Aluno: André Santos Rocha
Ra: 235887

Problema 1: A - Quick Sort

O problema fornece uma permutação de comprimento n e um inteiro positivo k. Devemos performar o menor número de operações para ordenar a permutação em ordem crescente. Uma operação é definida como:

- escolha k elementos distintos;
- remova-os da lista, ordene-os e insira-os no fim da lista.

1.1 Ideia de solução

Para encontrar o menor número de operações é preciso que deixemos o maior número possível de elementos de fora das operações. Perceba que, se deixarmos w elementos de fora das operações, esses elementos aparecerão nas primeiras posições da lista, na mesma ordem que estavam na lista original. Logo, é preciso que esses elementos já estejam ordenados. Queremos, então, encontrar a maior sequência w de elementos já ordenados, mesmo que eles não estejam adjacentes na lista original. Daí nos resta n-w elementos para operar. Como a cada operação, alteramos k elementos, teremos (n-w)/k operações (teto). Logo, a solução fica da seguinte forma: percorremos a permutação em busca da maior subsequência ordenada e depois aplicamos a fórmula para determinar o número de operações.

1.2 Detalhes de implementação

Para encontrar o comprimento da maior subsequência, podemos adotar a seguinte estratégia: setamos uma variável w para 1; percorremos a permutação de entrada de 0 a n; cada vez que encontrarmos um elemento da permutação que seja igual a w, somamos 1 a w. Com isso, procuramos pelos elementos iniciais da ordenação e verificamos se eles aparecem em sequência.

Problema 2: E - Sum of Medians

O problema nos fornece nk inteiros positivos, devemos dividir esses inteiros em k arrays de tamanho n, tal que cada inteiro pertença a exatamente um array e a soma dos medianos seja a maior possível. Um mediano consiste no elemento que ocupa a posição n/2 (arredondando para cima) do array ordenado de modo não-decrescente.

2.1 Ideia de solução

Para este problema, utilizamos uma abordagem gulosa. A cada iteração do nosso algoritmo percorremos os maiores medianos, nos aproveitando do fato do vetor estar ordenado. Na primeira iteração, sabemos que o índice para o primeiro mediano pode ser encontrado com (n-1)/2*k. A partir daí, iteramos para os próximos medianos utilizando o incremento n/2+1. Isso funciona porque n/2 representa a metade do array e o +1 ajusta o índice para pegarmos o elemento correto. Por fim, a cada iteração computamos o valor do mediano e o acrescentamos a uma variável s. A cada caso de teste, basta printar a soma s e teremos o resultado correto.

2.2 Detalhes de implementação

Essa implementação é no pior caso O(n*k), que serve também de limite para o nosso loop for.