Documentação QuickSort em Go Lang

Executation()

Função encarregada de chamar e organizar as outras funções, primeiro ela chama a *generationSlice()*, que é encarregada de gerar o vetor, faz uso de duas variáveis start e end, que recebem o tempo inicial de final da execução da função *quicksort()*, só sendo possível pelo uso da função *time.Now()*, assim chamando a *quicksort()* para ordenar o vetor passado de tamanho e conteúdo aleatórios para ela. Para finalizar utiliza-se a funcao *.Sub()* entre start e end nos retornando na variável resultado o tempo de execução em milissegundos, porém como queremos o resultado em segundos utilizamos a função *.Seconds()* pra fazer esse tratamento por nós.

- ❏ *Time.Now():*
  - Uma variável recebe o tempo marcado com a chamada da time.Now(), no caso do nosso código a variável start e end recebem o tempo de início e fim respectivamente o tempo retornado pela funcao é em milisegundos.
- ❏ *.Sub():*
  - Função que realiza a subtração entre duas variáveis.
- ❏ *.Seconds()*
  - Função que transforma a medida de tempo recebida para segundos.

GenerationSlice()

Função encarregada da criação do vetor e seu preenchimento, para a criação do vetor utilizou-se a função *make()* que recebeu seu tamanho como parâmetro. Após sua criação inicia-se o preenchimento com números aleatórios, tais números serão gerados pela função *rand.Intn()*, que gerará números de 0 até o limite que será passado como parâmetro, para o vetor possuir números negativos utilizou-se a função para gerar dois números randômicos e fazer a subtração entre eles assim podendo gerar números aleatórios inteiros tanto positivos quanto negativos, após sua finalização retorna um vetor preenchido aleatoriamente.

- ❏ *Make()*
  - Função que cria uma matriz(vetor) dinâmica e à retorna.
- ❏ *Rand.Intn()*
  - Função que gera um valor inteiro randômico de 0 até o valor passado como parâmetro para esta função.

QuickSort()

A função recebe como parâmetro o vetor passado pela *executation()*, a partir do vetor passado será feito a seleção de um pivô, o qual escolhemos utilizar sempre o primeiro elemento como pivô, após isso todos os elementos menores que o pivô ficaram à sua esquerda e os maiores aa sua direita, assim o pivô já se encontrará em sua posição correta, como a função é recursiva ela chamara ela mesmo em duas situações, onde ordenará todo o lado esquerdo da função até que se encontre ordenado, logo após ordenara todo o lado direito da função, esse método de partição corresponde a algoritmos recursivos que utilizam o método dividir para conquistar, que é nítido no quicksort pelo fato de poder ver esse particionamento ocorrer.