[Python트랙] 월말평가3 - 알고리즘 응용

| Background

- ✓ 배열을 응용한 연산 활용
- ✓ 그래프 최적화에 대한 이해와 활용
- ✓ 그래프 탐색

| Goal

- ✓ 2차원 배열에서 합성곱 연산을 구현할 수 있다.
- ✓ 그래프의 최소비용 문제를 해결할 수 있다.
- ✓ DFS와 BFS로 그래프를 탐색할 수 있다.

| 환경 설정

1) Pycharm과 pypy 또는 python을 이용해서 코드를 작성하고 결과를 확인한다.

- 새로운 프로젝트를 생성하지 않고 기존 프로젝트 사용시 부정행위로 간주 함.

- Pypy에서만 동작하는 비표준 문법 사용시 0점 처리됨.

2) 파일 이름 및 제출 방법

- 1, 2번 문제에 대한 소스 파일 이름은 다음과 같이 영문으로 작성한다.

서울 1반 이싸피라면, algo문제번호_반_이름.py 순서로 영문으로 작성

algo1_01_leessafy.py, algo2_01_leessafy.py

- 3번 문제에 대한 답안 파일 이름은 .txt 형식으로 다음과 같이 영문으로 작성한다.

algo3_01_leessafy.txt

- 위 3개의 파일만 지역_반_이름.zip으로 압축하여 제출한다.

서울_1반_이싸피.zip

(탐색기에서 파일 선택 후 오른쪽 클릭 – 압축대상 – Zip 선택)

3) 채점

- 문제별로 부분 점수가 부여된다.

- 주석이 없는 경우, 주석이 코드 내용과 맞지 않는 경우, 지정된 출력 형식을 만족하지 않는 경우 해당 문제는 0점 처리될 수 있다.

- import를 사용한 경우 해당 문제는 0점 처리될 수 있다. (import sys도 예외 없음)

4) 테스트케이스는 부분적으로 제공되며, 전체가 공개되지는 않는다.

5) 각 문제의 배점이 다르므로 표기된 배점을 반드시 확인한다.

- 1번 40점, 2번 35점, 3번 25점

성실과 신뢰로 테스트에 임할 것 (부정 행위시 강력 조치 및 근거가 남음)

※ 소스코드 유사도 판단 프로그램 기준 부정 행위로 판단될 시, 0점 처리 및 학사 기준에 의거 조치 실시 예정

[Python트랙] 월말평가3 – 알고리즘 응용

| 문제1 : 합성곱 Convolution (배점 : 40점)

import 사용 금지

김싸피는 AI를 학습하는 학생이다. 김싸피는 합성곱 연산을 눈으로 확인할 수 있는 프로그램을 작성하려고 한다. 합성곱이란 N*N 크기 입력에 M*M 크기의 필터를 슬라이딩 하며 각 위치에서 **요소별 곱을 합산**해 새로운 출력을 만들어 특징을 추출하는 연산이다.

[예시]

입력 데이터

1	2	3	4	5	6
7	8	9	10	11	12
13	14	15	16	17	18
19	20	21	22	23	24
25	26	27	28	29	30
31	32	33	34	35	36

필터

0	-1	0
-1	5	-1
0	-1	0

- 첫 번째 연산은 필터를 입력의 좌측 상단에 두고 연산을 진행한다.
- 입력데이터와 필터의 연산결과는 연산의 순서대로 출력 행렬에 저장한다.
- $1 * (0) + 2 * (-1) + 3 * 0 + 7 * (-1) + 8 * 5 + 9 * (-1) + 13 * (0) + 14 * (-1) + 15 * 0 = 8$

1	2	3	4	5	6
7	8	9	10	11	12
13	14	15	16	17	18
19	20	21	22	23	24
25	26	27	28	29	30
31	32	33	34	35	36

0	-1	0
-1	5	-1
0	-1	0

8			

[Python트랙] 월말평가3 – 알고리즘 응용

- 두 번째 연산은 필터를 옆으로 한 칸 이동하여 진행한다.
- $2 * (0) + 3 * (-1) + 4 * 0 + 8 * (-1) + 9 * 5 + 10 * (-1) + 14 * (0) + 15 * (-1) + 16 * 0 = 9$

1	2	3	4	5	6
7	8	9	10	11	12
13	14	15	16	17	18
19	20	21	22	23	24
25	26	27	28	29	30
31	32	33	34	35	36

0	-1	0
-1	5	-1
0	-1	0

8	9		

- 필터를 한 칸씩 이동하며 합성곱 연산을 수행하면 마지막 연산 및 결과는 다음과 같다.
- $22 * (0) + 23 * (-1) + 24 * 0 + 28 * (-1) + 29 * 5 + 30 * (-1) + 34 * (0) + 35 * (-1) + 36 * 0 = 29$

1	2	3	4	5	6
7	8	9	10	11	12
13	14	15	16	17	18
19	20	21	22	23	24
25	26	27	28	29	30
31	32	33	34	35	36

0	-1	0
-1	5	-1
0	-1	0

8	9	10	11
14	15	16	17
20	21	22	23
26	27	28	29

- 합성곱 연산의 결과의 크기는 (N – M + 1) * (N – M + 1) 이 된다.

[Python트랙] 월말평가3 – 알고리즘 응용

[입력]

첫 번째 줄에 테스트케이스의 개수 T가 주어진다.

각 테스트 케이스의 첫 번째 줄에 N, M 이 주어진다. 이어서N 줄에 거쳐 N * N 크기의 입력 데이터가 주어지고, 이후 M줄에 M*M 크기의 필터가 주어진다.

(2 ≤ N ≤ 50, 2 ≤ M ≤ 10, M < N)

[출력]

각 테스트케이스마다 '#tc'(tc는 테스트케이스 번호)를 출력하고, 연산의 결과 행렬을 띄어쓰기로 구분하여 출력한다.

[입력 예시]

```
1
6 3
1 2 3 4 5 6
7 8 9 10 11 12
13 14 15 16 17 18
19 20 21 22 23 24
25 26 27 28 29 30
31 32 33 34 35 36
0 -1 0
-1 5 -1
0 -1 0
```

(algo1_sample_in.txt 참고)

[출력 예시]

```
#1
8 9 10 11
14 15 16 17
20 21 22 23
26 27 28 29
```

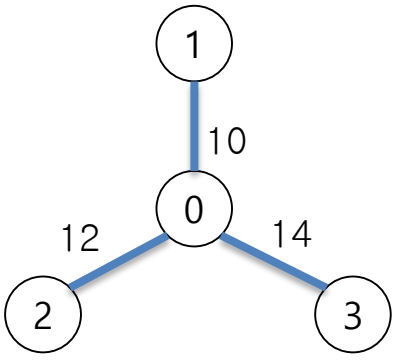
(algo1_sample_out.txt 참고)

[Python트랙] 월말평가3 – 알고리즘 응용

| 문제2 : (배점 : 35점)

import 사용 금지

서기 2742년, 은하 연방은 외곽 성단 개척을 위한 싸피 원정대를 출범했다. 원정대는 행성(또는 궤도 정거장)들 사이에 하이퍼항로(양방향)를 개통해 모든 거점을 서로 오갈 수 있도록 연결해야 한다. 각 후보 항로는 개통에 드는 에너지 크레딧 비용이 다르며, 전체 예산이 빠듯한 원정대는 총 개통 비용의 합이 최소가 되도록 항로를 선택하려 한다. 당신의 임무는 각 성단(테스트케이스)마다 **모든 거점을 한 네트워크로 잇는 데 필요한 최소 총 비용을 계산하는 것**이다.



예를 들어 첫 번째 성단의 경우, 0번에서 3번까지 4개의 거점과 개통 가능한 3개의 항로 후보가 주어지고, 이 경우 3개의 항로를 모두 개통하면 모든 거점을 36의 비용으로 연결할 수 있다.

[입력]

첫 줄에 테스트케이스 개수 T, 각 테스트케이스에 대해 첫 줄에 V E, 다음 E줄에 걸쳐 u v w가 주어진다.

[Python트랙] 월말평가3 – 알고리즘 응용

- V : 마지막 정점 번호 (정점 ID는 0 ... V로 총 V+1개 거점), 2 <=V <= 30
- E : 후보 하이퍼항로(간선)의 개수, V <= E <= V * (V + 1) / 2
- u, v, w : 거점 u와 v를 잇는 양방향 하이퍼항로의 개통 비용 w (정수),

  0 <= u, v, <= V, 1 <= w <= 50

[출력]

각 줄에 #과 테스트케이스 번호, 빈 칸에 이어 정답을 출력한다.

[입력 예시]

```
3
3 3
0 1 10
0 2 12
0 3 14
4 6
3 4 12
1 2 7
2 3 5
0 1 10
0 3 12
0 4 11
5 9
0 2 11
2 3 14
0 1 14
1 3 10
1 2 10
0 5 1
2 5 13
4 5 11
1 4 11
```

(algo1_sample_in.txt 참고)

[출력 예시]

```
#1 36
#2 33
#3 43
```

(algo1_sample_out.txt 참고)

[Python트랙] 월말평가3 – 알고리즘 응용

| 문제3 : 그래프 탐색 (배점 : 25점)

첫 페이지 파일 저장 방법 참고

1. DFS(깊이 우선 탐색)와 BFS(너비 우선 탐색)의 차이를 간단히 설명하시오.

2. 다음 그래프의 1번 정점부터 BFS로 탐색하는 경우, 가능한 정점 방문 순서를 쓰시오.