

5조 알고리즘 스터디

박예린

230409 Sun

목차

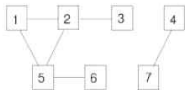
- 백준 2606번 - 바이러스
- 백준 1012번 - 유기농 배추
- 프로그래머스 42576번 - 완주하지 못한 선수
- 프로그래머스 1845번 - 폰켓몬

백준 2606번 - 바이러스

문제

신종 바이러스인 웜 바이러스는 네트워크를 통해 전파된다. 한 컴퓨터가 웜 바이러스에 걸리면 그 컴퓨터와 네트워크 상에서 연결되어 있는 모든 컴퓨터는 웜 바이러스에 걸리게 된다.

예를 들어 7대의 컴퓨터가 <그림 1>과 같이 네트워크 상에서 연결되어 있다고 하자. 1번 컴퓨터가 웜 바이러스에 걸리면 웜 바이러스는 2번과 5번 컴퓨터를 거쳐 3번과 6번 컴퓨터까지 전파되어 2, 3, 5, 6 네 대의 컴퓨터는 웜 바이러스에 걸리게 된다. 하지만 4번과 7번 컴퓨터는 1번 컴퓨터와 네트워크 상에서 연결되어 있지 않기 때문에 영향을 받지 않는다.



< 그림 1 >

어느 날 1번 컴퓨터가 웜 바이러스에 걸렸다. 컴퓨터의 수와 네트워크 상에서 서로 연결되어 있는 정보가 주어질 때, 1번 컴퓨터를 통해 웜 바이러스에 걸리게 되는 컴퓨터의 수를 출력하는 프로그램을 작성하시오.

인접 행렬 vs. 인접 리스트

	정점 간의 간선 여부	공간 복잡도
인접 행렬	정점 v_1, v_2 에 대해 한 번의 접근으로 확인 가능	<p>V개의 노드 표현을 위해 V^2 만큼의 공간이 필요</p> <p>인접 노드를 찾기 위해선 모든 노드를 순회해야 함</p> <p>공간복잡도 : $O(V^2)$</p>
인접 리스트	리스트의 처음부터 하나씩 확인해야 함	<p>V개의 리스트에 간선(E) 만큼 원소가 들어있음</p> <p>인접 노드를 쉽게 찾을 수 있음</p> <p>공간복잡도 : $O(V+E)$</p>

백준 1012번 - 유기농 배추

문제

차세대 영농인 한나는 강원도 고랭지에서 유기농 배추를 재배하기로 하였다. 농약을 쓰지 않고 배추를 재배하려면 배추를 해충으로부터 보호하는 것이 중요하기 때문에, 한나는 해충 방지에 효과적인 배추흰지렁이를 구입하기로 결심한다. 이 지렁이는 배추근처에 서식하며 해충을 잡아 먹음으로써 배추를 보호한다. 특히, 어떤 배추에 배추흰지렁이가 한 마리라도 살고 있으면 이 지렁이는 인접한 다른 배추로 이동할 수 있어, 그 배추들 역시 해충으로부터 보호받을 수 있다. 한 배추의 상하좌우 네 방향에 다른 배추가 위치한 경우에 서로 인접해있는 것이다.

한나가 배추를 재배하는 땅은 고르지 못해서 배추를 군데군데 심어 놓았다. 배추들이 모여있는 곳에는 배추흰지렁이가 한 마리만 있으면 되므로 서로 인접해있는 배추들이 몇 군데에 퍼져있는지 조사하면 총 몇 마리의 지렁이가 필요한지 알 수 있다. 예를 들어 배추밭이 아래와 같이 구성되어 있으면 최소 5마리의 배추흰지렁이가 필요하다. 0은 배추가 심어져 있지 않은 땅이고, 1은 배추가 심어져 있는 땅을 나타낸다.

1	1	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	1	1	1
0	0	0	0	1	0	0	1	1	1

1	1	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	1	1	1
0	0	0	0	1	0	0	1	1	1

B14502_연구소

문제

인체에 치명적인 바이러스를 연구하던 연구소에서 바이러스가 유출되었다. 다행히 바이러스는 아직 퍼지지 않았고, 바이러스의 확산을 막기 위해서 연구소에 벽을 세우려고 한다.

연구소는 크기가 $N \times M$ 인 직사각형으로 나타낼 수 있으며, 직사각형은 1×1 크기의 정사각형으로 나누어져 있다. 연구소는 빈 칸, 벽으로 이루어져 있으며, 벽은 칸 하나를 가득 차지한다.

일부 칸은 바이러스가 존재하며, 이 바이러스는 상하좌우로 인접한 빈 칸으로 모두 퍼져나갈 수 있다. 새로 세울 수 있는 벽의 개수는 3개이며, 꼭 3개를 세워야 한다.

예를 들어, 아래와 같이 연구소가 생긴 경우를 살펴보자.

```
2 0 0 0 1 1 0
0 0 1 0 1 2 0
0 1 1 0 1 0 0
0 1 0 0 0 0 0
0 0 0 0 0 1 1
0 1 0 0 0 0 0
0 1 0 0 0 0 0
```

Python의 2차원 배열 초기화

```
34     '''
35     배추를 심은 배추밭의 가로길이 M( $1 \leq M \leq 50$ ),
36                                     세로길이 N( $1 \leq N \leq 50$ ),
37     배추가 심어져 있는 위치의 개수 K( $1 \leq K \leq 2500$ )
38     그 다음 K줄에는 배추의 위치 X( $0 \leq X \leq M-1$ ), Y( $0 \leq Y \leq N-1$ )
39     '''
40
41     N, M, K = map(int, input().split())
42     graph = [[0]*M for _ in range(N)] # Python의 2차원 배열 초기화
```


Python의 2차원 배열 초기화

List Comprehension

- 파이썬에서 직관적으로 리스트를 생성하는 방법
- 대괄호로 감싸고 내부에서 for문과 if문을 사용해서 조건에 만족하는 것만 리스트로 생성할 수 있다.

Python의 2차원 배열 초기화

만일, 아래와 같이 배열을 초기화 한다면 값을 하나 변경했을 때 의도하지 않은 것까지 변경될 수 있다.

```
# N*M 크기의 2차원 배열
```

```
n = 5
```

```
m = 2
```

```
arr = [[0]*m]*n
```

```
arr[0][0] = 5
```

```
# [[5,0],[5,0],[5,0],[5,0],[5,0]]
```

파이썬은 * 연산자로 초기화할 때 값을 각각 할당하지 않고 얇은 복사로 값을 할당한다.

Python의 Hash

1. 리스트를 쓸 수 없을 때

리스트는 숫자 인덱스를 이용하여 원소에 접근하는데 즉 `list[1]`은 가능하지만 `list['a']`는 불가능합니다. 인덱스 값을 숫자가 아닌 다른 값 '문자열, 튜플'을 사용하려고 할 때 딕셔너리를 사용하면 좋습니다.

2. 빠른 접근 / 탐색이 필요할 때

아래에서 표로 정리해 보여드릴 예정이지만, 딕셔너리 함수의 시간복잡도는 대부분 $O(1)$ 이므로 아주 빠른 자료구조입니다!

3. 집계가 필요할 때

원소의 개수를 세는 문제는 코딩 테스트에서 많이 출제되는 문제입니다. 이때 해시와, `collections` 모듈의 `Counter` 클래스를 사용하면 아주 빠르게 문제를 푸실 수 있을 것입니다.

프로그래머스 42576번 - 완주하지 못한 선수

#1

```
7  def solution(participant, completion):
8      # 참가자 hash dictionary 초기화
9      for p in participant:
10         dict1[hash(p)] = p # key: hash(p), value: p
11
12     for v in dict1.values():
13         if(v in completion):
14             del dict[v] # value로 dictionary 삭제 안 됨.
15
16
17     answer = ''
18     return answer
```

#2

```
main.py × input.txt output.txt
main.py > solution
1 participant = ["leo", "kiki", "eden"]
2 completion = ["eden", "kiki"]
3
4 dict1 = {}
5
6 print(hash("leo"))
7 print(hash("leo"))
8
```

```
main.py input.txt output.txt ×
output.txt
1 -3301271480885639641
2 -3301271480885639641
3
```

[파이썬 | 자료구조] 6. 해시 테이블(Hash Table)

공삼21 2020. 9. 4. 19:19

해시 구조란?

Key와 Value 두 쌍으로 이루어진 데이터이다.

Key를 이용해서 Value를 찾는다. (최선 : $O(1)$ 최악 : $O(N)$)

파이썬에서는 기본적으로 제공되는 딕셔너리 자료형이 해시 테이블과 같은 구조이다.

검색, 저장, 삭제가 잦은 경우 사용하면 속도가 빠르다.

하지만 저장공간이 더 많이 필요하며(넉넉하게 필요하며), 해시 값이 같은 경우 충돌을 해결하는 알고리즘이 필요하고, 최악의 경우 시간복잡도가 $O(N)$ 으로 증가하게 된다.

어떤 Key값에 대해서 숫자(주소값)로 바꿔주는 함수를 해시 함수라고 한다.

해시 테이블이란 해시 함수로 접근하는 데이터 구조를 뜻한다.

파이썬에서는 특정 문자열을 숫자값으로 바꿔주는 `hash()` 함수가 있는데 파이썬3 부터 보안을 이유로 프로그램을 실행할 때마다 값이 달라진다.

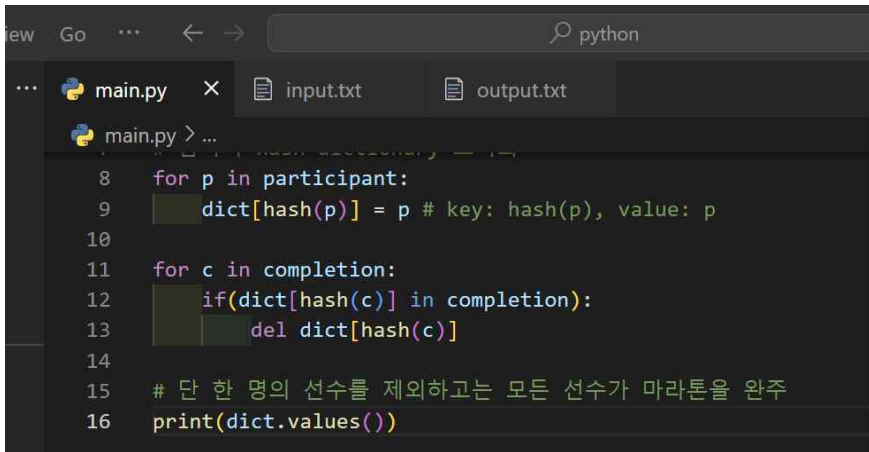
그래서 해시 테이블을 구현할 때 간단하게 해시값을 만드는 방법으로는 각 문자열의 아스키코드 값을 다 더하는 것이다. 그렇지만 이 방법은 "ABC"와 "ACB"의 해시값이 같아서 충돌이 빈번하게 일어나기 때문에

각 자리 아스키코드값을 더할때마다 특정 소수를 곱해서 충돌을 피하게 만든다.

그럼에도 불구하고 결국 충돌을 완전히 피할 수는 없다.

충돌 해결 알고리즘은 크게 2가지로 나눌 수 있다.

#3



The image shows a code editor window with a dark theme. The top bar includes a search icon and the text 'python'. Below the top bar, there are tabs for 'main.py', 'input.txt', and 'output.txt'. The 'main.py' tab is active, showing the following Python code:

```
8   for p in participant:
9       dict[hash(p)] = p # key: hash(p), value: p
10
11  for c in completion:
12      if(dict[hash(c)] in completion):
13          del dict[hash(c)]
14
15  # 단 한 명의 선수를 제외하고는 모든 선수가 마라톤을 완주
16  print(dict.values())
```


#4

```
main.py • input.txt output.txt
main.py > ...
1 participant = ["mislav", "stanko", "mislav", "ana"]
2 completion = ["stanko", "ana", "mislav"]
3
4 dict = {}
5 sum = 0
6
7 # 참가자 hash dictionary 초기화
8 for p in participant:
9     dict[hash(p)] = p # key: hash(p), value: p
10    sum += hash(p)
11
12 for c in completion:
13     if(dict[hash(c)] in completion):
14         sum -= hash(c)
15
16 # 단 한 명의 선수를 제외하고는 모든 선수가 마라톤을 완주
17 print(dict[sum])
```

프로그래머스 1845번 - 폰켓몬



코딩테스트 연습 > 해시 > 폰켓몬

도움말 컴파일 옵션

폰켓몬

dark

light

sublime

vim

emacs

Python3

입출력 예 설명

입출력 예 #1

문제의 예시와 같습니다.

입출력 예 #2

6마리의 폰켓몬이 있으므로, 3마리의 폰켓몬을 골라야 합니다.

가장 많은 종류의 폰켓몬을 고르기 위해서는 3번 폰켓몬 한 마리, 2번 폰켓몬 한 마리, 4번 폰켓몬 한 마리를 고르면 되며, 따라서 3을 return 합니다.

입출력 예 #3

6마리의 폰켓몬이 있으므로, 3마리의 폰켓몬을 골라야 합니다.

가장 많은 종류의 폰켓몬을 고르기 위해서는 3번 폰켓몬 한 마리와 2번 폰켓몬 두 마리를 고르거나, 혹은 3번 폰켓몬 두 마리와 2번 폰켓몬 한 마리를 고르면 됩니다. 따라서 최대 고를 수 있는 폰켓몬 종류의 수는 2 입니다.

solution.py

```
1 def solution(nums):
2     dict = {}
3
4     # 폰켓몬 hash dictionary 초기화
5     for n in nums:
6         if(n not in dict):
7             dict[hash(n)] = n
8
9     answer = len(dict)
10    if(len(nums)/2 < len(dict)):
11        answer = len(nums)/2
```

실행 결과

정확성: 100.0

합계: 100.0 / 100.0

질문하기 (34)

테스트 케이스 추가하기

다른 사람의 풀이

초기화

코드 실행

제출 후 채점하기