**분할 정복의 기초 - Merge-Sort**

|  |
| --- |
| *#n/2로 나누고, 1개씩 요소가 남기까지 재귀적으로 conquer 한다. #그 후, 그 다음에 2개씩의 요소들을 반복적으로 merge 한다.* |

1. Merge\_Sort

|  |  |
| --- | --- |
| 4 5 2 9 23 55 1 8  4 5 2 9 23 55 1 8  4 5 2 9 23 55 1 8  4 5 2 9 23 55 1 8 | **def** merge\_sort(file):  **if** len(file) <= 1:  **return** file  mid = len(file)//2  left, right = file[:mid], file[mid:]  *#재귀를 이용하여 나뉜 부분들을 다시 반으로 나눈다.* left1, right1 = merge\_sort(left), merge\_sort(right)  **return** merge(left1, right1) |

2. Merge

|  |  |
| --- | --- |
| 4 5 2 9 23 55 1 8  4 5 2 9 23 55 1 8  2 4 5 9 1 8 23 55  4 5 2 9 23 55 1 8 | **def** merge(left, right):  i,j = 0,0  sorted\_list = [] *#정렬할 리스트  #left, right길이 안에서 각각 0번째 자리부터 비교를 하고 작은 값을 sorted\_list에 넣어준다.* **while** i < len(left) **and** j < len(right):  **if** left[i] < right[j]:  sorted\_list.append(left[i])  i += 1  **else**:  sorted\_list.append(right[j])  j += 1  *#남은 값들을 넣어준다.* **while** (i < len(left)):   sorted\_list.append(left[i])  i += 1  **while** (j < len(right)):  sorted\_list.append(right[j])  j += 1  **return** sorted\_list |

3. Print Answer

|  |
| --- |
| print(merge\_sort(file)) |

#10090번 Counting Inversions

|  |
| --- |
| Inversion의 개수는 하나의 배열 안에서 자신보다 큰 수의 개수를 세면 되는 것이다.  그런데 이를 그냥 무턱대고 브루트 포스로 구하면 당연히 시간 초과가 발생할 것이기 때문에 merge 정렬을 이용해서 문제를 풀고자 한다.  분할 이후에 합치는 과정에서 왼쪽 분할 리스트에 있는 숫자는 당연히 왼쪽에 있는 것이기 때문에 left[i] > right[j]일 때에 len(left)-i만큼 더해주면 된다.  정렬된 상태에서의 부분 리스트이기 때문에 앞에 있는 수들 보다는 당연히 연속적으로 만족 될 것이다. |

|  |  |
| --- | --- |
| **import** sys n = int(sys.stdin.readline()) file = list(map(int, sys.stdin.readline().split()))  temp = [0]\*n ans = 0 **def** merge(l,m,r):  **global** ans  idx1, idx2, idx3 = l,m,0  **while** (idx1 < m **and** idx2 < r):  **if** file[idx1] < file[idx2]:  temp[idx3] = file[idx1]  idx3, idx1 = idx3+1, idx1+1  **elif** file[idx1] > file[idx2]:  ans += (m-idx1)  temp[idx3] = file[idx2]  idx3, idx2 = idx3+1, idx2+1  **else**:  temp[idx3] = file[idx2]  idx3, idx2 = idx3+1, idx2+1  **while** idx1 < m:  temp[idx3] = file[idx1]  idx3, idx1 = idx3+1, idx1+1  **while** idx2 < r:  temp[idx3] = file[idx2]  idx3, idx2 = idx3+1, idx2 + 1  **for** i **in** range(r-l):  file[l+i] = temp[i]  **def** merge\_sort(start, end):  mid = (start+end)//2  **if** start <mid:  merge\_sort(start, mid)  merge\_sort(mid, end)  merge(start, mid, end)  merge\_sort(0, n) print(ans) | **import** sys n = int(sys.stdin.readline()) file = list(map(int, sys.stdin.readline().split())) *#n/2로 나누고, 1개씩 요소가 남기까지 재귀적으로 conquer 한다. #그 후, 그 다음에 2개씩의 요소들을 반복적으로 merge 한다.* ans = 0 **def** merge(left, right):  **global** ans  i,j = 0,0  sorted\_list = [] *#정렬할 리스트  #left, right길이 안에서 각각 0번째 자리부터 비교를 하고 작은 값을 sorted\_list에 넣어준다.* **while** i < len(left) **and** j < len(right):  **if** left[i] < right[j]:  sorted\_list.append(left[i])  i += 1  **elif** left[i] > right[j]:  ans += len(left)-i  sorted\_list.append(right[j])  j += 1  **else**:  sorted\_list.append(left[i])  i += 1  *#남은 값들을 넣어준다.* **while** (i < len(left)):  sorted\_list.append(left[i])  i += 1  **while** (j < len(right)):  sorted\_list.append(right[j])  j += 1  **return** sorted\_list   **def** merge\_sort(file):  **if** len(file) <= 1:  **return** file  mid = len(file)//2  left, right = file[:mid], file[mid:]  *#재귀를 이용하여 나뉜 부분들을 다시 반으로 나눈다.* left1, right1 = merge\_sort(left), merge\_sort(right)  **return** merge(left1, right1) merge\_sort(file) print(ans) |
| 메모리 244884KB 시간 1092ms | 메모리 256548KB 시간 1084ms |

#2104번 부분 배열 고르기

|  |
| --- |
| i부터 j까지의 합에 i부터 j까지의 최솟값을 곱한 것이 점수가 된다.  여기서 재귀적으로 계속 반으로 나누어서 구하고 뿐만 아니라 가운데 mid값을 뺴 놓을 수도 있기 때문에 가운데부터 밖으로 최댓값을 갱신하는 conquer을 한번 더 진행함으로서 시간 초과 없이 답을 구할 수 있다. |

|  |  |
| --- | --- |
| **import** sys n = int(input()) file = list(map(int, sys.stdin.readline().split()))  **def** check(a,b):  **if** a == b:  **return** file[a]\*file[b]  mid = (a+b)//2  ans = max(check(mid+1, b), check(a, mid))  left, right = mid, mid+1  add = file[left]+file[right]  small = min(file[left], file[right])  **while** (left > a **or** right < b):  **if** right < b **and** (left == a **or** file[left-1] <file[right+1]):  right += 1  add += file[right]  small = min(small, file[right])  **else**:  left -= 1  add += file[left]  small = min(small, file[left])  ans = max(ans, small\*add)  **return** ans  print(check(0, n-1)) | 전체 답을 구하기 위해, 즉 앞, 뒤 각각 뿐 아니라 앞쪽 절반, 뒤쪽 절반 각각에서 정보를 추출하여 합쳐서 전체 답을 구하는 과정  그래서 mid, mid+1값을 반드시 포함하도록 해야 함  부분 답을 구하는 과정 |

#1629번 곱셈

|  |
| --- |
| 자연수 a를 b번 곱한 수를 c로 나눈 나머지를 구하는 프로그램을 작성하는 문제이다.  솔직히 왜 분할정복을 이용해야 하는지는 잘 모르겠지만  어쨌든 pow()라는 파이썬 함수가 존재하는 관계로 분할정복 카테고리에 해당된다는 사실에 너무 짜증이 난다.  pow(a,b,c)를 하면 알아서 위의 값이 계산되어 나온다. |

|  |  |
| --- | --- |
| **import** sys a,b,c = map(int, sys.stdin.readline().split()) ans = 1 **def** multi(x):  **if** x == 1:  **return** a%c  now = multi(x//2)  **if** x%2 == 0:  **return**((now\*\*2)%c)  **else**:  **return**((now\*\*2\*a)%c) print(multi(b)) | 이 문제 한정으로 중복을 없애서 시간 복잡도를 줄여주는 방법이 존재한다.  그러나 b가 홀수일때는 양쪽이 항상 같은 값이 나올 수가 없다. 곱셈에는 결합법칙이 성립하기 때문에 왼쪽과 같은 풀이가 가능하다.  따라서 만약에 홀수 번 곱해야 할 때에는 구한 한쪽 값의 제곱을 c로 나눈 나머지를, 짝수 번 곱해야 할 때는 구한 한쪽 값의 제곱에 a를 한번 더 곱한 값의 나머지를 구해준다.  이를 뒤에 나오는 행렬의 곱셈에도 적용해 보자. |

#1780번 종이의 개수

|  |
| --- |
| 이 마저도 파이썬의 느린 속도 때문에 pypy3으로 해야만 시간초과가 발생하지 않는다.  어쨌든 함수의 변수로 주어진 값들은 시작점의 x,y 좌표와 현재 정사각형의 한 변의 길이이다.  이를 이용하여서 원래 나는 2차원 리스트 슬라이싱을 통해 재귀적으로 함수를 부르려 했으나 그렇게 할 때 마지막, 혹은 첫 칸이 포함이 안될 수가 있었기 때문에 좌표를 이용하는 것으로 방법을 바꾸었다.  이후에 모두 같은 숫자로 이루어진 것이 아니면 2중 for문을 통해 재귀적으로 함수를 불러서 최종적으로 답인 ans1, ans2, ans3을 출력하도록 했다. |
| **import** sys n = int(sys.stdin.readline()) file = [list(map(int, sys.stdin.readline().split())) **for** \_ **in** range(n)] ans1, ans2, ans3 = 0,0,0  **def** check(x,y,l):  **global** ans1, ans2, ans3  **if** l == 0:  **return** c1,c2,c3 = 0,0,0  **for** i **in** range(x, x+l):  **for** j **in** range(y, y+l):  **if** file[i][j] == -1:  c1 += 1  **elif** file[i][j] == 0:  c2 += 1  **else**:  c3 += 1  **if** c1 == l\*\*2:  ans1 += 1  **elif** c2 == l\*\*2:  ans2 += 1  **elif** c3 == l\*\*2:  ans3 += 1  **else**:  **for** i **in** range(3):  **for** j **in** range(3):  check(x+i\*l//3, y+j\*l//3, l//3) check(0,0,n) print(ans1, ans2, ans3) |