MC621 - Desafios de Programação

Relatório para o placar de 23/11

João Pedro Martins (176117)

Problema J - Train Passengers

O problema consiste basicamente na manutenção do estado de lotação do trem em questão, dado que ocorrem 3 eventos em cada iteração da entrada: a saída de passageiros do trem, a entrada de passageiros no trem e a permanência de passageiros devido ao trem estar lotado. Para isso, utilizei a variável *passenger_count*, inicializada com 0, que vai fazer o track de quantos passageiros estão no trem em um dado momento. A primeira linha da entrada informa a capacidade máxima *C* do trem e o número de estações em que o trem para, i.e., o número de vezes que os 3 eventos citados vão ocorrer. A ideia é verificar a consistência dos dados informados na entrada, printando *possible* ou *impossible,* de acordo: não pode sair mais passageiros do que o trem possui, não pode entrar mais passageiros do que a capacidade permite e não pode ficar passageiros esperando se o trem ainda tem lugar. Isso é equivalente a pedir as invariantes: $0 <= passenger\_count <= C$ e $n\_stay > 0 \Leftrightarrow passenger\_count = C$, onde *n_stay* é o número de passageiros que permaneceram na estação segundo a entrada. Por fim, o problema pede que $passenger\_count = 0$ após a última linha de entrada ser processada.

Para resolver o problema, então, o algoritmo inicializa *passenger_count* e vai atualizando essa variável a cada linha da entrada de acordo com os números de saída, entrada e permanência de passageiros, sempre verificando a manutenção das invariantes mencionadas anteriormente. Se em qualquer ponto uma das invariantes é quebrada, o programa imprime *impossible* e encerra. Caso o contrário, após processar a última linha da entrada o programa verifica se $passenger\_count = 0$. Se for, imprime *possible* e encerra. Caso contrário imprime *impossible* e encerra.

Problema D - Flower Garden

O problema pede que, dado *N* flores espalhadas num plano e um walker *W* que pode percorrer no máximo *D* unidades de distância, se imprima o número primo máximo de flores que *W* consegue visitar na sequência definida pela entrada, ou 0 se ele não conseguir visitar pelo menos 2 flores. O problema, então, é equivalente ao seguinte problema: dado *N* pontos espalhados em um plano e um valor *D*, imprima *M* que é o número primo de pontos da subsequência máxima da sequência $\{p_1, p_2, \ldots, p_{N-1}, p_N\}$ dada pela entrada tal que $dist(p_1, origem) + \sum_{k=2}^{M} dist(p_k, p_{k+1}) <= D$ e $M <= N$, ou imprima 0 se $M < 2$.

Para isso, é necessário inicializar *M* com 0 e decrementar $dist(p_1, origem)$ de *D* e verificar se a distância máxima permitida já foi violada, i.e., verificar se *D < 0*; se foi, a solução do problema é 0 pois $dist(p_1, origem) > D \rightarrow dist(p_1, origem) + \sum_{k=2}^{M} dist(p_k, p_{k+1}) > D$ já que $\sum_{k=2}^{M} dist(p_k, p_{k+1}) > 0$, pois é uma soma de distâncias. Caso contrário, os próximos pontos devem ser processados. Para isso, é suficiente receber cada ponto p_{k+1} da entrada e decrementar $dist(p_{k+1}, p_k)$ de *D* e incrementar *M* por 1 até que os pontos se acabem ou *D < 0*. Após uma dessas condições ser satisfeita, *M* será o número máximo de pontos que satisfaz as condições mencionadas anteriormente, mas *M* não é necessariamente primo. Porém, o máximo primo que atende as condições é o primeiro primo menor que *M.* Logo, basta decrementar *M* enquanto *M* não for primo e for maior que 0. Após isso, *M* será a solução do problema.