Trabalho 5: Problema BeeCrowd #1084 via Programação Gulosa

Nomes: Henrique Fugie de Macedo e Luiz Otávio Silva Cunha
Matrículas: 0056151 e 0056153

Problema BeeCrowd #1084

O problema tratado pelo BeeCrowd#1084 é chamado de Apagando e Ganhando, o problema é este:

Juliano é fã do programa de auditório Apagando e Ganhando, um programa no qual os participantes são selecionados via um sorteio e recebem prêmios em dinheiro por participarem. No programa, o apresentador escreve um número de N dígitos em uma lousa. O participante então deve apagar exatamente D dígitos do número que está na lousa; o número formado pelos dígitos que restaram é então o prêmio do participante. Juliano foi finalmente selecionado para participar do programa, e pediu que você escrevesse um programa que, dados o número que o apresentador escreveu na lousa, e quantos dígitos Juliano tem que apagar, determina o valor do maior prêmio que Juliano pode ganhar.

Basicamente, o problema vai receber como entrada dois números de começo, um contendo a quantidade de números que o número que será fornecido possui e a quantidade de número que o programa precisa retirar para, retornando o maior prêmio que o Juliano pode receber. Estes são alguns exemplos de entradas, e o programa deve parar ao receber 0 0.

Exemplo de Entrada	*Exemplo de Saída*
4 2	
3759	*79*
6 3	
123123	*323*
7 4	
1000000	*100*
7 3	
1001234	*1234*
6 2	
103759	*3759*
0 0	

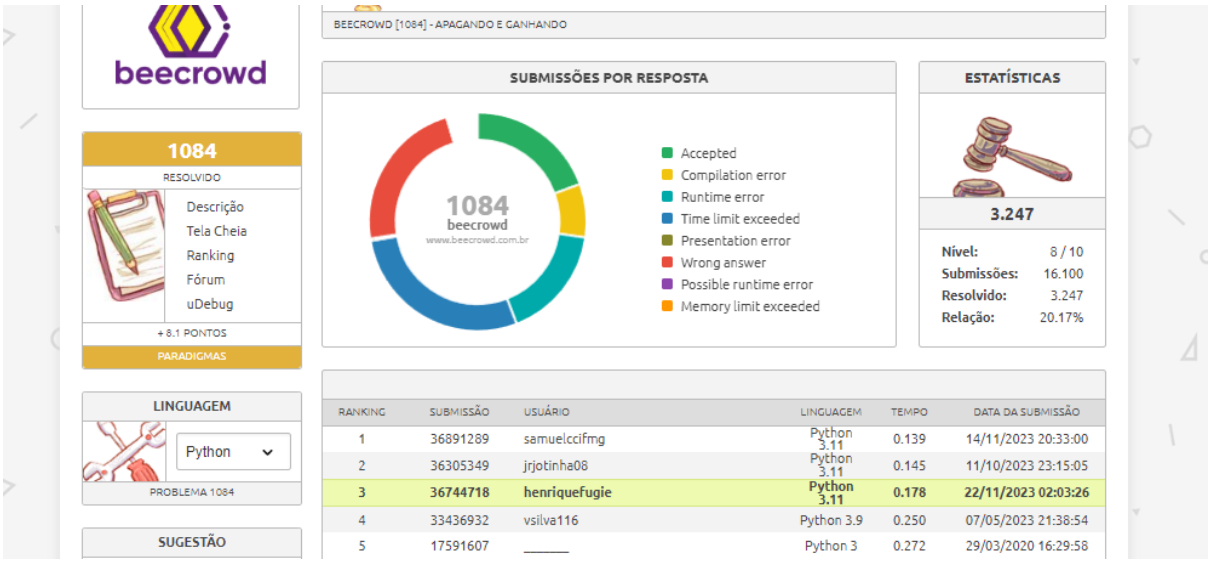
Programação Gulosa

O paradigma de programação gulosa, também conhecido como algoritmo guloso ou greedy, é uma abordagem de resolução de problemas que segue a estratégia de fazer a escolha localmente ótima em cada estágio com a esperança de encontrar uma solução globalmente ótima. Em outras palavras, as decisões são tomadas de maneira incremental, escolhendo a opção que parece ser a melhor no momento, sem considerar as escolhas posteriormente. Alguns elementos fundamentais do paradigma são: a escolha gulosa, onde em cada estágio do algoritmo, uma escolha mais vantajosa é escolhida no momento, que uma vez tomada, ela não é revogada ou alterada, e isso é o que diferencia ela de outros paradigmas, como a programação dinâmica, onde as decisões podem ser avaliadas à medida que o algoritmo avança.

A eficácia da programação gulosa depende da natureza do problema, ela possui uma velocidade muito boa, comparada com as demais ,porém, a escolha de soluções locais ótimas em cada estágio, não garante que a solução global seja a melhor possível em todos os casos.

Explicação do código

A solução deste trabalho foi upada para o bee crowd e recebeu colocação, na sessão de python, de terceiro lugar.



O código começa pedindo os 2 números de entrada, o primeiro número sendo ele o tamanho do número que o usuário vai digitar e o segundo a quantidade de números que o programa terá de remover para encontrar o maior prêmio. Ele também faz uma verificação se a entrada com os dois números não foi igual a zero, pois a entrada 0 0 termina o while True

que faz o programa ficar executando até uma ordem de parada, por fim na main ele chama a função maior_premio() que é a função que faz todo o trabalho do código utilizando programação gulosa.

A função maior_premio() recebe os dois números digitados na entrada e o número que vai ser o maior prêmio, ele começa gerando um pilham foi utilizado “pilha” por fica mais fácil a manipulação dos números e para garantir que ao remover algum número do meio, os números não fiquem embaralhados, ou na ordem errada, seguido de um laço dígito em número, ou seja, ele vai percorrer todos os números, 1 por 1 que foi digitado para executar o algoritmo. A escolha gulosa utilizada para escolher quais números serão removidos, é feita através de uma verificação dentro de um while no código, que possui junto disso, algumas marcações para garantir que o programa vai executar corretamente, não vou detalhar pois no código os comentários já explicam o que essas condições fazem, eu sempre vou tentar deixar o número de maior valor na casa mais à esquerda do número, para assim alcançar um número de valor maior. Ela é feita de seguinte forma, ele adiciona o número na pilha, depois disso, observando outro dígito com o for, ele verifica se o dígito é maior que o último número contido na pilha, se sim, ele vai remover o numero que esta na pilha e decrementar o d, com isso, ele vai repetindo esse processo até não entrar nessa condição e simplesmente adicionar o número na pilha. A ideia do não ter volta na escolha ocorre neste ponto, pois um número que foi removido da pilha, nunca mais vai voltar e com isso, ele chega ao final do função, que possui uma condição para garantir que todos os números pedidos foram removidos, e para cobrir alguns casos específicos, caso tenha sobrado número para serem removidos, ele simplesmente remove os números mais à direita da pilha, isso depois de ter percorrido o número completo e por fim manda a pilha para resultado e retorna para a main. Aqui estão algumas imagens apara representar isso que foi dito.