과목 명: 시스템프로그래밍

담당 교수 명: 소 정 민

<<Assignment 4>>

**서강대학교 컴퓨터공학과**

**[20161631]**

**[임동진]**

목 차

1. 프로그램 개요 3

2. 프로그램 설명 3

2.1 Max counters 문제 설명 3

2.1.1 문제 내용 3

2.1.2 입출력 예시 3

2.2 Common prime divisors 문제 설명 3

2.2.1 문제 내용 3

2.2.2 입출력 예시 4

2.3 Min abs sum of two 문제 설명 4

2.3.1 문제 내용 4

2.3.2 입출력 예시 4

2.4 Max pooling 문제 설명 5

2.4.1 문제 내용 5

2.4.2 입출력 예시 5

3. 모듈 정의 6

3.1 Max counters 문제 풀이 및 알고리즘 설명 6

3.1.1 전체 구현 모습 6

3.1.2 풀이 및 알고리즘 설명 6

3.1.3 사용 변수 및 함수 6

3.2 Common prime divisors 문제 풀이 및 알고리즘 설명 7

3.2.1 전체 구현 모습 7

3.2.1 풀이 및 알고리즘 설명 7

3.2.2 사용 변수 및 함수 8

3.3 Min abs sum of two 문제 풀이 및 알고리즘 설명 8

3.3.1 전체 구현 모습 8

3.3.2 풀이 및 알고리즘 설명 8

3.3.3 사용 변수 및 함수 8

3.4 Max pooling 문제 풀이 및 알고리즘 설명 9

3.4.1 전체 구현 모습 9

3.4.2 풀이 및 알고리즘 설명 10

3.4.3 사용 변수 및 함수 10

4. 전역 변수 정의 10

5. 코드 10

5.1 p1.ipynb 10

5.2 p2.ipynb 14

5.3 p3.ipynb 17

5.4 p4.ipynb 19

# 프로그램 개요

총 네 개의 문제를 Python Programming 으로 해결한다.

Python 3 로 진행하며 Colaboratory 또는 Jupyter Notebook 환경에서 코드를 작성한다.

# 프로그램 설명

## Max counters 문제 설명

### 문제 내용

You are given N counters, initially set to 0, and you have two possible operations on them

Operation 1. increase(X) − counter X is increased by 1,

Operation 2. max counter − all counters are set to the maximum value of any counter.

A non-empty array A of M integers is given. This array represents consecutive operations

if A[K] = X, such that 1 ≤ X ≤ N, then operation K is increase(X),

if A[K] = N + 1 then operation K is max counter.

For example, given integer N = 5 and array A [3, 4, 4, 6, 1, 4, 4]

The values of the counters after each consecutive operation will be:

(0, 0, 1, 0, 0) → (0, 0, 1, 1, 0) → (0, 0, 1, 2, 0) → (2, 2, 2, 2, 2) →

(3, 2, 2, 2, 2) → (3, 2, 2, 3, 2) → (3, 2, 2, 4, 2)

For the above example, your output will be:

3 2 2 4 2

The objective is to calculate the value of every counter after all operations

### 입출력 예시

Input:

5 # N

3 4 4 6 1 4 4 # Array A

Output:

3 2 2 4 2 # Result of counter

## Common prime divisors 문제 설명

### 문제 내용

A prime is a positive integer X that has exactly two distinct divisors: 1 and X. The

first few prime integers are 2, 3, 5, 7, 11 and 13.

A prime D is called a prime divisor of a positive integer P if there exists a positive

integer K such that D \* K = P. For example, 2 and 5 are prime divisors of 20.

You are given two positive integers N and M. The goal is to check whether the

sets of prime divisors of integers N and M are exactly the same.

For example, given:

N = 15 and M = 75, the prime divisors are the same: {3, 5};

N = 10 and M = 30, the prime divisors aren't the same: {2, 5} is not equal to {2, 3, 5};

N = 9 and M = 5, the prime divisors aren't the same: {3} is not equal to {5}.

You pair two elements that have the same index for a given lists A and B.

For example, given:

A = [15, 10, 3]

B = [75, 30, 5]

the result in this input should be 1, because only one pair (15, 75) has the same

set of prime divisors.

You must use the datatype “set” to save the prime divisor for each number.

The result of prime divisor 75 must be {3, 5}.

And check whether each pair of prime divisor set has the same common prime

divisor.

### 입출력 예시

Input:

15 10 3 # Array A

75 30 5 # Array B

Output:

1 # Number of set of prime divisors.

## Min abs sum of two 문제 설명

### 문제 내용

Let A be a non-empty array consisting of N integers.

The abs sum of two for a pair of indices (P, Q) is the absolute value |A[P] +

A[Q]|, for 0 ≤ P ≤ Q < N.

For example, the following array A:

A = [1, 4, -3]

For example, the following array A:

A = [1, 4, -3]

has pairs of indices (0, 0), (0, 1), (0, 2), (1, 1), (1, 2), (2, 2).

The abs sum of two for the pair (0, 0) is A[0] + A[0] = |1 + 1| = 2.

The abs sum of two for the pair (0, 1) is A[0] + A[1] = |1 + 4| = 5.

The abs sum of two for the pair (0, 2) is A[0] + A[2] = |1 + (−3)| = 2.

The abs sum of two for the pair (1, 1) is A[1] + A[1] = |4 + 4| = 8.

The abs sum of two for the pair (1, 2) is A[1] + A[2] = |4 + (−3)| = 1.

The abs sum of two for the pair (2, 2) is A[2] + A[2] = |(−3) + (−3)| = 6.

### 입출력 예시

Input:

-8 4 5 -10 3 # Array A

Output:

3 # Result of min abs sum of two

# |(−8) + 5| = 3

## Max pooling 문제 설명

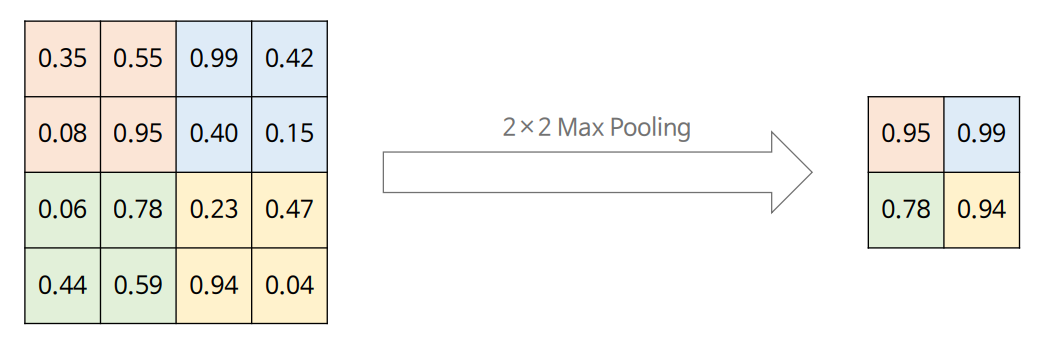
### 문제 내용

You want to compress for a given two-dimensional numpy array.

So, You would like to reduce the size of numpy array by half, with the maximum

number for each 2×2 size area as the representative value.

This method is called max pooling.



For example the following two-dimensional numpy array A:

A = [[0.35 0.55 0.99 0.42]

[0.08 0.95 0.40 0.15]

[0.06 0.78 0.23 0.47]

[0.44 0.59 0.94 0.04]]

the result in this input will be:

B = [[0.95 0.99]

[0.78 0. 94]]

You must use the numpy and datatype “numpy.ndarray” to generate array A,

and also use numpy for processing time.

### 입출력 예시

Input:

4 # size of the square matrix

Output:

[[0.35 0.55 0.99 0.42] # Randomly generated numpy array

[0.08 0.95 0.40 0.15] # np.random.random((size, size))

[0.06 0.78 0.23 0.47]

[0.44 0.59 0.94 0.04]]

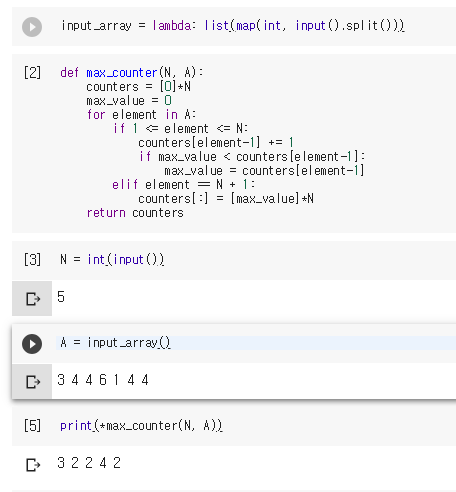
[[0.95 0.99] # Result of max pooling about array

[0.78 0.94]]

# 모듈 정의

## Max counters 문제 풀이 및 알고리즘 설명

### 전체 구현 모습



### 풀이 및 알고리즘 설명

전체 알고리즘 설명:

문제의 조건을 그대로 코드로 적용하였음. 사용자로부터 입력을 받고 max\_counter를 수행한다. Max\_counter 함수는 리스트를 기준으로 반복문을 돌리면서 1<=리스트의 값<=N이면 increase operation을 수행하고 리스트의 값이 N+1이면 Max Counter operation 을 수행함. 반복이 끝나고 함수의 결과로 나온 최종적으로 나온 결과를 출력함.

### 사용 변수 및 함수

* input\_array : 배열을 사용자로부터 입력 받는 익명 함수 변수
* max\_counter : N과 A에 대한 max\_counter를 구하여 리턴하는 함수
* N : counter 의 길이
* A : array

## Common prime divisors 문제 풀이 및 알고리즘 설명

### 전체 구현 모습



### 풀이 및 알고리즘 설명

전체 알고리즘 설명:

사용자로부터 Array 두개(A, B)를 입력 받고, zip 함수를 통해 A와 B를 같은 index 끼리 묶어준다.

그러면 [(15, 75), (10, 30). (3, 5)] 와 같은 형태의 리스트(AB)가 나오게 된다.

Filter 함수를 이용하여 리스트(AB)의 요소들 중에 Prime divisors 가 서로 같은 경우만을 찾아서 결과 리스트(res)에 저장한다.(위에 예시로 든 경우에 res는 [(15, 75)]가 된다.)

최종적으로 결과 리스트(res)의 길이를 출력한다.

Prime divisors 구하는 알고리즘 설명:

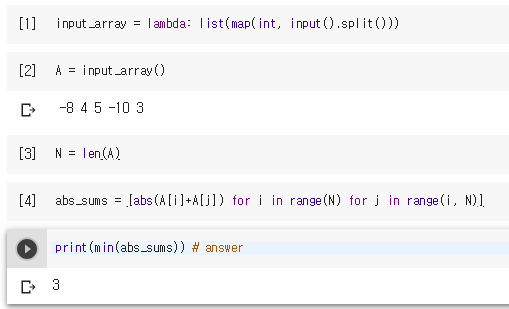
소수는 1과 자기자신만으로 나뉘어져야하므로, 1~N 을 순차적으로 검사하면서 N을 나눌수있는 수가 오직 두개만 있다면 Prime divisors 집합에 추가한다.

### 사용 변수 및 함수

* input\_array : 사용자로부터 배열을 입력받는 익명 함수 변수
* prime\_divisors : prime\_divisors를 구하는 함수
* is\_same\_prime\_divisors\_pair : pair 가 서로 같은 prime\_divisors를 가졌는지 여부를 리턴하는 익명함수 변수
* A, B : 입력받은 배열
* AB : A, B를 합친 배열
* Res : 같은 prime\_divisors 를 가진 pair 만을 갖는 배열

## Min abs sum of two 문제 풀이 및 알고리즘 설명

### 전체 구현 모습

.

### 풀이 및 알고리즘 설명

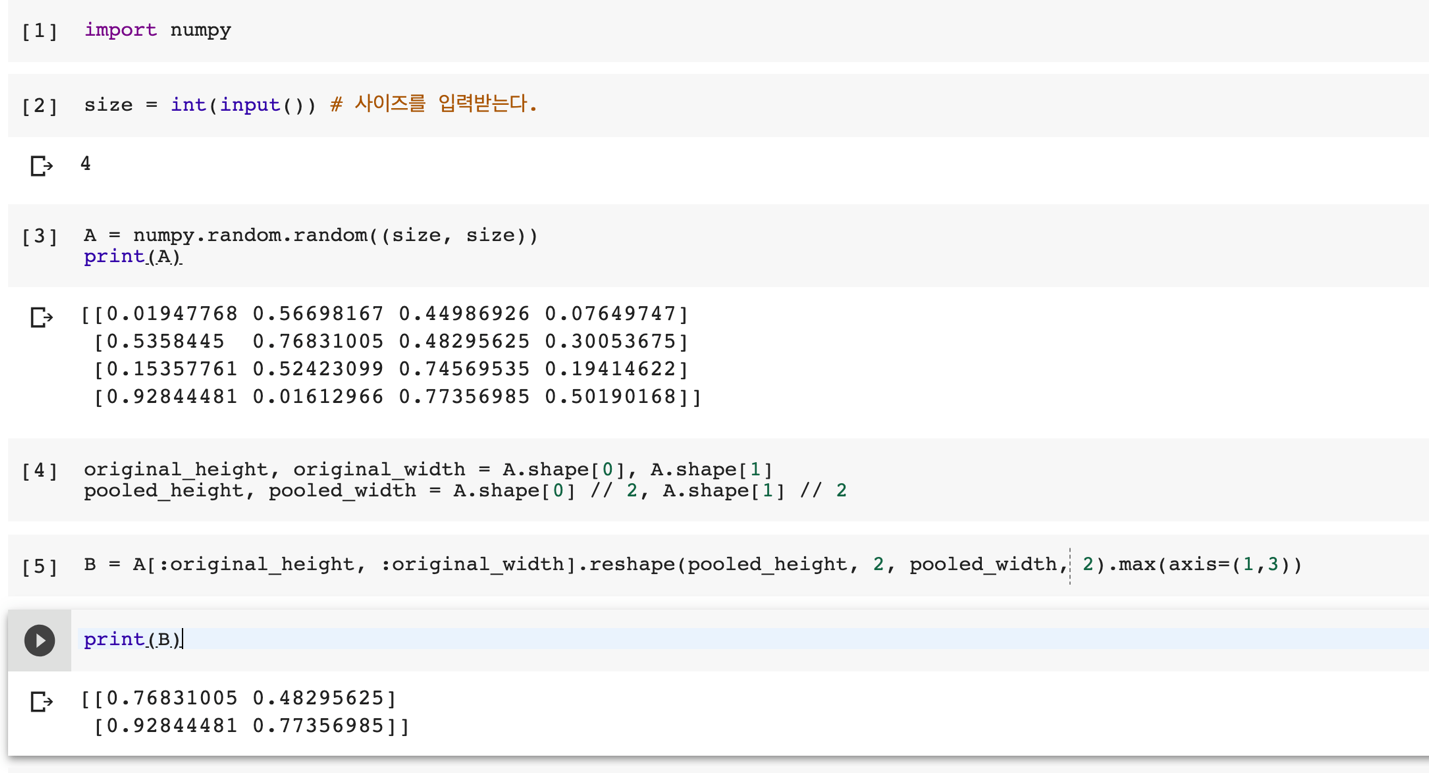
사용자로부터 입력을 받고, (0,0) ~ (N-1,N-1) 을 돌면서 (i, j)에 대하여 abs\_sum 을 진행하고 이를 abs\_sums에 저장한다. 최종적으로 abs\_sums 리스트의 최소값을 출력한다.

### 사용 변수 및 함수

* input\_array : 사용자로부터 배열을 입력받는 익명 함수 변수
* A : 입력받은 배열
* N : A의 길이
* abs\_sums : abs sum 의 결과들을 저장하는 리스트

## Max pooling 문제 풀이 및 알고리즘 설명

### 전체 구현 모습



### 풀이 및 알고리즘 설명

사용자로 부터 사이즈를 입력받고 랜덤한 배열(A)를 numpy를 통해 생성한다.

그 다음 A 의 shape를 4 차원 형태인 (원래높이//2)\*2\*(원래너비//2)\*2 형태로 변환(reshape)한다.

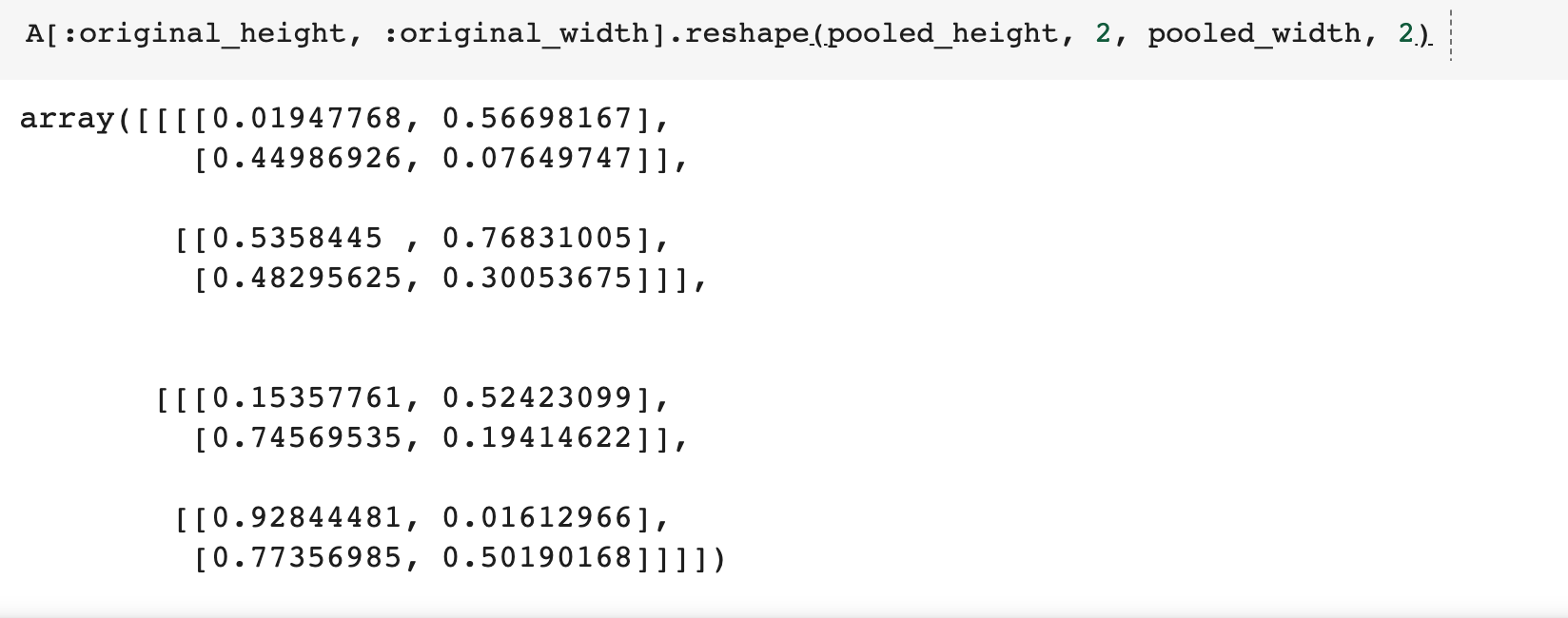
그 다음 max 를 axis (1, 3) 을 기준으로 실행 시키면 맥스 풀링한 결과가 나오게 된다.

추가로 설명을 하자면 A 를 reshape 한 형태는 아래와 같다.

첫번째 결과인 0.768은 (0.019, 0.566) 과 (0.535, 0.768) 에 대하여 max 를 하는 것임.

Axis=1이라는 것은 (같은 일차원 배열 끼리)“0.019”와 “0.566”을 max 비교하는 것이라면

Axis=(1,3)이라는 것은 같은 일차원 배열 끼리 + 같은 삼차원 끼리 max 비교 하는 것이라고 볼수있다.



### 사용 변수 및 함수

* size: 사용자로부터 입력받은 size
* A : size\*size 인 랜덤한 배열
* B : 결과를 저장할 배열
* original\_height : 원래 높이
* original\_width : 원래 너비
* pooled\_height : 맥스풀링 이후의 높이
* pooled\_width : 맥스풀링 이후의 너비

# 전역 변수 정의

해당사항 없음.

# 코드

## p1.ipynb

{

"cells": [

{

"cell\_type": "code",

"execution\_count": 1,

"metadata": {},

"outputs": [],

"source": [

"input\_array = lambda: list(map(int, input().split()))"

]

},

{

"cell\_type": "code",

"execution\_count": 2,

"metadata": {},

"outputs": [],

"source": [

"def max\_counter(N, A):\n",

" counters = [0]\*N\n",

" max\_value = 0\n",

" for element in A:\n",

" if 1 <= element <= N:\n",

" counters[element-1] += 1\n",

" if max\_value < counters[element-1]:\n",

" max\_value = counters[element-1]\n",

" elif element == N + 1:\n",

" counters[:] = [max\_value]\*N\n",

" return counters"

]

},

{

"cell\_type": "code",

"execution\_count": 3,

"metadata": {},

"outputs": [

{

"name": "stdin",

"output\_type": "stream",

"text": [

" 5\n"

]

}

],

"source": [

"N = int(input())"

]

},

{

"cell\_type": "code",

"execution\_count": 4,

"metadata": {},

"outputs": [

{

"name": "stdin",

"output\_type": "stream",

"text": [

" 3 4 4 6 1 4 4\n"

]

}

],

"source": [

"A = input\_array()"

]

},

{

"cell\_type": "code",

"execution\_count": 5,

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"3 2 2 4 2\n"

]

}

],

"source": [

"print(\*max\_counter(N, A))"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": []

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": []

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": []

}

],

"metadata": {

"kernelspec": {

"display\_name": "Python 3",

"language": "python",

"name": "python3"

},

"language\_info": {

"codemirror\_mode": {

"name": "ipython",

"version": 3

},

"file\_extension": ".py",

"mimetype": "text/x-python",

"name": "python",

"nbconvert\_exporter": "python",

"pygments\_lexer": "ipython3",

"version": "3.7.3"

}

},

"nbformat": 4,

"nbformat\_minor": 2

}

## p2.ipynb

{

"cells": [

{

"cell\_type": "code",

"execution\_count": 1,

"metadata": {},

"outputs": [],

"source": [

"input\_array = lambda: list(map(int, input().split()))"

]

},

{

"cell\_type": "code",

"execution\_count": 2,

"metadata": {},

"outputs": [],

"source": [

"def prime\_divisors(N):\n",

" i=1\n",

" res = set()\n",

" for i in range(1, N+1):\n",

" k = 0\n",

" if N%i == 0:\n",

" for j in range(1, i+1):\n",

" if i%j == 0:\n",

" k = k+1\n",

" if k == 2:\n",

" res.add(i)\n",

" return res"

]

},

{

"cell\_type": "code",

"execution\_count": 3,

"metadata": {},

"outputs": [],

"source": [

"is\_same\_prime\_divisors\_pair = lambda num\_pair: prime\_divisors(num\_pair[0]) == prime\_divisors(num\_pair[1])"

]

},

{

"cell\_type": "code",

"execution\_count": 4,

"metadata": {},

"outputs": [

{

"name": "stdin",

"output\_type": "stream",

"text": [

" 15 10 3\n"

]

}

],

"source": [

"A = input\_array()"

]

},

{

"cell\_type": "code",

"execution\_count": 5,

"metadata": {},

"outputs": [

{

"name": "stdin",

"output\_type": "stream",

"text": [

" 75 30 5\n"

]

}

],

"source": [

"B = input\_array()"

]

},

{

"cell\_type": "code",

"execution\_count": 6,

"metadata": {},

"outputs": [],

"source": [

"AB = list(zip(A, B))"

]

},

{

"cell\_type": "code",

"execution\_count": 7,

"metadata": {},

"outputs": [],

"source": [

"res = list(filter(is\_same\_prime\_divisors\_pair, AB))"

]

},

{

"cell\_type": "code",

"execution\_count": 8,

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"1\n"

]

}

],

"source": [

"print(len(res)) # answer"

]

}

],

"metadata": {

"kernelspec": {

"display\_name": "Python 3",

"language": "python",

"name": "python3"

},

"language\_info": {

"codemirror\_mode": {

"name": "ipython",

"version": 3

},

"file\_extension": ".py",

"mimetype": "text/x-python",

"name": "python",

"nbconvert\_exporter": "python",

"pygments\_lexer": "ipython3",

"version": "3.7.3"

}

},

"nbformat": 4,

"nbformat\_minor": 2

}

## p3.ipynb

{

"cells": [

{

"cell\_type": "code",

"execution\_count": 1,

"metadata": {},

"outputs": [],

"source": [

"input\_array = lambda: list(map(int, input().split()))"

]

},

{

"cell\_type": "code",

"execution\_count": 2,

"metadata": {},

"outputs": [

{

"name": "stdin",

"output\_type": "stream",

"text": [

" -8 4 5 -10 3\n"

]

}

],

"source": [

"A = input\_array()"

]

},

{

"cell\_type": "code",

"execution\_count": 3,

"metadata": {},

"outputs": [],

"source": [

"N = len(A)"

]

},

{

"cell\_type": "code",

"execution\_count": 4,

"metadata": {},

"outputs": [],

"source": [

"abs\_sums = [abs(A[i]+A[j]) for i in range(N) for j in range(i, N)]"

]

},

{

"cell\_type": "code",

"execution\_count": 5,

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"3\n"

]

}

],

"source": [

"print(min(abs\_sums)) # answer"

]

}

],

"metadata": {

"kernelspec": {

"display\_name": "Python 3",

"language": "python",

"name": "python3"

},

"language\_info": {

"codemirror\_mode": {

"name": "ipython",

"version": 3

},

"file\_extension": ".py",

"mimetype": "text/x-python",

"name": "python",

"nbconvert\_exporter": "python",

"pygments\_lexer": "ipython3",

"version": "3.7.3"

}

},

"nbformat": 4,

"nbformat\_minor": 2

}

## p4.ipynb

{

"cells": [

{

"cell\_type": "code",

"execution\_count": 9,

"metadata": {},

"outputs": [],

"source": [

"import numpy"

]

},

{

"cell\_type": "code",

"execution\_count": 10,

"metadata": {},

"outputs": [

{

"name": "stdin",

"output\_type": "stream",

"text": [

" 4\n"

]

}

],

"source": [

"size = int(input()) # 사이즈를 입력받는다."

]

},

{

"cell\_type": "code",

"execution\_count": 11,

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"[[0.30186356 0.38132771 0.69927142 0.80925284]\n",

" [0.55359032 0.89944734 0.271927 0.02417919]\n",

" [0.3207496 0.59262604 0.3929922 0.62027862]\n",

" [0.82409611 0.48994403 0.82402699 0.37745075]]\n"

]

}

],

"source": [

"A = numpy.random.random((size, size))\n",

"print(A)"

]

},

{

"cell\_type": "code",

"execution\_count": 12,

"metadata": {},

"outputs": [],

"source": [

"original\_height, original\_width = A.shape[0], A.shape[1]\n",

"pooled\_height, pooled\_width = A.shape[0] // 2, A.shape[1] // 2"

]

},

{

"cell\_type": "code",

"execution\_count": 13,

"metadata": {},

"outputs": [],

"source": [

"B = A[:original\_height, :original\_width].reshape(pooled\_height, 2, pooled\_width, 2).max(axis=(1,3)) "

]

},

{

"cell\_type": "code",

"execution\_count": 14,

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"[[0.89944734 0.80925284]\n",

" [0.82409611 0.82402699]]\n"

]

}

],

"source": [

"print(B)"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": []

}

],

"metadata": {

"kernelspec": {

"display\_name": "Python 3",

"language": "python",

"name": "python3"

},

"language\_info": {

"codemirror\_mode": {

"name": "ipython",

"version": 3

},

"file\_extension": ".py",

"mimetype": "text/x-python",

"name": "python",

"nbconvert\_exporter": "python",

"pygments\_lexer": "ipython3",

"version": "3.7.3"

}

},

"nbformat": 4,

"nbformat\_minor": 2

}