.count(ID) || hmap.count(Seq)) return 0;
    ids[ID] = ++ridCnt;
    hmap[Seq] = ridCnt;
    A[ridCnt] = {ID, Seq};
    trie.insert(Seq);
    return 1;
}

int searchSeq(int Len, char prefix[])
{
    return trie.searchSeq(prefix);
}
```

Code example 3

TS염기서열

```
int eraseSeq(int ID)
{
    if (ids.count(ID) == 0) return 0;
    int rid = ids[ID];
    ids.erase(ID);
    hmap.erase(A[rid].dna);
    trie.erase(A[rid].dna.c_str());
    return 1;
}

int changeBase(int ID, int Pos, char Base)
{
    if (ids.count(ID) == 0) return 0;
    int rid = ids[ID];
    string from = A[rid].dna, to = from;
    if (to.size() < Pos || to[Pos] == Base) return 0;
    to[Pos] = Base;
    if (hmap.count(to)) return 0;
    eraseSeq(ID);
    char s[61];
    strcpy(s, to.c_str());
    addSeq(ID, (int)to.size(), s);
    return 1;
}
```

Thank you.