과목 명: 시스템프로그래밍

담당 교수 명: 소 정 민

<<Assignment 4>>

**서강대학교 컴퓨터공학과**

**[학번] 20171665**

**[이름] 이선호**

목 차

1. 프로그램 개요 2

2. 프로그램 설명 3

3. 모듈 정의 3

3.1 모듈 이름: p1.ipynb 3

3.1.1 풀이 및 알고리즘

3.2 모듈이름: p2.ipynb 4

3.2.1 풀이 및 알고리즘

3.3 모듈이름: p3.ipynb 4

3.3.1 풀이 및 알고리즘

3.4 모듈이름: p4.ipynb 5

3.4.1 풀이 및 알고리즘

4. 전역 변수 정의 5

5. 코드 5

5.1 p1.ipynb 5

5.2 p2.ipynb 6

5.3 p3.ipynb 7

5.4 p4.ipynb 7

# 프로그램 개요

이번 프로젝트에서는 파이썬 프로그래밍의 이해도를 높이고 실력을 향상시키기 위해 주어진 네 가지의 문제를 해결하는 프로그램을 작성하는 것을 목표로 했다. 모든 코드는 Python 3로 작성했으며, 구글에서 제공하는 Colab 환경을 이용했다. 프로그램을 처음 실행했을 때 사용자로부터 Input을 입력받고 문제에 맞는 연산을 수행한 후 화면에 Output을 출력하도록 프로그램을 작성했다. 네 개의 각 프로그램은 최종적으로 ‘.ipynb’ 확장자로 된 코드로 구현했다.

# 프로그램 설명

1번에서는 사용할 counter의 개수인 N을 사용자로부터 입력받고, 해당 개수만큼 counter들을 생성하여 처음에 0으로 모두 값을 초기화한다. 그리고 사용자로부터 일렬의 수로 이루어진 array A를 입력받아서 각 element 마다 처음부터 순서대로 앞에서 생성한 counter에 연산을 해준다. 만약 element의 값이 1 이상이고 N 이하이면 해당 element의 서수에 해당하는 counter를 1만큼 증가시킨다. 그렇지 않고 element 값이 N + 1이면 counter 중에서 가장 최댓값을 갖는 element의 값으로 모든 counter를 set해준다. 이와 같은 연사을 array A의 모든 element에 관해 차례대로 수행하며, 모든 연산이 완료되면 결과를 화면에 출력한다.

2번에서는 사용자로부터 두 개의 양의 정수들의 set인 N과 M을 입력받고, N과 M의 element들 중에서 순서대로 같은 prime divisor 의 set을 가지고 있는 element들의 개수를 모두 세서 출력하는 프로그램을 제작하는 것을 목표로 한다. 어떤 임의의 정수 P에 대하여 prime divisor를 구하라는 의미는 P를 어떠한 정수 D로 나누었을 때 나머지가 0이면서 해당 정수의 약수가 1과 자기자신 밖에 없는 소수인 D를 모두 구하라는 것을 뜻한다. 단, N과 M 배열을 입력받아서 각 element들마다의 prime divisor set을 구할 때 반드시 pyhton에 있는 data type인 set을 사용해야 한다.

3번에서는 사용자로부터 배열 A를 입력받고, 배열의 각 element 쌍마다의 합을 구하여 해당 값을 절댓값을 취한 결과 중에서 가장 작은 것의 값을 출력하는 것을 목표로 한다. 단, 쌍을 취할 때 쌍의 원소가 반드시 다를 필요는 없으며, 같은 값을 쌍으로 더했을 때의 절댓값도 연산에 포함한다.

4번에서는 사용자로부터 2차원 square 행렬의 size (행 또는 열의 크기)를 입력받고, np.random.random((size, size))로 임의의 numpy.ndarray를 data type으로 갖는 해당 size의 2차원 square 행렬을 생성한다. 그리고 size가 2이면서 서로 겹치치 않는 각 square로 해당 numpy array를 part로 구획지었을 때, part마다 가장 큰 값을 representative 값으로 갖도록 해서(이를 max pooling이라 칭함) 기존의 numpy array를 half size로 축소하는 과정을 수행한다. 그리고 결과를 화면에 출력한다. array A를 생성할 때와 max pooling 과정을 수행할 때 반드시 numpy 라이브러리를 사용하고 numpy.ndarray datat type을 사용해야 한다.

# 모듈 정의

## p1.ipynb

### 풀이 및 알고리즘

1. Input 함수를 통해 사용자로부터 생성할 카운터의 개수 N을 입력받는다. 생성할 카운터를 list data type인 CounterList로 선언하고, for loop을 N번 돌면서 CounterList에 append 함수를 사용하여 0을 삽입한다.
2. 카운터에 연산을 수행할 배열 A를 사용자로부터 입력받기 위해 InputList 변수를 list data type으로 선언한다. Input 함수를 통해 사용자로부터 입력받은 수로 된 string을 split 함수를 사용하여 공백을 기준으로 수들을 분리한다. 그리고 map 함수를 통해 이를 integer data type으로 mapping한 다음에 list 생성 함수를 사용하여 list로 생성한다. 이를 InputList에 할당한다.
3. For loop에서 i를 step 변수로 사용하여 i가 0부터 InputList의 길이 미만만큼 돌도록 하며, 만약 InputList의 i번째 element가 1 이상이고 N 이하이면 InputList의 i번째 element 값에서 1을 뺀만큼을 index로 CounterList에 대응되는 element 값에 1을 더한다. 그렇지 않고 InputList의 i번째 element 값이 N + 1과 같으면 j를 step 변수로 갖는 for loop을 0부터 N – 1까지 돌면서 가장 큰 MaxValue 값을 구하고, 다시 처음부터 CounterList를 for loop을 돌면서 모든 변수마다 MaxValue 값을 할당한다.
4. 모든 연산을 수행한 이후 l을 step 변수로 갖는 for loop을 0부터 InputList의 길이 미만만큼 돌면서 CounterList의 각 원소들을 print 함수를 사용하여 출력한다.

## p2.ipynb

### 풀이 및 알고리즘

1. 각 원소들마다 prime divisor set을 구할 양의 정수로 이루어진 배열 A와 B를 사용자로부터 입력받기 위해 변수 A와 B를 list data type으로 선언한다. Input 함수를 통해 사용자로부터 입력받은 수로 된 string을 split 함수를 사용하여 공백을 기준으로 수들을 분리한다. 그리고 map 함수를 통해 이를 integer data type으로 mapping한 다음에 list 생성 함수를 사용하여 list로 생성한다. 이를 각각 A와 B에 할당한다.
2. 변수 A와 B에 대해 Prime divisor set을 구하는 연산을 별도의 함수인 GetPDofList를 구현했다. A와 B list에 관해 각 대응되는 원소마다 같은 prime divisor set을 갖는 것들의 개수 값을 갖는 변수인 NumberofSamePD를 선언하고 이를 0으로 초기화한다. I를 step 변수로 갖는 for loop을 0부터 ListA의 길이만큼 돌면서 A와 B list의 각 원소마다 prime divisor set을 저장하기 위해 PDofListA, PDofListB라는 set data type 변수를 선언한다.
3. J를 step 변수로 갖는 for looop을 1부터 ListA[i]만큼 돌면서 ListA의 i번째 element를 j로 나눴을 때 나머지가 0인 것들을 구한다. 구한 j 값이 소수인지 확인하기 위해 k를 step 변수로 갖는 for loop을부터 j까지 돌면서 j를 k로 나눴을 때 0이 되는 경우 Counter값을 1만큼 늘린다. For loop을 모두 수행했을 때 Counter 값이 2라는 것은 해당 수 j가 소수라는 의미이므로 PDofListA에 add함수를 사용하여j를 set에 삽입한다. 마찬가지로 B에 관해서도 이를 수행한다.
4. 만약 PDofListA와 PDofListB가 같은 set이면 NumberofSamePD 값을 1만큼 늘린다. 이를 모든 ListA의 element에 관해 for loop을 돌면서 수행한다.
5. GetPDofList 함수는 최종적으로 NumberofSamePD값을 반환하고, 반환한 값을 print 함수를 이용하여 화면에 출력한다.

## p3.ipynb

### 풀이 및 알고리즘

1. 정수로 이루어진 배열을 사용자로부터 입력받기 위해 변수 ListA를 list data type으로 선언한다. Input 함수를 통해 사용자로부터 입력받은 수로 된 string을 split 함수를 사용하여 공백을 기준으로 수들을 분리한다. 그리고 map 함수를 통해 이를 integer data type으로 mapping한 다음에 list 생성 함수를 사용하여 list로 생성한다. 이를 ListA에 할당한다.
2. I를 step 변수로 갖는 for loop을 0부터 ListA의 길이 미만만큼 돌고, 또 그안에서 이중 for loop로써 j를 step 변수로 가지면서 j가 i에서 ListA의 길이 미만만큼 돌도록 한다. 이중 for loop 안에서 각 원소 쌍들의 합의 절댓값을 구하기 위해 AbsValue라는 변수를 선언하고 처음에 이를 0으로 초기화한다. ListA[i]와 ListA[j]를 더한 값을 AbsValue에 할당하고, 만약 AbsValue가 음수이면 절댓값이 아니므로 해당 변수에 -1를 곱한 값을 다시 할당한다.
3. 만약 i와 j가 모두 0값이어서 첫 원소에 관한 쌍일 때는 MinAbsValue 값이 아직 정해지지 않았으므로 무조건 AbsValue를 MinAbsValue에 할당한다. 그렇지 않고 만약 MinAbsValue가 AbsValue보다 크면 더 작은 절댓값을 가지는 쌍의 합을 찾았다는 뜻이므로 MinAbsValue를 AbsValue 값으로 업데이트한다. 이중 For loop을 모두 돌면 MinAbsValue 값을 print함수를 사용하여 출력한다.

## p4.ipynb

### 풀이 및 알고리즘

1. Numpy package를 사용하기 위해 impor를 사용하여 numpy를 불러오고 이를 축약하여 np로 지정한다. Input 함수를 통해 사용자로부터 생성할 numpy ndarray data type의 square matrix의 크기인 Size를 정수로 입력받는다.
2. Size만큼의 크기를 행 또는 열의 개수로 갖는 numpy data type의 square matrix를 0부터 1 사이의 임의의 값을 element 값으로 갖도록 생성하기 위해 numpy package에 있는 random module의 random 함수를 이용한다. 이 함수를 사용하면 0에서 1사이의 수를 반환하며, random 함수의 인자로 프로그램이 받고 싶은 자료의 크기를 입력할 수 있다. (Size, Size)를 random 함수의 인자로 입력하면 행렬과 열이 Size 크기이면서 각 원소의 값이 0에서 1 사이의 임의의 값인 numpy array가 반환된다. 따라서 np.random.random((Size, Size))를 사용하여 이를 numpy array인 A에 할당한다.
3. 우선 생성된 square matrix인 A를 출력해야 하므로 print 함수를 사용하여 A를 출력한다. 그리고 RowNum과 ColNum 변수에 A의 행 또는 열의 크기를 할당한다. 그리고 NewRowNum과 NewColNum에 각각 RowNum과 ColNum을 2로 integer 형식으로 나눈 값을 할당한다. 이를 수행하는 이유는 사용자로부터 입력받은 Size 값이 홀수일 때를 대처하기 위해서다.
4. Matrix A를 행과 열에 관해 각각 0부터 NewRowNum과 NewColNum의 두 배한 영역에서 reshape 함수를 이용하여 A를 축소하는 작업을 수행한다. 필터를 size가 2인 square matrix로 설정해야 하므로 reshape 함수의 인자 전달 순서에 주의하여 NewRowNum, 2, NewColNum, 2를 넘긴다. 크기를 축소할 때 해당 필터에서 가장 큰 값을 뽑아야 하므로 최댓값을 뽑는 axis 값을 1과 3으로 설정한다. Axis가 1이라는 것은 각 행마다 가장 최댓값을 계산한다는 뜻이고, 다시 axis가 3이라는 것은 뽑은 최댓값 중에서 다시 최댓값을 계산하는 것이다.
5. 계산한 결과를 변수 A에 할당하고 print 함수를 사용하여 max pooling을 수행한 결과를 출력한다.

# 전역 변수 정의

# 코드

## p1.ipynb

N = int(input("Input: \n"))

CounterList = []

for i in range(N):

CounterList.append(0)

InputList = []

InputList = list(map(int, input().split()))

for i in range(len(InputList)):

if (InputList[i] >= 1 and InputList[i] <= N):

CounterList[InputList[i] - 1] += 1

elif (InputList[i] == N + 1):

MaxValue = 0

for j in range(N):

if (CounterList[j] > MaxValue):

MaxValue = CounterList[j]

for k in range(N):

CounterList[k] = MaxValue

print("\nOutput: ")

for l in range(len(CounterList)):

print("%d"%CounterList[l], end=' ')

## p2.ipynb

def GetPDofList (ListA, ListB):

NumberofSamePD = 0

for i in range(len(ListA)):

PDofListA = set()

PDofListB = set()

for j in range(1, ListA[i] + 1):

if (ListA[i] % j == 0):

Counter = 0

for k in range(1, j + 1):

if (j % k == 0):

Counter += 1

if (Counter == 2):

PDofListA.add(j)

for j in range(1, ListB[i] + 1):

if (ListB[i] % j == 0):

Counter = 0

for k in range(1, j + 1):

if (j % k == 0):

Counter += 1

if (Counter == 2):

PDofListB.add(j)

if (PDofListA == PDofListB):

NumberofSamePD += 1

return NumberofSamePD

ListA = []

ListB = []

ListA = list(map(int, input("Input: \n").split()))

ListB = list(map(int, input().split()))

print("\nOutput: \n%d"%GetPDofList(ListA, ListB))

## p3.ipynb

ListA = []

ListA = list(map(int, input("Input: \n").split()))

for i in range(len(ListA)):

for j in range(i, len(ListA)):

AbsValue = 0

AbsValue = ListA[i] + ListA[j]

if (AbsValue < 0):

AbsValue \*= - 1

if (i == 0 and j == 0):

MinAbsValue = AbsValue

if (MinAbsValue > AbsValue):

MinAbsValue = AbsValue

print("\nOutput: \n%d"%MinAbsValue)

## p4.ipynb

import numpy as np

Size = int(input("Input: \n"))

A = np.random.random((Size, Size))

print("\nOutput: ")

print(A)

RowNum, ColNum = A.shape

NewRowNum = RowNum // 2

NewColNum = ColNum // 2

A = A[0:NewRowNum\*2, 0:NewColNum\*2].reshape(NewRowNum, 2, NewColNum, 2).max(axis=(1, 3))

print(A)