Aula 03 - Exercício prático Ordenação Quadrática

Aluno: Gian Franco Joel Condori Luna

August 27, 2024

Exercício no Beecrowd

Para resolver o problema de permutações ordenadas no ambiente Beecrowd contamos com o algoritmo Trotter-Johnson.

Referências:

- Algorithms in C.
- Slides decom.ufop.

1.1 Algoritmo Trotter-Johnson

O algoritmo Trotter-Johnson, também conhecido como algoritmo de permutação Steinhaus-Johnson-Trotter ou algoritmo de permutação de transposição, foi desenvolvido independentemente por várias pessoas. O algoritmo foi popularizado na literatura matemática e é amplamente utilizado devido à sua simplicidade e eficiência.

Este algoritmo é conhecido por sua capacidade de gerar permutações sucessivas através de transposições simples (trocas). O algoritmo segue estas etapas gerais:

- Caso base: Se o conjunto tiver apenas um elemento, a permutação é o próprio conjunto.
- Recursão: Para gerar as permutações de um conjunto de n elementos:
  - Primeiro, gere todas as permutações do subconjunto dos primeiros n-1 elementos.
  - Em seguida, insere o enésimo elemento em todas as posições possíveis das permutações geradas acima.
- Inserções alternadas: A direção em que o enésimo elemento é inserido alterna da esquerda para a direita e da direita para a esquerda para evitar duplicatas e garantir que cada permutação difere da anterior em apenas uma troca.

Aqui mostramos o pseudocódigo que descreve o algoritmo Trotter – Johnson, que é a base para o código feito em Python:

```
procedure generate_permutations(n):
    p := array of integers from 1 to n
    c := array of zeros with length n
    print p

    i := 0
    while i < n:
        if c[i] < i:
            if i is even then
                swap p[0] and p[i]
            else
                swap p[c[i]] and p[i]

            print p
            c[i] += 1
            i := 0
        else
            c[i] := 0
            i += 1
```

1.2 Análise de complexidade

```
def trotter_johnson_permutations(lista):
  result = []
  n = len(lista)
  c = [0] * n  # Inicialize os contadores
  result.append(''.join(lista))

  i = 0
  while i < n:
    if c[i] < i:
      if i % 2 == 0:
        swap_with = 0
      else:
        swap_with = c[i]

      # Troque os itens
      lista[swap_with], lista[i] = lista[i], lista[
          swap_with]
      result.append(''.join(lista))

      c[i] += 1
```

```
      i = 0
    else:
      c[i] = 0
      i += 1
  return result
//Fim da funcao trotter_johnson_permutations

n = int(input())
for _ in range(n):
  cadena = input()
  lista = list(cadena)
  result = trotter_johnson_permutations(lista)
  result = list(set(result))
  result.sort()

  for r in result:
    print(r)
  print()
```

1.2.1 Inicialización:

- $result = []$: Operação constante $O(1)$
- $n = len(lista)$: Operação constante $O(1)$
- $c = [0] * n$: Operação linear $O(n)$

1.2.2 Loop principal ($while \quad i < n$):

Tudo dentro do $while$ será executado n vezes

- $lista[swap\_with], lista[i] = lista[i], lista[swap\_with]$: Operação fatorial $O(n!)$ pois terá que realizar todas as permutações que existem de n

Portanto o while e o que ele contém (permutações) serão executados $O(n! * n)$

1.3 Resposta

$$O(n) + O(n! * n) = O(n! * n)$$