**Задача A? Сортиране**

**Пояснение към решенията**

**Чрез динамично оптимиране със сложност O(n3).** Запълваме таблица dp[i][j] за оптимална стойност при подзадачи, за които се ограничават с първите i елемента от дадената редица и при които промяната на стойностите е такава, че i-тият елемент е получил стойност, равна на j. Лесно се съобразява, че оптималното решение е такова, че променените стойности на елементите на дадената редица са в интервал от a1=min a[i] do a2=max a[i]. Така запълваме dp[0][j]=abs(a[0]−j).

След като сме запълнили dp[0][j], dp[1][j], …, dp[i−1][j] за всички j=a1, …, a2, може да запълним стойността dp[i][j] за всяко j=а1, …, а2, като ползваме dp[i−1][jj] за j >= jj, чрез намираме минимума по jj на dp[i−1][jj]+abs(a[i]−j). След запълване на цялата таблица dp[i][j], вземаме минимума по j от стойностите dp[n−1][j]. Програмата, реализираща този алгоритъм ше осигури около 54% от точките.

**Чрез динамично оптимиране със сложност O(n2).** Чрез подходяща модификация на гореописания алгоритъм може да избегнем двойния цикъл при пресмятането на стойността dp[i][j], както се вижда от съответната програма, което ще осигури около 85% от точките.

**Решение със сложност O(n log n)**, което осигурява 100% от точките се построява чрез използване на приоритетна опашка pq. Вкарваме последователно елементите от дадената редица в pq и всеки път проверяваме дали най-големият елемент от pq в по-голям от елемента от редицата, който ще вкарваме (pq.top() > a[i]) и ако е така, в променливата s натрупваме разликата pq.top()−a[i]), с което записваме необходимия брой операции за тази стъпка и изкарваме този най-голям елемент от pq. Накрая натрупаната стойност в s дава решението.

*Емил Келеведжиев*