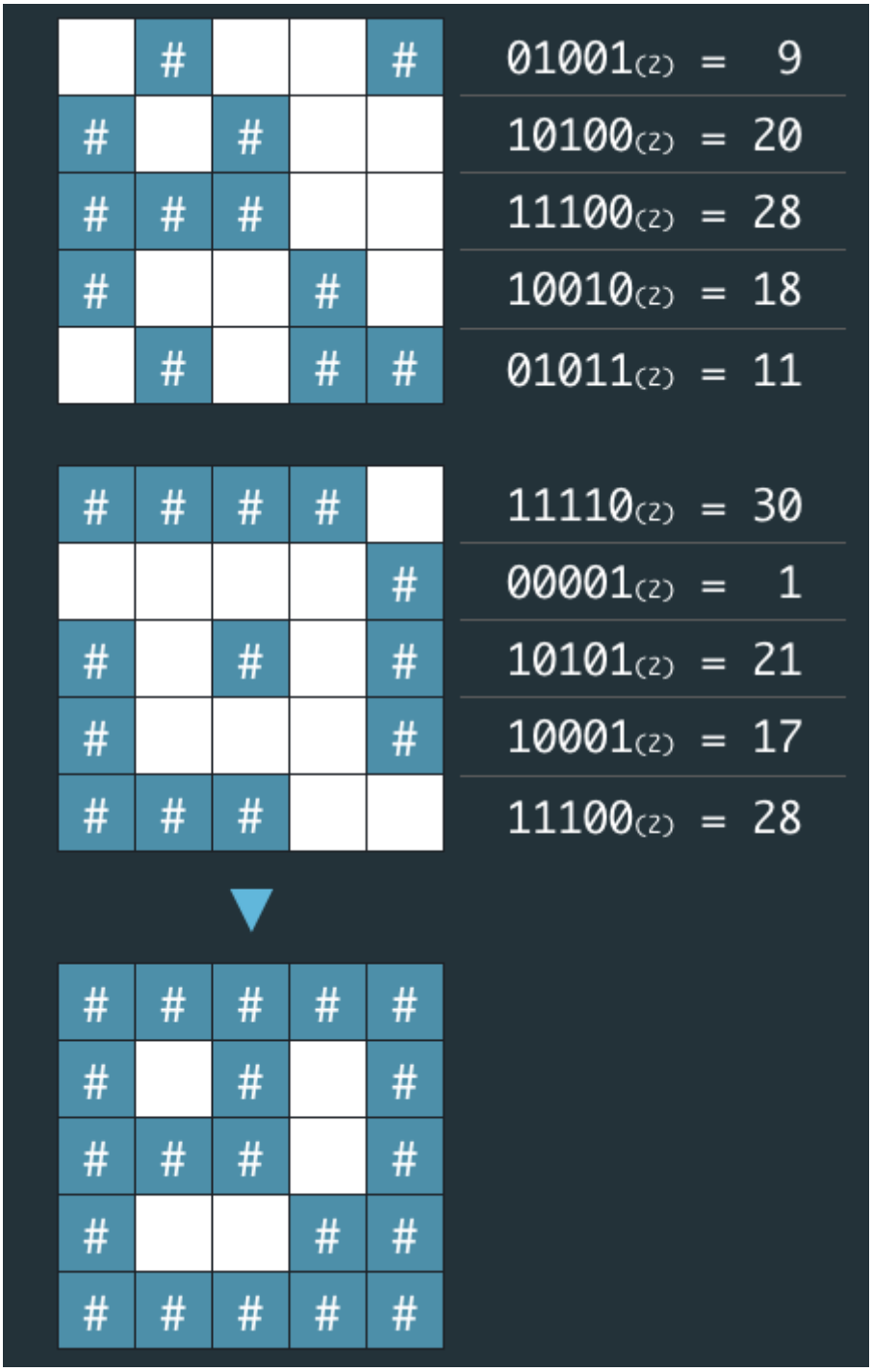
카카오 신입 공채 1차 코딩 테스트 문제 해설

문제 설명

1. 비밀 지도(난이도: 하)

네오는 평소 프로도가 비상금을 숨겨놓는 장소를 알려줄 비밀지도를 손에 넣었다. 그런데 이 비밀지도는 숫자로 암호화되어 있어 위치를 확인하기 위해서는 암호를 해독해야 한다. 다행히 지도 암호를 해독할 방법을 적어놓은 메모도 함께 발견했다.

1. 지도는 한 변의 길이가 `n`인 정사각형 배열 형태로, 각 칸은 "공백"(" ") 또는 "벽"("#") 두 종류로 이루어져 있다.
2. 전체 지도는 두 장의 지도를 겹쳐서 얻을 수 있다. 각각 "지도 1"과 "지도 2"라고 하자. 지도 1 또는 지도 2 중 어느 하나라도 벽인 부분은 전체 지도에서도 벽이다. 지도 1과 지도 2에서 모두 공백인 부분은 전체 지도에서도 공백이다.
3. "지도 1"과 "지도 2"는 각각 정수 배열로 암호화되어 있다.
4. 암호화된 배열은 지도의 각 가로줄에서 벽 부분을 `1`, 공백 부분을 `0`으로 부호화했을 때 얻어지는 이진수에 해당하는 값의 배열이다.



네오가 프로도의 비상금을 손에 넣을 수 있도록, 비밀지도의 암호를 해독하는 작업을 도와줄 프로그램을 작성하라.

입력 형식

입력으로 지도의 한 변 크기 `n` 과 2개의 정수 배열 `arr1`, `arr2`가 들어온다.

- 1 ≦ `n` ≦ 16
- `arr1`, `arr2`는 길이 `n`인 정수 배열로 주어진다.

- 정수 배열의 각 원소 `x`를 이진수로 변환했을 때의 길이는 `n` 이하이다. 즉, 0 ≦ `x` ≦ 2^n - 1을 만족한다.

출력 형식

원래의 비밀지도를 해독하여 `"#"`, `공백`으로 구성된 문자열 배열로 출력하라.

입출력 예제

매개변수	값
n	5
arr1	[9, 20, 28, 18, 11]
arr2	[30, 1, 21, 17, 28]
출력	`["#####","# # #", "### #", "# ##", "#####"]`

매개변수	값
n	6
arr1	[46, 33, 33 ,22, 31, 50]
arr2	[27 ,56, 19, 14, 14, 10]
출력	`["######", "### #", "## ##", " #### ", " #####", "### # "]`

문제 해설

이 문제는 비트 연산Bitwise Operation을 묻는 문제입니다. 이미 문제 예시에 2진수로 처리하는 힌트가 포함되어 있고, 둘 중 하나가 1일 경우에 벽 `#`이 생기기 때문에 OR로 처리하면 간단히 풀 수 있습니다. 아주 쉬운 문제였던 만큼 `if else`로 풀이한 분들도 많이 발견되었는데요. 정답으로는 간주되지만 이 문제는 비트 연산을 잘 다룰 수 있는지를 묻고자 하는 의도였던 만큼 앞으로 이런 유형의 문제를 풀 때는 비트 연산을 꼭 기억하시기 바랍니다.

이 문제의 정답률은 81.78%입니다. 첫 번째 문제이고 가장 쉬운 문제였던 만큼 많은 분들이 잘 풀어주셨습니다.

2. 다트 게임(난이도: 하)

카카오톡에 뜬 네 번째 별! 심심할 땐? 카카오톡 게임별~

카카오톡 게임별의 하반기 신규 서비스로 다트 게임을 출시하기로 했다. 다트 게임은 다트판에 다트를 세 차례 던져 그 점수의 합계로 실력을 겨루는 게임으로, 모두가 간단히 즐길 수 있다.
갓 입사한 무지는 코딩 실력을 인정받아 게임의 핵심 부분인 점수 계산 로직을 맡게 되었다. 다트 게임의 점수 계산 로직은 아래와 같다.

1. 다트 게임은 총 3번의 기회로 구성된다.
2. 각 기회마다 얻을 수 있는 점수는 0점에서 10점까지이다.
3. 점수와 함께 Single(S), Double(D), Triple(T) 영역이 존재하고 각 영역 당첨 시 점수에서 1제곱, 2제곱, 3제곱 (점수^1 , 점수^2 , 점수^3)으로 계산된다.
4. 옵션으로 스타상(*) , 아차상(#)이 존재하며 스타상(*) 당첨 시 해당 점수와 바로 전에 얻은 점수를 각 2배로 만든다. 아차상(#) 당첨 시 해당 점수는 마이너스된다.

5. 스타상(*)은 첫 번째 기회에서도 나올 수 있다. 이 경우 첫 번째 스타상(*)의 점수만 2배가 된다. (예제 4번 참고)
6. 스타상(*)의 효과는 다른 스타상(*)의 효과와 중첩될 수 있다. 이 경우 중첩된 스타상(*) 점수는 4배가 된다. (예제 4번 참고)
7. 스타상(*)의 효과는 아차상(#)의 효과와 중첩될 수 있다. 이 경우 중첩된 아차상(#)의 점수는 -2배가 된다. (예제 5번 참고)
8. Single(S), Double(D), Triple(T)은 점수마다 하나씩 존재한다.
9. 스타상(*), 아차상(#)은 점수마다 둘 중 하나만 존재할 수 있으며, 존재하지 않을 수도 있다.

0~10의 정수와 문자 `S, D, T, *, #`로 구성된 문자열이 입력될 시 총점수를 반환하는 함수를 작성하라.

입력 형식

"점수|보너스|[옵션]"으로 이루어진 문자열 3세트.
예) `1S2D*3T`

- 점수는 0에서 10 사이의 정수이다.
- 보너스는 S, D, T 중 하나이다.
- 옵션은 `*`이나 `#` 중 하나이며, 없을 수도 있다.

출력 형식

3번의 기회에서 얻은 점수 합계에 해당하는 정수값을 출력한다.
예) 37

입출력 예제

예제	dartResult	answer	설명
1	`1S2D*3T`	37	1^1 * 2 + 2^2 * 2 + 3^3
2	`1D2S#10S`	9	1^2 + 2^1 * (-1) + 10^1
3	`1D2S0T`	3	1^2 + 2^1 + 0^3

예제	dartResult	answer	설명
4	1S*2T*3S	23	1^1 * 2 * 2 + 2^3 * 2 + 3^1
5	1D#2S*3S	5	1^2 * (-1) * 2 + 2^1 * 2 + 3^1
6	1T2D3D#	-4	1^3 + 2^2 + 3^2 * (-1)
7	1D2S3T*	59	1^2 + 2^1 * 2 + 3^3 * 2

문제 해설

문자열 처리String Manipulation를 묻는 문제입니다. 앞에서부터 한 글자씩 잘라서 처리할 수 있고, 또는 간단한 컴파일러를 만들듯이 토큰화Tokenizing와 의미 분석Semantic Analysis을 통해 어렵지 않게 계산할 수 있습니다.

점수 중에는 한 자리뿐만 아니라 두 자리인 10점도 포함되어 있기 때문에 한 글자씩 잘라서 처리할때는 그 부분에 유의해야겠네요. 토큰화로 처리할 때는 정규식을 사용하면 비교적 쉽게 잘라낼 수 있습니다. S, D, T는 각각 원점수, 제곱, 세제곱으로 처리하고 스타상은 두 배로 계산하면 됩니다. 참, 아차상은 마이너스 점수라는 점에 유의하세요.

이 문제의 정답률은 73.47%입니다. 앞서 비밀지도 보다는 낮지만 그래도 많은 분들이 잘 풀어주셨습니다.

3. 캐시(난이도: 하)

지도개발팀에서 근무하는 제이지는 지도에서 도시 이름을 검색하면 해당 도시와 관련된 맛집 게시물들을 데이터베이스에서 읽어 보여주는 서비스를 개발하고 있다.
이 프로그램의 테스팅 업무를 담당하고 있는 어피치는 서비스를 오픈하기 전 각 로직에 대한 성능 측정을 수행하였는데, 제이지가 작성한 부분 중 데이터베이스에서 게시물을 가져오는 부분의 실행시간이 너무 오래 걸린다는 것을 알게 되었다.
어피치는 제이지에게 해당 로직을 개선하라고 닦달하기 시작하였고, 제이지

는 DB 캐시를 적용하여 성능 개선을 시도하고 있지만 캐시 크기를 얼마로 해야 효율적인지 몰라 난감한 상황이다.

어피치에게 시달리는 제이지를 도와, DB 캐시를 적용할 때 캐시 크기에 따른 실행시간 측정 프로그램을 작성하시오.

입력 형식

- 캐시 크기(`cacheSize`)와 도시이름 배열(`cities`)을 입력받는다.
- `cacheSize`는 정수이며, 범위는 0 ≦ `cacheSize` ≦ 30 이다.
- `cities`는 도시 이름으로 이뤄진 문자열 배열로, 최대 도시 수는 100,000개이다.
- 각 도시 이름은 공백, 숫자, 특수문자 등이 없는 영문자로 구성되며, 대소문자 구분을 하지 않는다. 도시 이름은 최대 20자로 이루어져 있다.

출력 형식

- 입력된 도시이름 배열을 순서대로 처리할 때, "총 실행시간"을 출력한다.

조건

- 캐시 교체 알고리즘은 `LRU`(Least Recently Used)를 사용한다.
- `cache hit`일 경우 실행시간은 `1`이다.
- `cache miss`일 경우 실행시간은 `5`이다.

입출력 예제

캐시크기(cacheSize)	도시이름(cities)	실행시간
3	["Jeju", "Pangyo", "Seoul", "NewYork", "LA", "Jeju", "Pangyo", "Seoul", "NewYork", "LA"]	50
3	["Jeju", "Pangyo", "Seoul", "Jeju", "Pangyo", "Seoul", "Jeju", "Pangyo", "Seoul"]	21

캐시크기(cacheSize)	도시이름(cities)	실행시간
2	[“Jeju”, “Pangyo”, “Seoul”, “NewYork”, “LA”, “SanFrancisco”, “Seoul”, “Rome”, “Paris”, “Jeju”, “NewYork”, “Rome”]	60
5	[“Jeju”, “Pangyo”, “Seoul”, “NewYork”, “LA”, “SanFrancisco”, “Seoul”, “Rome”, “Paris”, “Jeju”, “NewYork”, “Rome”]	52
2	[“Jeju”, “Pangyo”, “NewYork”, “newyork”]	16
0	[“Jeju”, “Pangyo”, “Seoul”, “NewYork”, “LA”]	25

문제 해설

여기서부터 문제가 좀 어려워졌던 거 같습니다. 정답률이 많이 낮은데요. 이 문제는 '조건'에도 나와있지만 LRU 캐시 교체 알고리즘을 구현하는 문제이고, 이미 잘 알고 있다면 또는 검색해봤다면 잘 구현된 LRU 알고리즘 코드는 많이 찾을 수 있습니다.

단, 이 문제에는 입출력 예제에 캐시 사이즈 0이 포함되어 있습니다. 공개된 대부분의 LRU 구현 코드는 0일 때의 비정상적인 상황은 가정하지 않고 있기 때문에 생각 없이 그냥 가져와 붙인다면 에러가 나서 많이 고생했을 거 같네요. 하지만 사이즈 0을 처리하는 예외 처리 자체는 어렵지 않게 구현할 수 있으므로 입출력 예제가 왜 자꾸 틀리는지를 유심히 살펴봤다면 쉽게 풀 수 있는 문제입니다.

아울러 검색해서 가져온 코드는 반드시 사용 가능한지 라이센스를 확인하고, 가져올때는 꼭 출처를 명시해야 한다는 점 잊지 마세요.

이 문제의 정답률은 45.26%입니다.

4. 셔틀버스(난이도: 중)

카카오에서는 무료 셔틀버스를 운행하기 때문에 판교역에서 편하게 사무실로 올 수 있다. 카카오의 직원은 서로를 '크루'라고 부르는데, 아침마다 많은 크루들이 이 셔틀을 이용하여 출근한다.

이 문제에서는 편의를 위해 셔틀은 다음과 같은 규칙으로 운행한다고 가정하자.

- 셔틀은 `09:00`부터 총 `n`회 `t`분 간격으로 역에 도착하며, 하나의 셔틀에는 최대 `m`명의 승객이 탈 수 있다.
- 셔틀은 도착했을 때 도착한 순간에 대기열에 선 크루까지 포함해서 대기 순서대로 태우고 바로 출발한다. 예를 들어 `09:00`에 도착한 셔틀은 자리가 있다면 `09:00`에 줄을 선 크루도 탈 수 있다.

일찍 나와서 셔틀을 기다리는 것이 귀찮았던 콘은, 일주일간의 집요한 관찰 끝에 어떤 크루가 몇 시에 셔틀 대기열에 도착하는지 알아냈다. 콘이 셔틀을 타고 사무실로 갈 수 있는 도착 시각 중 제일 늦은 시각을 구하여라.

단, 콘은 게으르기 때문에 같은 시각에 도착한 크루 중 대기열에서 제일 뒤에 선다. 또한, 모든 크루는 잠을 자야 하므로 `23:59`에 집에 돌아간다. 따라서 어떤 크루도 다음날 셔틀을 타는 일은 없다.

입력 형식

셔틀 운행 횟수 `n`, 셔틀 운행 간격 `t`, 한 셔틀에 탈 수 있는 최대 크루 수 `m`, 크루가 대기열에 도착하는 시각을 모은 배열 `timetable`이 입력으로 주어진다.

- $0 < n \leqq 10$
- $0 < t \leqq 60$
- $0 < m \leqq 45$
- `timetable`은 최소 길이 1이고 최대 길이 2000인 배열로, 하루 동안 크루가 대기열에 도착하는 시각이 `HH:MM` 형식으로 이루어져 있다.
- 크루의 도착 시각 `HH:MM`은 `00:01`에서 `23:59` 사이이다.

출력 형식

콘이 무사히 셔틀을 타고 사무실로 갈 수 있는 제일 늦은 도착 시각을 출력한다. 도착 시각은 `HH:MM` 형식이며, `00:00`에서 `23:59` 사이의 값이 될 수 있다.

입출력 예제

n	t	m	timetable	answer
1	1	5	[“08:00”, “08:01”, “08:02”, “08:03”]	“09:00”
2	10	2	[“09:10”, “09:09”, “08:00”]	“09:09”
2	1	2	[“09:00”, “09:00”, “09:00”, “09:00”]	“08:59”
1	1	5	[“00:01”, “00:01”, “00:01”, “00:01”, “00:01”]	“00:00”
1	1	1	[“23:59”]	“09:00”
10	60	45	[“23:59”,”23:59”, “23:59”, “23:59”, “23:59”, “23:59”, “23:59”, “23:59”, “23:59”, “23:59”, “23:59”, “23:59”, “23:59”, “23:59”, “23:59”, “23:59”, “23:59”, “23:59”, “23:59”, “23:59”]	“18:00”

문제 해설

쉬워 보이는데 어려운 문제가 바로 이 문제였던 거 같네요. 당초 난이도를 ‘중’으로 두고 문제를 중간 즈음에 배치하였는데, 시간을 계산하는 부분에서 많은 분들이 어려워하셨던 거 같습니다.

예를 들어 2번 입출력 예제의 경우 `["09:10", "09:09", "08:00"]`인데 이 경우 두 번째 버스는 9:10분에 출발하기 때문에 9:10분에 오면 되지 않느냐 많이들 혼동하셨을 거 같아요. 하지만 9:00에 오는 버스는 8:00에 대기하는 크루 1명만 탑승할 수 있고, 따라서 9:10 버스에는 남아 있는 두 명이 모두 타게 됩니다. 따라서 좀 더 이른 9:09에 와야 탑승할 수 있습니다.

전체 계산은 어렵지 않지만 이처럼 정확하게 시간 계산을 해야 하는 부분이 많고 마지막 버스 시간까지 빈틈없이 계산해야 해서 많은 분들이 실수를 한 거 같습니다. 이 문제는 정답률이 두 번째로 낮은 26.79%입니다.

5. 뉴스 클러스터링(난이도: 중)

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 = 0.5`가 된다. 집합 A와 집합 B가 모두 공집합일 경우에는 나눗셈이 정의되지 않으니 따로 `J(A, B) = 1`로 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 ∩ B`는 원소 "1"을 min(3, 5)인 3개, 합집합 `A ∪ B`는 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

str1	str2	answer
E=M*C^2	e=m*c^2	65536

문제 해설

이 문제는 자카드 유사도를 설명해주고 자카드 유사도를 직접 계산하는 프로그램을 작성하는 문제입니다. 자카드 유사도는 실무에서도 유사한 문서를 판별할 때 주로 쓰이는데요, 몰랐더라도 문제에서 자세히 설명해주기 때문에 이해하는데 어려움은 없었을 거 같습니다. 공식은 매우 간단한데요, 교집합을 합집합으로 나눈 수입니다. 다만, 이 값은 0에서 1 사이의 실수가 되는데, 여기서는 이를 다루기 쉽도록 65536을 곱한 후 소수점 아래를 버리고 정수부만 취하도록 합니다.

문제 설명은 원소의 중복을 허용하는 다중집합multiset으로 되어 있는데, 자주 접하는 자료구조가 아니고, 일부 언어에서는 기본으로 제공하는 자료구조가 아니라 어려워하는 분들이 있었습니다. 하지만 다중집합 자료구조를 쓰지 않더라도, 각 원소를 정렬된 배열에 넣은 후 병합 정렬Merge sort에서 배웠던 코드를 응용, 어렵지 않게 합집합과 교집합 함수를 직접 구현할 수도 있습니다.

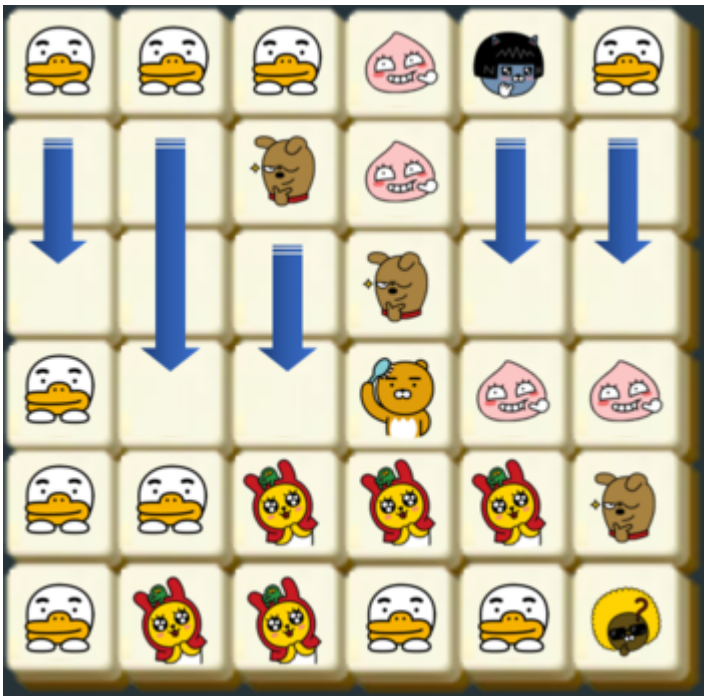
다중집합의 교집합, 합집합만 잘 구해낸다면 이 문제는 어렵지 않게 풀 수 있으며, 다만 집합 A와 B가 모두 공집합일 경우에는 나눗셈이 정의되지 않으므로division by zero 따로 `J(A,B) = 1`로 정의합니다. 즉, 65536을 곱하면 이 경우 `1 * 65536 = 65536`이 정답이 됩니다. 예제 입출력에도 합집합이 공집합인 경우가 포함되어 있으므로 이 경우만 주의한다면 쉽게 풀 수 있는 문제입니다.

이 문제의 정답률은 41.84%입니다.

6. 프렌즈4블록(난이도: 상)

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 `n`과 판의 배치 정보 `board`가 들어온다.
- 2 <= `n`, `m` <= 30
- `board`는 길이 `n`인 문자열 `m`개의 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.

문제 해설

게임 요구 사항을 구현해보는 문제입니다. 같은 모양의 카카오프렌즈 블록이 2x2 형태로 4개가 붙어있을 경우 사라지면서 점수를 얻는 게임인데요. 인접한 모든 블록이 사라지는 실제 게임들과 달리 계산을 쉽게 하기 위해 2x2로 제한하고, 사라진 블록 자리에는 새로운 블록이 채워지지 않습니다. 그럼에도 불구하고 인접한 블록을 모두 스캔해야 하는 문제라 짧지 않은 코드가 필요했을 것 같네요. 이번 시험에서 가장 긴 코드가 필요한 문제였습니다. 자바의 경우 무려 80라인이나 필요했네요. 블록 매트릭스를 생성하여 스캔하고 제거해 나가는 작업을 반복하면서 더 이상 제거되지 않을 때 사라진 블록 자리의 수를 계산하면 됩니다.

이 문제의 정답률은 48.01%입니다.

7. 추석 트래픽(난이도: 상)

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 `s`로 끝난다.
- 예를 들어, 로그 문자열 `2016-09-15 03:10:33.020 0.011s`은 "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

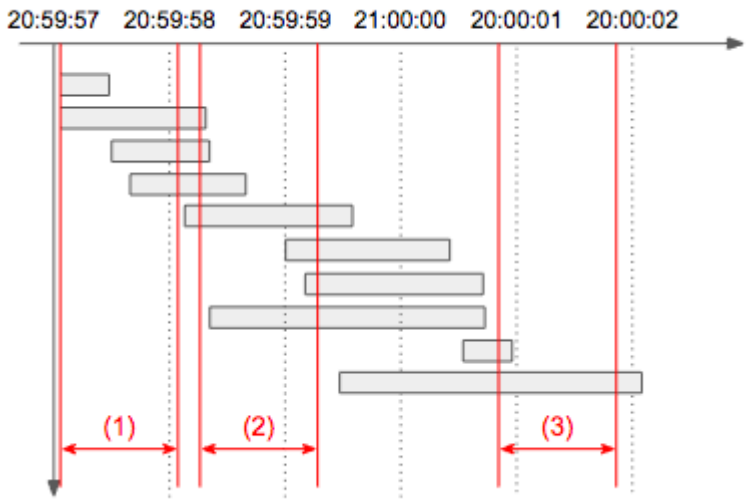
예제 2

- 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 ~ 01:00:04.002`에서 2초 동안 처리되었으며, 두 번째 로그는 `01:00:05.001 ~ 01:00:07.000`에서 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.



문제 해설

이번 테스트의 마지막 문제이고, 가장 어려운 문제입니다. 초당 최대 처리량이 되는 구간 윈도우를 찾아야 하는 문제인데요. 당연히 처음부터 끝까지 스캔하기에는 범위가 너무 크고, 게다가 ms 단위로 되어 있기 때문에 첫 로그 시각부터 마지막 로그 시각까지 1ms씩 증가시키면서 1000ms 단위의 슬라이딩 윈도우로 풀면 `24 * 3600 * 1000 * n * 1000ms` 만큼의 연산이 필요하기 때문에 이렇게는 풀 수가 없습니다.

그렇다고 각 로그의 시작 시각부터 마지막 시각까지 1ms 씩 움직이면 `time(ms) * n^2` 이 되며, time(ms)의 값은 대부분 천 단위 이상이기 때문에 마찬가지로 타임아웃이 발생하여 풀 수가 없습니다. 그런데 자세히 살펴보면 요청량이 변하는 순간은 각 로그의 시작과 끝뿐임을 알 수 있습니다. 따라서, 각 로그 별 2번의 비교 연산만 수행하면 되며 `2 * n^2`, 빅오로 정리하면 `O(n^2)`에 풀 수가 있습니다. 빅오에서 제거된 상수항도 매우 작기 때문에 이 경우 무리 없이 문제를 풀 수 있게 되며 C++ 기준으로 10ms를 넘지 않습니다.

물론, 이 문제는 윈도우를 사용하지 않고도 풀 수 있는 방법이 있습니다. 효율적인 알고리즘을 쓴다면, `O(n log n)`으로 풀 수 있는 방법도 있으니 한 번 고민

해보세요. 이 문제는 가장 어려운 문제였던 만큼 정답률은 가장 낮은 17.99%입니다.