과목명: 시스템프로그래밍

담당교수명 : 소정민

<<Assignment 4>>

서강대학교 컴퓨터공학과

이정원20171672

목차

1. **프로그램 개요3**
2. **프로그램 설명3**
3. 모듈 정의**3**
   1. 3
   2. 4
   3. 4
   4. 4
4. **코드5**
5. **프로그램 개요**

이 프로젝트에서는 파이썬 프로그래밍에 대한 이해도를 높이는 것을 목적으로 한다.

1. **프로그램 설명**

* **1번 Max counters 문제**
* 정수 N이 주어지고, 비어 있지 않은 배열 A가 주어진다. A[k]가 1이상 N이하인 X이면, 크기가 N인 새 배열의 X-1 번째 인덱스에 1을 증가시키고, X가 N+1이면 새 배열의 모든 원소를 1씩 증가시킨다. 그리고 배열의 결과물을 출력한다.
* **2번 Common prime divisors 문제**
* 같은 크기인 리스트 A와 B가 주어지면, 각 원소의 소수를 구해 한 쌍의 소수 set이 같은 set인지 판별하여 그 개수를 출력한다.
* **3번 Min abs sum of two 문제**
* 비어 있지 않은 배열 A가 주어지고, 인덱스 쌍들의 합의 절댓값을 구해 그것들 중 가장 작은 (min) 값을 출력한다.
* **4번 Max pooling 문제**
* 정방 행렬의 크기를 입력받은 다음, 그 크기 x 크기에 해당하는 2차원 numpy 배열 (A)을 랜덤으로 생성하여 출력한다. 그런 다음, 그 numpy 배열을 반으로 줄인 다음 max pooling 과정을 통해 각 영역의 최대값을 반으로 줄인 배열(B)에 넣는다. 그리고 이 배열을 출력한다.

1. **모듈 정의**

|  |  |
| --- | --- |
| 풀이 및 알고리즘 | input() 함수를 이용해 N을 입력받는다. 그리고 배열 A를 입력받아 공백으로 split() 하여 정수 배열로 저장한다. 그리고 마지막에 출력할 결과가 담긴 배열 result를 N 크기만큼 0으로 초기화한다.  for문을 이용하여 A 원소를 X라고 하면, X가 1이상 N이하일 때 result[X-1]을 1 증가시킨다. 현재 최대값이 result[X-1]보다 작을 경우, current값을 업데이트한다. 그리고 X가 1이상 N이하가 아닐 경우 max\_counter에 current값을 넣어 준다. 따라서 result[X-1]을 1 증가시키기 전에 max\_counter가 result[X-1]보다 큰지 판별하여 그 자리에 max\_counter을 넣어 주는 일을 한다. for문이 끝나면, max\_counter보다 작은 result값이 있으면 그 자리에 max\_counter을 넣어 준다. 그리고 result를 출력한다. |

|  |  |
| --- | --- |
| 풀이 및 알고리즘 | 배열 A와 B를 입력받은 후 새로운 리스트 new 를 선언해 A[i], B[i]를 차례대로 append 함수를 이용해 넣어 준다. (0번째, 1번째 인덱스가 서로 한 쌍, 2번째, 3번째 인덱스가 서로 한 쌍이 되도록) new의 길이만큼 2씩 증가하는 for문을 이용하여 한 쌍의 한 원소의 소수 set을 구해 a에 저장하고, 나머지 원소의 소수 set을 구해 b에 저장한다.  소수를 구하는 방법은 1로 초기화된 i가 해당 숫자와 같아질 때까지 i가 해당 숫자를 나눴을 때 나머지가 0일 경우 그 과정을 똑같이 반복하여 또다른 몫을 구하면 k를 증가시킨다. k가 2일 경우 1과 그 수 자신 외의 정수로 똑 떨어지게 나눌 수 없는 정수, 즉 소수인 것이다.  for 문 마지막에 set a와 b가 같은지 판별하여 같은 경우 result 값을 1 증가시킨다. 마지막에 result를 출력한다. |



|  |  |
| --- | --- |
| 풀이 및 알고리즘 | 배열 A를 입력받은 후, i가 0부터 A의 길이만큼 돌아가는 for 문 안에 j가 i부터 A의 길이만큼 돌아가는 for문을 중첩하여 A[i]와 A[j]를 더해 abs 함수를 이용하여 절댓값을 구한 후 sums란 list에 append 한다. for문이 모두 끝나면 리스트 sums의 최소값을 min 함수를 이용해 구하여 출력한다. |

|  |  |
| --- | --- |
| 풀이 및 알고리즘 | numpy를 np로 import 한 다음 행렬의 크기를 size로 입력받는다. np.random.random((size,size))를 이용하여 size\*size 크기의 numpy 배열 A를 생성하여 출력한다. 그리고 size의 반 크기만큼 0인 행렬을 B로 생성한다. i,j가 size의 반크기 만큼 돌아가는 for문을 이용하여 A를 해당 영역만큼 slicing하여 np.max를 이용해 최대값을 구해 B[i,j] 자리에 넣는다. 그리고 B를 출력한다. |

1. **코드**

|  |
| --- |
| **p1.ipynb** |
| N = int(input())  temp= input().split()  A = [int (i) for i in temp]  result = N\*[0]  max\_counter = 0  current = 0  for X in A:  if 1 <= X <= N:  if max\_counter > result[X-1]:  result[X-1] = max\_counter  result[X-1] += 1  if current < result[X-1]:  current = result[X-1]  else:  max\_counter = current  for i in range(0,N):  if result[i] < max\_counter:  result[i] = max\_counter    for X in result:  print(X, end=' ') |

|  |
| --- |
| **p2.ipynb** |
| temp1= input().split()  A = [int (i) for i in temp1]  temp2= input().split()  B = [int (i) for i in temp2]  new = []  for i in range(len(A)):  new.append(A[i])  new.append(B[i])  result = 0  for x in range(0, len(new), 2):  a = set()  b = set()  i = 1  while i <= new[x]:  k = 0  if new[x] % i == 0:  p = 1  while p <= i:  if i % p == 0:  k = k+1  p = p +1  if(k == 2):  a.add(i)  i = i+1    i = 1  while i <= new[x+1]:  k = 0  if new[x+1] % i == 0:  p = 1  while p <= i:  if i % p == 0:  k = k+1  p = p +1  if(k == 2):  b.add(i)  i = i+1  if a == b:  result = result + 1  print(result) |

|  |
| --- |
| **p3.ipynb** |
| temp= input().split()  A = [int (i) for i in temp]  sums = []  for i in range(len(A)):  for j in range(i, len(A)):  sums.append(abs(A[i] + A[j]))  print(min(sums)) |

|  |
| --- |
| **p4.ipynb** |
| import numpy as np  size = int(input())  A = np.random.random((size,size))  B = np.zeros((size//2, size//2))  print(A)  for i in range(size//2):  for j in range(size//2):  B[i, j] = np.max(A[i\*2 : i\*2 +2, j\*2 : j\*2 + 2])    print(B) |