1. 뉴스 클러스터링 (좀 까다로울 듯…)

여러 언론사에서 쏟아지는 뉴스, 특히 속보성 뉴스를 보면 비슷비슷한 제목의 기사가 많아 정작 필요한 기사를 찾기가 어렵다. Daum 뉴스의 개발 업무를 맡게 된 신입사원 튜브는 사용자들이 편리하게 다양한 뉴스를 찾아볼 수 있도록 문제점을 개선하는 업무를 맡게 되었다.

개발의 방향을 잡기 위해 튜브는 우선 최근 화제가 되고 있는 “카카오 신입 개발자 공채” 관련 기사를 검색해보았다.

카카오 첫 공채..’블라인드’ 방식 채용

카카오, 합병 후 첫 공채.. 블라인드 전형으로 개발자 채용

카카오, 블라인드 전형으로 신입 개발자 공채

카카오 공채, 신입 개발자 코딩 능력만 본다

카카오, 신입 공채.. “코딩 실력만 본다”

카카오 “코딩 능력만으로 2018 신입 개발자 뽑는다”

기사의 제목을 기준으로 “블라인드 전형”에 주목하는 기사와 “코딩 테스트”에 주목하는 기사로 나뉘는 걸 발견했다. 튜브는 이들을 각각 묶어서 보여주면 카카오 공채 관련 기사를 찾아보는 사용자에게 유용할 듯싶었다.

유사한 기사를 묶는 기준을 정하기 위해서 논문과 자료를 조사하던 튜브는 “자카드 유사도”라는 방법을 찾아냈다.

자카드 유사도는 집합 간의 유사도를 검사하는 여러 방법 중의 하나로 알려져 있다. 두 집합 A, B 사이의 자카드 유사도 J(A, B)는 두 집합의 교집합 크기를 두 집합의 합집합 크기로 나눈 값으로 정의된다.

예를 들어 집합 A = {1, 2, 3}, 집합 B = {2, 3, 4}라고 할 때, 교집합 A ∩ B = {2, 3}, 합집합 A ∪ B = {1, 2, 3, 4}이 되므로, 집합 A, B 사이의 자카드 유사도 J(A, B) = 2/4 = 0.5가 된다. 집합 A와 집합 B가 모두 공집합일 경우에는 나눗셈이 정의되지 않으니 따로 J(A, B) = 1로 정의한다.

자카드 유사도는 원소의 중복을 허용하는 다중집합에 대해서 확장할 수 있다. 다중집합 A는 원소 “1”을 3개 가지고 있고, 다중집합 B는 원소 “1”을 5개 가지고 있다고 하자. 이 다중집합의 교집합 A ∩ B는 원소 “1”을 min(3, 5)인 3개, 합집합 A ∪ B는 원소 “1”을 max(3, 5)인 5개 가지게 된다. 다중집합 A = {1, 1, 2, 2, 3}, 다중집합 B = {1, 2, 2, 4, 5}라고 하면, 교집합 A ∩ B = {1, 2, 2}, 합집합 A ∪ B = {1, 1, 2, 2, 3, 4, 5}가 되므로, 자카드 유사도 J(A, B) = 3/7, 약 0.42가 된다.

이를 이용하여 문자열 사이의 유사도를 계산하는데 이용할 수 있다. 문자열 “FRANCE”와 “FRENCH”가 주어졌을 때, 이를 두 글자씩 끊어서 다중집합을 만들 수 있다. 각각 {FR, RA, AN, NC, CE}, {FR, RE, EN, NC, CH}가 되며, 교집합은 {FR, NC}, 합집합은 {FR, RA, AN, NC, CE, RE, EN, CH}가 되므로, 두 문자열 사이의 자카드 유사도 J("FRANCE", "FRENCH") = 2/8 = 0.25가 된다.

입력 형식

입력으로는 str1과 str2의 두 문자열이 들어온다. 각 문자열의 길이는 2 이상, 1,000 이하이다.

입력으로 들어온 문자열은 두 글자씩 끊어서 다중집합의 원소로 만든다. 이때 영문자로 된 글자 쌍만 유효하고, 기타 공백이나 숫자, 특수 문자가 들어있는 경우는 그 글자 쌍을 버린다. 예를 들어 “ab+”가 입력으로 들어오면, “ab”만 다중집합의 원소로 삼고, “b+”는 버린다.

다중집합 원소 사이를 비교할 때, 대문자와 소문자의 차이는 무시한다. “AB”와 “Ab”, “ab”는 같은 원소로 취급한다.

출력 형식

입력으로 들어온 두 문자열의 자카드 유사도를 출력한다. 유사도 값은 0에서 1 사이의 실수이므로, 이를 다루기 쉽도록 65536을 곱한 후에 소수점 아래를 버리고 정수부만 출력한다.

예제 입출력

str1 str2 answer

FRANCE french 16384

handshake shake hands 65536

aa1+aa2 AAAA12 43690

E=M\*C^2 e=m\*c^2 65536

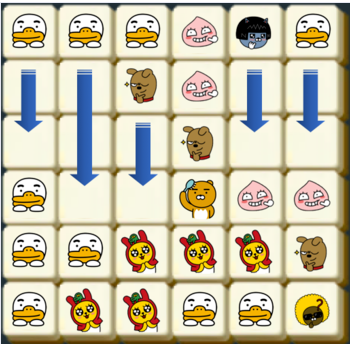
1. 프렌즈 4블록 (나도 못풀겠는데…)

블라인드 공채를 통과한 신입 사원 라이언은 신규 게임 개발 업무를 맡게 되었다. 이번에 출시할 게임 제목은 “프렌즈4블록”.

같은 모양의 카카오프렌즈 블록이 2×2 형태로 4개가 붙어있을 경우 사라지면서 점수를 얻는 게임이다.



만약 판이 위와 같이 주어질 경우, 라이언이 2×2로 배치된 7개 블록과 콘이 2×2로 배치된 4개 블록이 지워진다. 같은 블록은 여러 2×2에 포함될 수 있으며, 지워지는 조건에 만족하는 2×2 모양이 여러 개 있다면 한꺼번에 지워진다.



블록이 지워진 후에 위에 있는 블록이 아래로 떨어져 빈 공간을 채우게 된다.



만약 빈 공간을 채운 후에 다시 2×2 형태로 같은 모양의 블록이 모이면 다시 지워지고 떨어지고를 반복하게 된다.



위 초기 배치를 문자로 표시하면 아래와 같다.

TTTANT

RRFACC

RRRFCC

TRRRAA

TTMMMF

TMMTTJ

각 문자는 라이언(R), 무지(M), 어피치(A), 프로도(F), 네오(N), 튜브(T), 제이지(J), 콘(C)을 의미한다

입력으로 블록의 첫 배치가 주어졌을 때, 지워지는 블록은 모두 몇 개인지 판단하는 프로그램을 제작하라.

입력 형식

입력으로 판의 높이 m, 폭 n과 판의 배치 정보 board가 들어온다.

2 ≦ n, m ≦ 30

board는 길이 n인 문자열 m개의 배열로 주어진다. 블록을 나타내는 문자는 대문자 A에서 Z가 사용된다.

출력 형식

입력으로 주어진 판 정보를 가지고 몇 개의 블록이 지워질지 출력하라.

입출력 예제

m n board answer

4 5 [“CCBDE”, “AAADE”, “AAABF”, “CCBBF”] 14

6 6 [“TTTANT”, “RRFACC”, “RRRFCC”, “TRRRAA”, “TTMMMF”, “TMMTTJ”] 15

예제에 대한 설명

입출력 예제 1의 경우, 첫 번째에는 A 블록 6개가 지워지고, 두 번째에는 B 블록 4개와 C 블록 4개가 지워져, 모두 14개의 블록이 지워진다.

입출력 예제 2는 본문 설명에 있는 그림을 옮긴 것이다. 11개와 4개의 블록이 차례로 지워지며, 모두 15개의 블록이 지워진다.

1. 추석 트래픽

이번 추석에도 시스템 장애가 없는 명절을 보내고 싶은 어피치는 서버를 증설해야 할지 고민이다. 장애 대비용 서버 증설 여부를 결정하기 위해 작년 추석 기간인 9월 15일 로그 데이터를 분석한 후 초당 최대 처리량을 계산해보기로 했다. 초당 최대 처리량은 요청의 응답 완료 여부에 관계없이 임의 시간부터 1초(=1,000밀리초)간 처리하는 요청의 최대 개수를 의미한다.

입력 형식

solution 함수에 전달되는 lines 배열은 N(1 ≦ N ≦ 2,000)개의 로그 문자열로 되어 있으며, 각 로그 문자열마다 요청에 대한 응답완료시간 S와 처리시간 T가 공백으로 구분되어 있다.

응답완료시간 S는 작년 추석인 2016년 9월 15일만 포함하여 고정 길이 2016-09-15 hh:mm:ss.sss 형식으로 되어 있다.

처리시간 T는 0.1s, 0.312s, 2s 와 같이 최대 소수점 셋째 자리까지 기록하며 뒤에는 초 단위를 의미하는 s로 끝난다.

예를 들어, 로그 문자열 2016-09-15 03:10:33.020 0.011s은 “2016년 9월 15일 오전 3시 10분 33.010초”부터 “2016년 9월 15일 오전 3시 10분 33.020초”까지 “0.011초” 동안 처리된 요청을 의미한다. (처리시간은 시작시간과 끝시간을 포함)

서버에는 타임아웃이 3초로 적용되어 있기 때문에 처리시간은 0.001 ≦ T ≦ 3.000이다.

lines 배열은 응답완료시간 S를 기준으로 오름차순 정렬되어 있다.

출력 형식

solution 함수에서는 로그 데이터 lines 배열에 대해 초당 최대 처리량을 리턴한다.

입출력 예제

예제 1

입력: [ “2016-09-15 01:00:04.001 2.0s”, “2016-09-15 01:00:07.000 2s” ]

출력: 1

예제 2

입력: [ “2016-09-15 01:00:04.002 2.0s”, “2016-09-15 01:00:07.000 2s” ]

출력: 2

설명: 처리시간은 시작시간과 끝시간을 포함하므로 첫 번째 로그는 01:00:02.003 ~ 01:00:04.002에서 2초 동안 처리되었으며, 두 번째 로그는 01:00:05.001 ~ 01:00:07.000에서 2초 동안 처리된다. 따라서, 첫 번째 로그가 끝나는 시점과 두 번째 로그가 시작하는 시점의 구간인 01:00:04.002 ~ 01:00:05.001 1초 동안 최대 2개가 된다.

예제 3

입력: [ “2016-09-15 20:59:57.421 0.351s”, “2016-09-15 20:59:58.233 1.181s”, “2016-09-15 20:59:58.299 0.8s”, “2016-09-15 20:59:58.688 1.041s”, “2016-09-15 20:59:59.591 1.412s”, “2016-09-15 21:00:00.464 1.466s”, “2016-09-15 21:00:00.741 1.581s”, “2016-09-15 21:00:00.748 2.31s”, “2016-09-15 21:00:00.966 0.381s”, “2016-09-15 21:00:02.066 2.62s” ]

출력: 7

설명: 아래 타임라인 그림에서 빨간색으로 표시된 1초 각 구간의 처리량을 구해보면 (1)은 4개, (2)는 7개, (3)는 2개임을 알 수 있다. 따라서 초당 최대 처리량은 7이 되며, 동일한 최대 처리량을 갖는 1초 구간은 여러 개 존재할 수 있으므로 이 문제에서는 구간이 아닌 개수만 출력한다.

