버블 정렬 n^2 swap(i,i+1)

```
array = [9,8,7,6,5,4,3,2,1] # array = [8,4,6,2,9,1,3,7,5]

def bubble_sort(array):
    n = len(array)
    for i in range(n - 1):
        for j in range(n - i - 1):
            if array[j] > array[j + 1]:
                array[j], array[j + 1] = array[j + 1],
array[j]
        print(array)

print("before: ",array)
bubble_sort(array)
print("after:", array)
```

선택 정렬 앞에 부터 하나씩 맞춤(swap) n^2

```
array = [8,4,6,2,9,1,3,7,5]

def selection_sort(array):
     n = len(array)
     for i in range(n):
          min_index = i
          for j in range(i + 1, n):
               if array[j] < array[min_index]:
                    min_index = j
          array[i], array[min_index] =  array[min_index],
array[i]
          print(array[:i+1])

print("before: ",array)
selection_sort(array)
print("after:", array)
```

삽입 정렬 for i 크기의 정렬된 배열 n^2

```
array = [8,4,6,2,9,1,3,7,5]

def insertion_sort(array):
     n = len(array)
     for i in range(1, n):
          for j in range(i, 0, - 1):
               if array[j - 1] > array[j]:
                    array[j - 1], array[j] = array[j], array[j
- 1]
          print(array[:i+1])

print("before: ",array)
insertion_sort(array)
print("after:", array)
```

병합 정렬 n log n == 내장 함수 1->2->4->... 크기의 정렬된 배열

```
array = [8,4,6,2,9,1,3,7,5]

def merge_sort(array):
     if len(array) < 2:
          return array
     mid = len(array) // 2
     low_arr = merge_sort(array[:mid])
     high_arr = merge_sort(array[mid:])

     merged_arr = []
     l = h = 0
     while l < len(low_arr) and h < len(high_arr):
          if low_arr[l] < high_arr[h]:
               merged_arr.append(low_arr[l])
               l += 1
          else:
               merged_arr.append(high_arr[h])
               h += 1
     merged_arr += low_arr[l:]
     merged_arr += high_arr[h:]
     print(merged_arr)
     return merged_arr

print("before: ",array)
array = merge_sort(array)
print("after:", array)
```

퀵 정렬 mid기준 왼쪽은 mid보다 작고, 오른쪽은 mid보다 큼

```
array = [8,4,6,2,5,1,3,7,9]

def quick_sort(array):
     if len(array) <= 1:
          return array
     pivot = len(array) // 2
     front_arr, pivot_arr, back_arr = [], [], []
     for value in array:
          if value < array[pivot]:
               front_arr.append(value)
          elif value > array[pivot]:
               back_arr.append(value)
          else:
               pivot_arr.append(value)
     print(front_arr, pivot_arr, back_arr)
     return quick_sort(front_arr) + quick_sort(pivot_arr) +
quick_sort(back_arr)

print("before: ",array)
array = quick_sort(array)
print("after:", array)
```

2차원 특정 인덱스 기준 정렬

arr.sort(key=lambda x: x[1]); sorted_arr=sorted(arr,key=lambda x: x[1])

```
# 이진탐색    n log n
def binary_search(target, data):
    data.sort()
    start = 0                # 맨 처음 위치
    end = len(data) - 1      # 맨 마지막 위치

    while start <= end:
        mid = (start + end) // 2 # 중간값

        if data[mid] == target:
            return mid           # target 위치 반환

        elif data[mid] > target: # target이 작으면 왼쪽을 더 탐색
            end = mid - 1
        else:                    # target이 크면 오른쪽을 더 탐색
            start = mid + 1
    return

# 올림
math.ceil(n)
# 내림 -3.14 → -4
math.floor(n)
# 버림 -3.14 → -3
int(n), math.trunc(n)
# n자리까지 반올림
xx=format(x,".nf")
# n자리까지 반올림하여 출력
print(f"{x:.nf}")
```

유니온 파인드
1
```
parent = list(range(n+1))
def find(x):
    if parent[x]!=x:
        parent[x] = find(parent[x])
    return parent[x]

def union(a,b):
    ra, rb = find(a), find(b)
    if ra!=rb:
        parent[rb] = ra
```

2
```
import sys
sys.setrecursionlimit(10**7)

def find(k):
  global P
  if P[k]==-1: return k
  #P[P[k]]=find(P[k])
  P[k]=find(P[k])
  return P[k]

P={} #P 초기화. 모든 요소가 부모
for i in range(n+1):
  P[i]=-1

for i in range(m):
  k,a,b=map(int,input().split())
  if k==1: # check
    if find(a)==find(b): print('YES')
    else: print('NO')

  else: # union
    if find(a)!=find(b): P[find(a)]=find(b)
```

```
# GCD (최대공약수)
math.gcd(a,b,c,...)
# lcm (최소공배수)
math.lcm(a,b,c,...)

# 소수(2,3,5,7,...) 판별      n log log n
is_prime = [True]*(N+1)
is_prime[0] = is_prime[1] = False
for i in range(2, int(N**0.5)+1):
    if is_prime[i]:
        for j in range(i*i, N+1, i):
            is_prime[j] = False
print(is_prime[N])

# 약수(12->[1,2,3,4,6]) 구하기   n^1/2
def divisors(n):
    small, large = [], []
    for i in range(1, int(n**0.5) + 1):
        if n % i == 0:
            small.append(i)
            if i != n // i:
                large.append(n // i)
    return small + large[::-1]

#  list
arr.append(x)   arr.pop(idx)   arr.insert(idx,x)   arr.extend(iterable)
arr.sort()   arr.reverse()   arr.clear()   arr.count(x)   arr.index(x)
# dictionary
dic.keys()   dic.values()   dic.items()   del dic[key]
dic={1:11, 2:22, 3:33}  #모든 출력 순서는 추가순임
dic.items() → [(1,11),(2,22),(3,33)]
dic.keys() → [1,2,3]
dic.values() → [11,22,33]
# set
s.add(x)   s.remove(x)   s1|s2   s1&s2   s2-s1
```

# sys
import sys
input=sys.stdin.readline
sstr=intput().rstrip() → strip: space, tab, \n 제거

# 조합, 순열
import itertools
itertools.combination(arr,k)
itertools.permutation(arr,k)

+)
gcd(*arr), lcm(*arr)