Autor omówienia: Jakub Bartmiński

Piramidy

Zadanie

Po pierwsze dekodujemy treść: dla każdej liczby z wejścia mamy po prostu podać ile jest mniejszych liczb na lewo od niej (liczby są parami różne).

Rozwiązania

Rozwiązanie wzorcowe

Klepiemy drzewo licznikowe. Skalujemy liczby z wejścia a następnie iterujemy po przeskalowanych liczbach. Dla każdej przeskalowanej liczby a_i, najpierw wypisujemy sumę z drzewa na przedziale $[0, a_i - 1]$. Następnie dodajemy 1 do komórki a_i.

Rozwiązanie alternatywne

Klepiemy mergesort. Podczas merge'owania dwóch ciągów zliczamy inwersje dla każdego elementu osobno – najłatwiej do mapy z wartości elementu w liczbę inwersji.

Dla przypomnienia: w trakcie merge'owania posortowanych ciągów L i P, jeśli pierwszy pozostały element P jest mniejszy od pierwszego pozostałego elementu L, to jest też mniejszy od wszystkich pozostałych elementów L. Zabieramy go do wynikowego ciągu, więc "przeskakuje" wszystkie te pozostałe elementy L, czyli tworzył z każdym z nich inwersję.

Mamy liczyć "odwrotne" inwersje niż zwykle - mniejsza liczba stoi przed większą. Da się to zrobić na kilka sposobów, ale najprościej sortować ciąg malejąco zamiast rosnąco – wszystko magicznie zadziała dobrze.

Uwagi

Oryginalnie w paczce był bardzo restrykcyjny limit 32 MB – co wywalało skalowanie z użyciem unordered_map albo map. Można wspomnieć jak robić skalowanie przy limitowanej pamięci – po prostu posortować wektor par {wartość, oryginalna_kolejność} po wartościach, przeskalować wartości, następnie posortować po oryginalnej kolejności.