Aluno : Rodrigo Alves de Almeida
Laboratório 4 - Subset Sum Problem

1) **Descrição da implementação:**

- **PD:**

Meu algoritmo foi baseado no algoritmo de Bellman otimizado apresentado no livro 'Knapsack Problems' (pág. 75).

Houve uma alteração na estrutura do vetor R: em vez de ser um vetor de *longs* contendo os possíveis valores das somas dos itens, passou a ser um vetor de booleanas de tamanho *value*+1 em que cada índice do vetor corresponde a uma booleana indicando se aquele índice pode ou não ser uma soma dos itens.

Essa alteração pode implicar na utilização de mais memória por esse vetor mas agiliza a busca de valores nele, que passa de $\Theta(n)$ para $\Theta(1)$, sendo n o tamanho de R (além disso, o aumento de tamanho do vetor pode ser irrelevante, visto que variáveis *bool* ocupam 1 byte enquanto variáveis *long* ocupam 8 bytes) .

Assim, para cada iteração dos valores em *input*, os valores de R são somados ao valor da iteração e, se não excederem *value* nem forem repetidos, são acionados em R e o índice do último valor somado é adicionado à posição correspondente no vetor X.

Por fim, basta verificar se o valor de *value* está presente no vetor R e, se estiver, preencher o *output* de acordo com um *backtrace* no vetor X.

- **BB:**

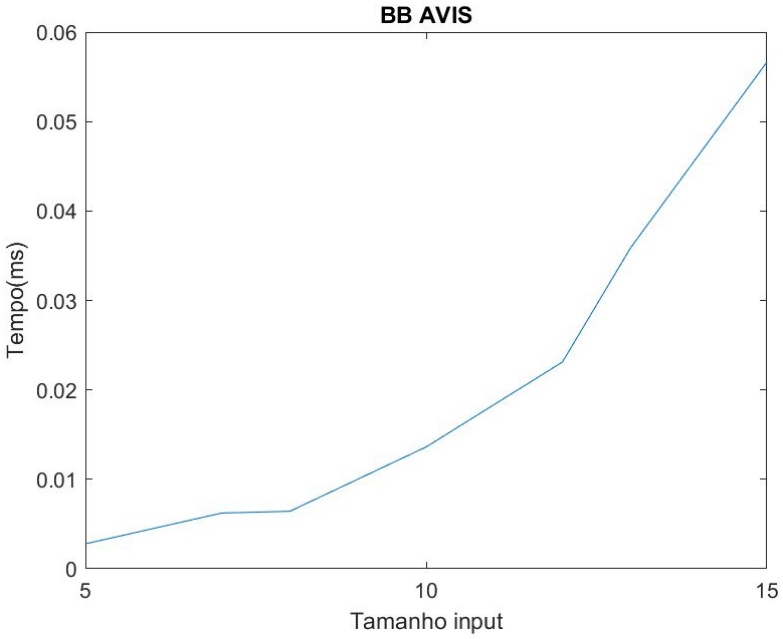
Aqui, foi utilizado um algoritmo *Meet-in-Middle*, onde o vetor de *inputs* é dividido no meio e, para cada metade, são obtidas todas as possíveis somas e caminhos dessas somas pela função fillSubVector().

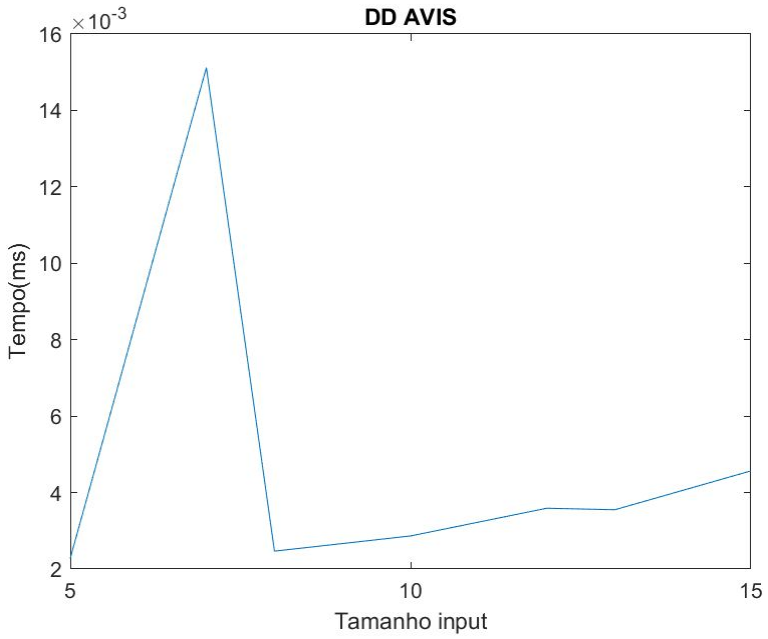
Essa função realiza todas as 2^n possíveis somas em um subvetor (sendo n o tamanho desse subvetor), verifica se as somas não são repetidas nem excedem *value*, e armazena a soma num vetor de booleanas (mesma ideia do vetor R do PD). Além disso, os valores que compõe cada soma são armazenados em um vetor de *longs*.

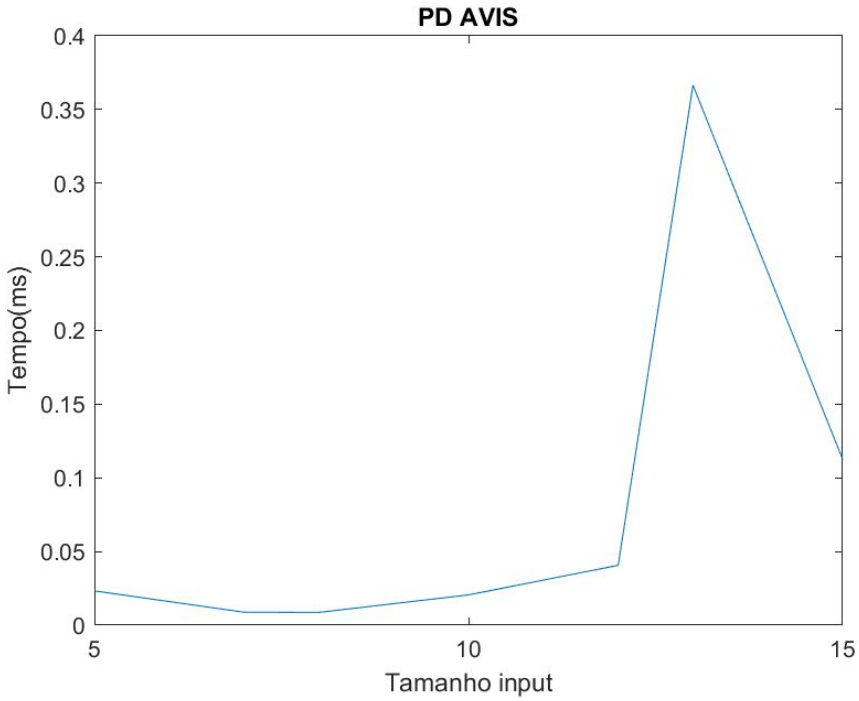
Por fim, para cada soma possível em uma metade, é verificado se o complemento em relação a *value* está presente na outra metade. Se estiver, *output* é preenchido de acordo com a concatenação dos vetores de caminho das duas metades.

2) Comparação dos resultados

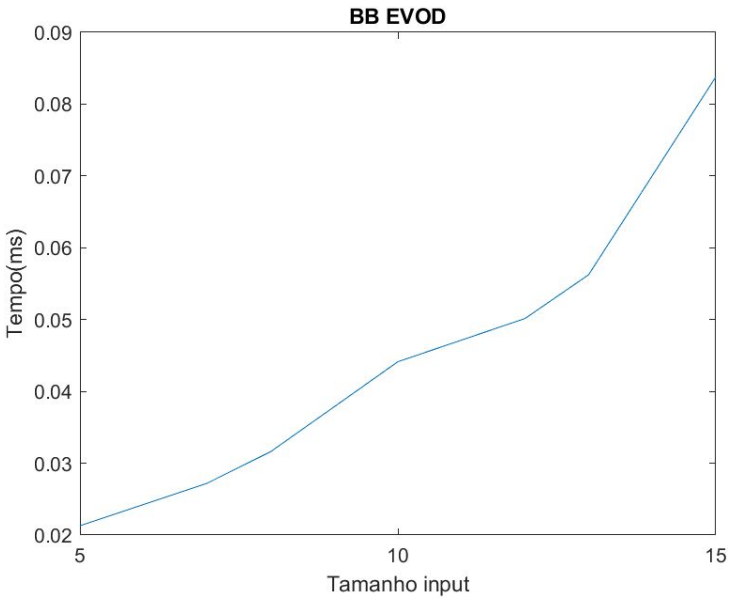
Para o gerador AVIS:

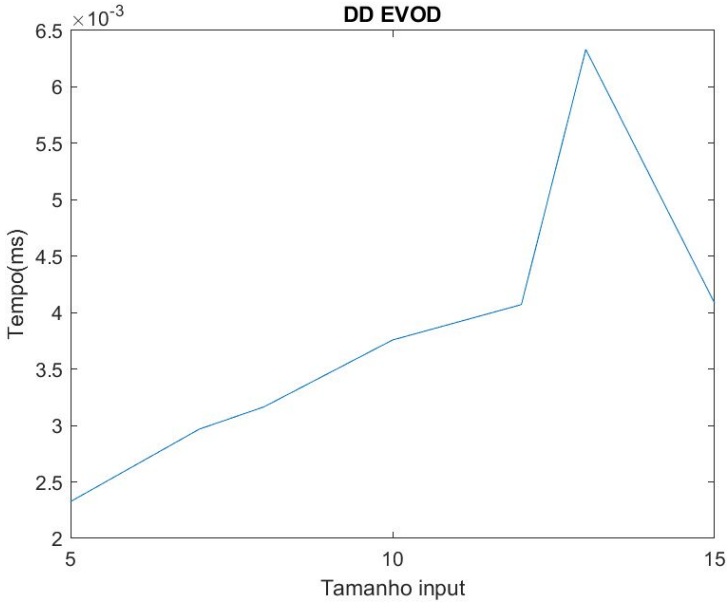


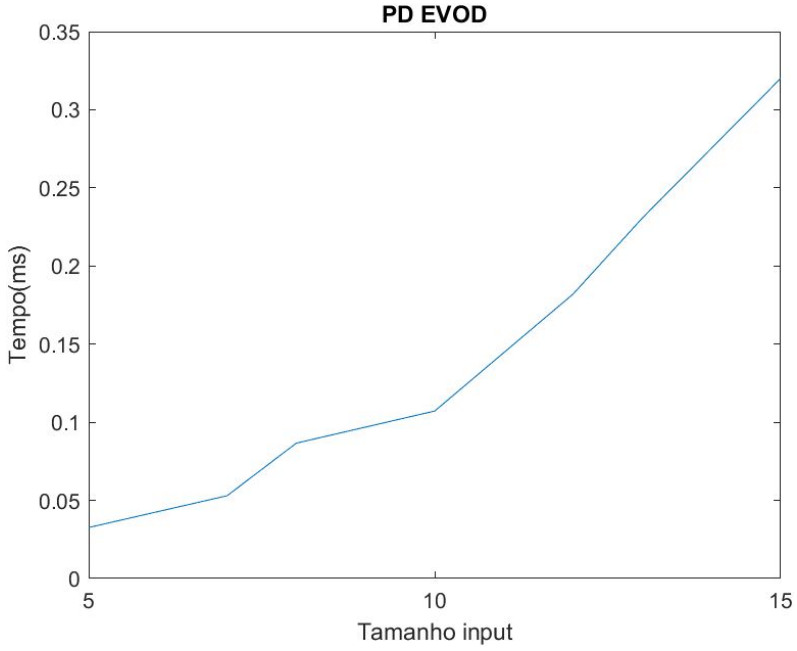




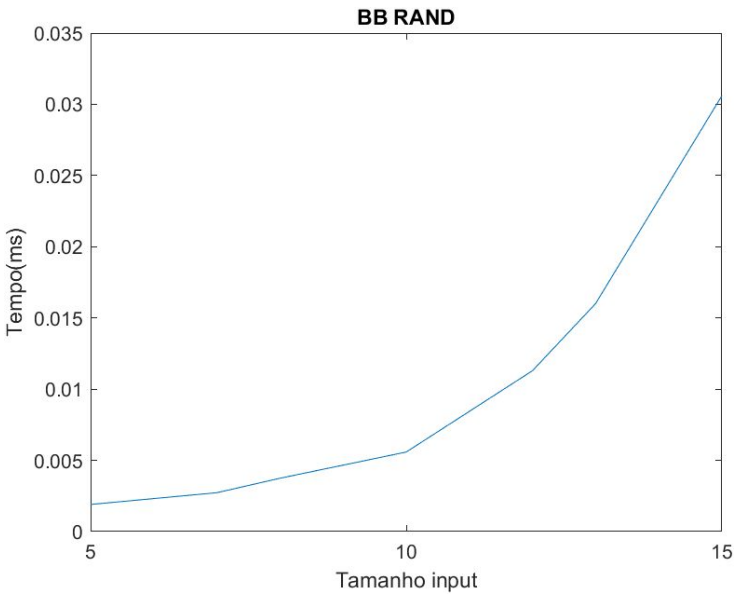
Para o gerador EVOD:

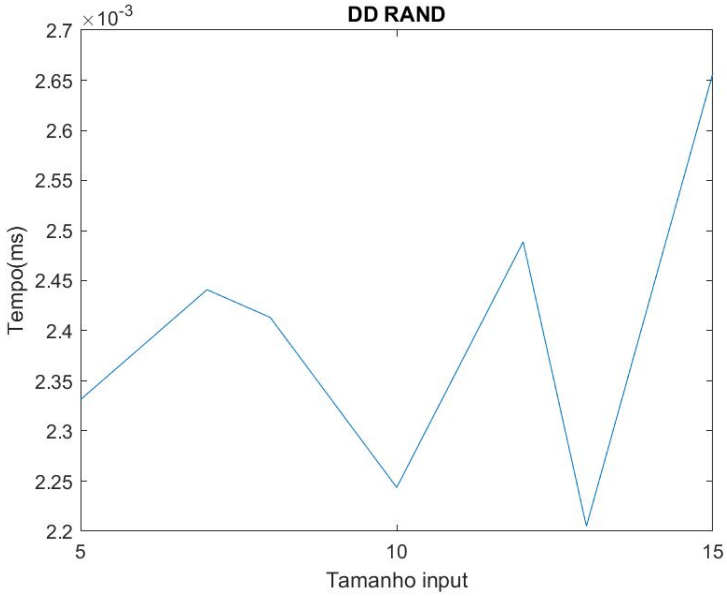


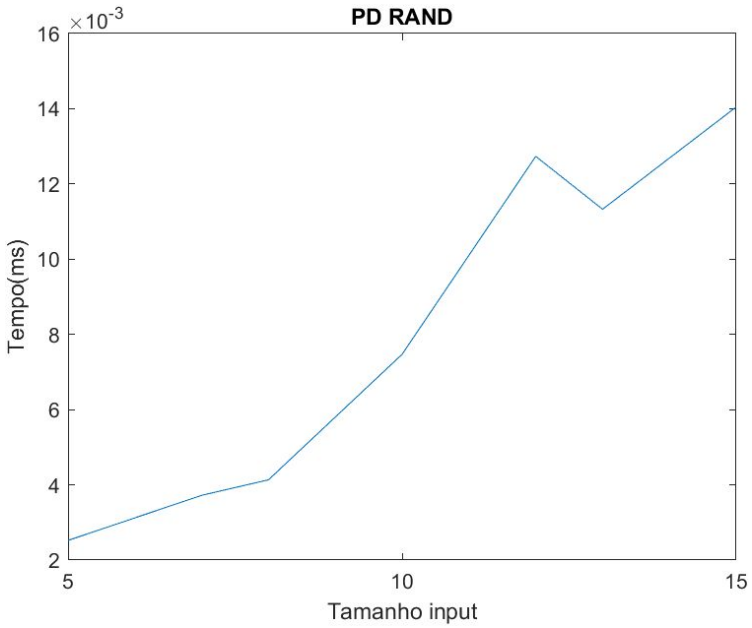




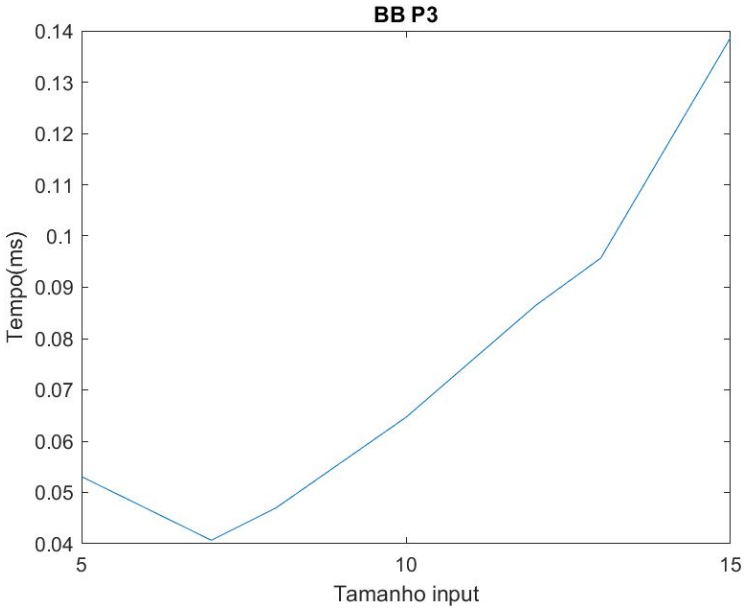
Para o gerador RAND:

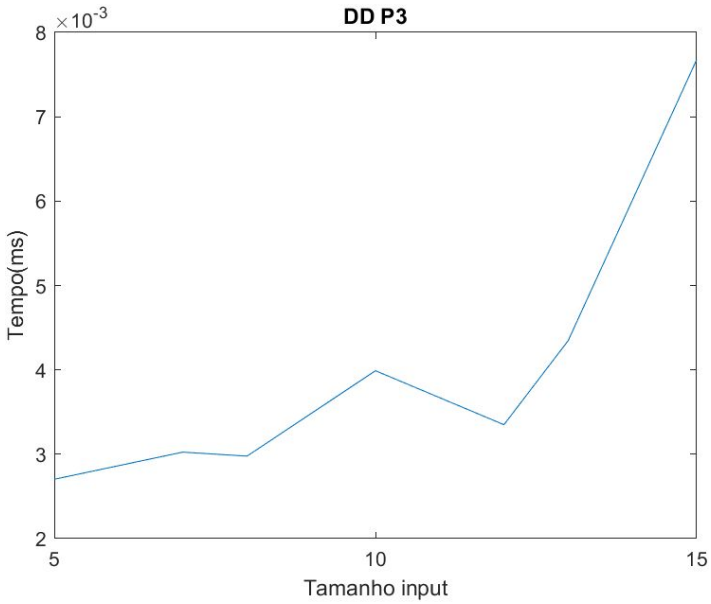


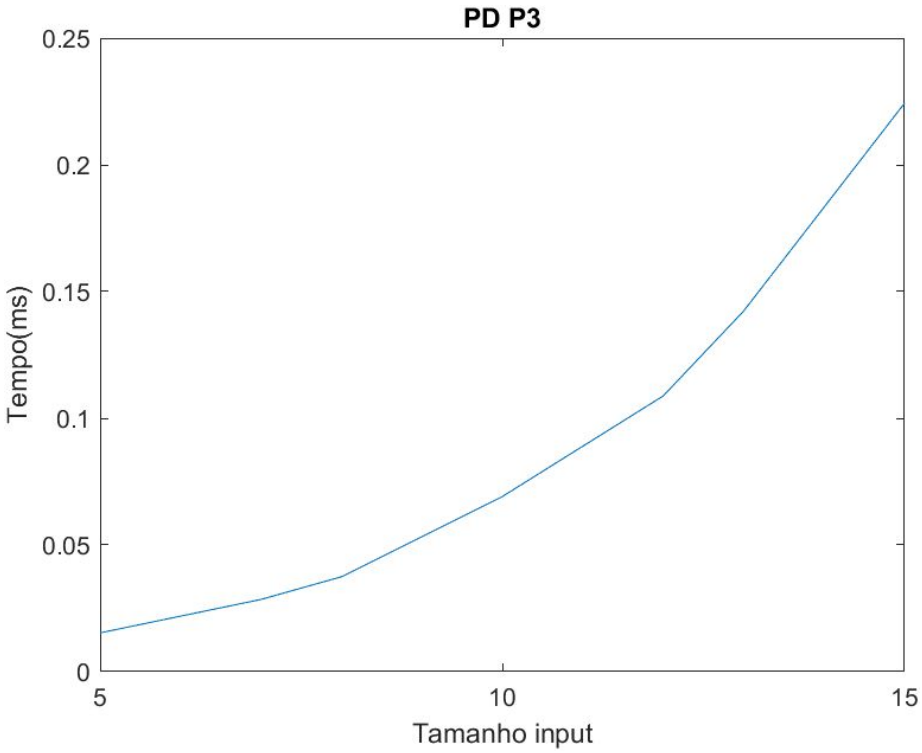




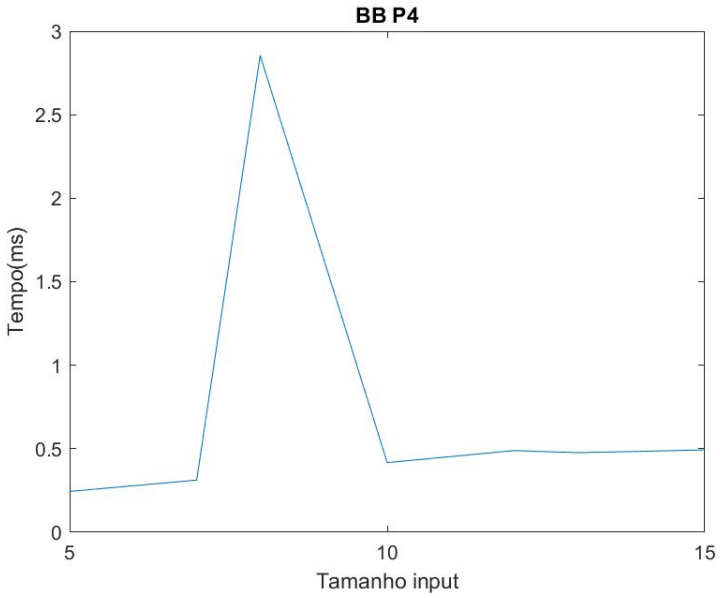
Para o gerador P3:

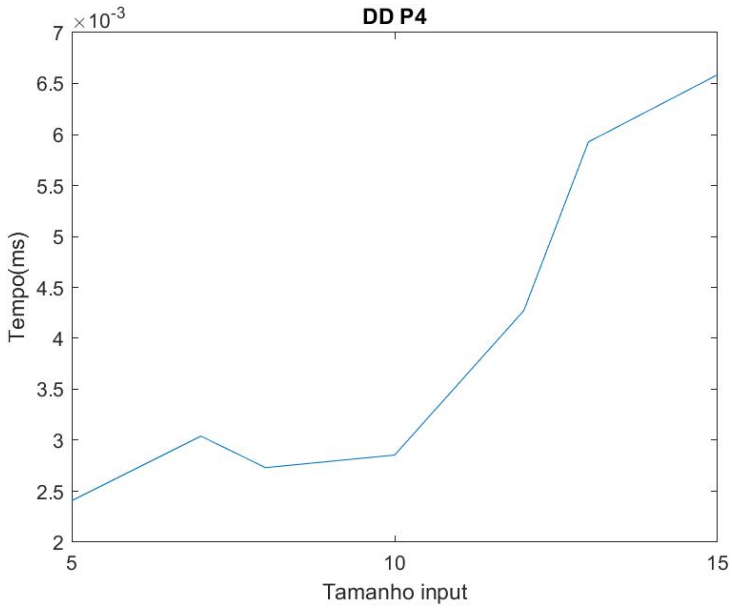


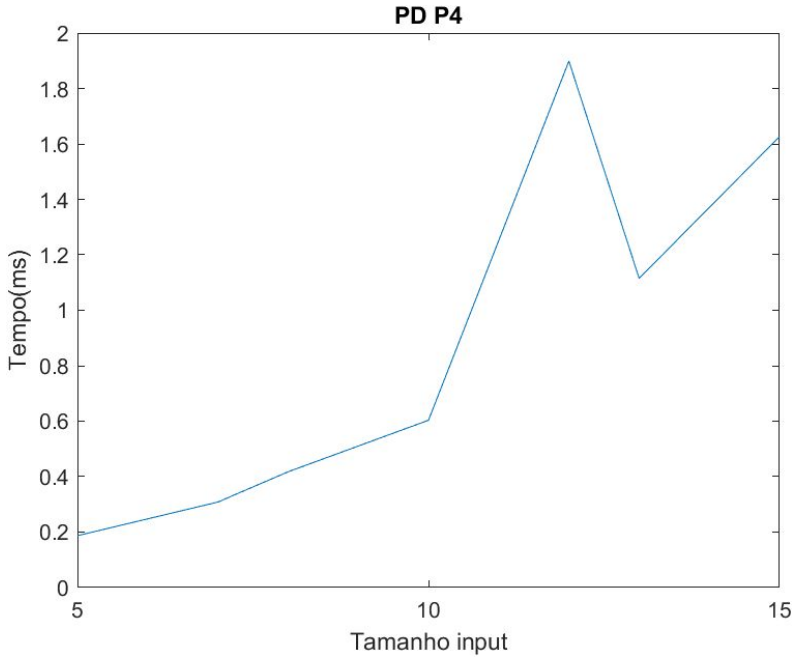




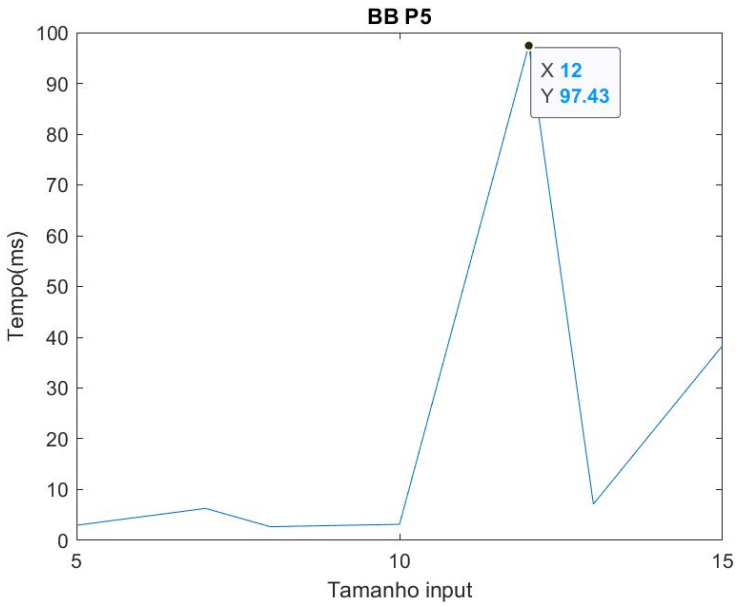
Para o gerador P4:

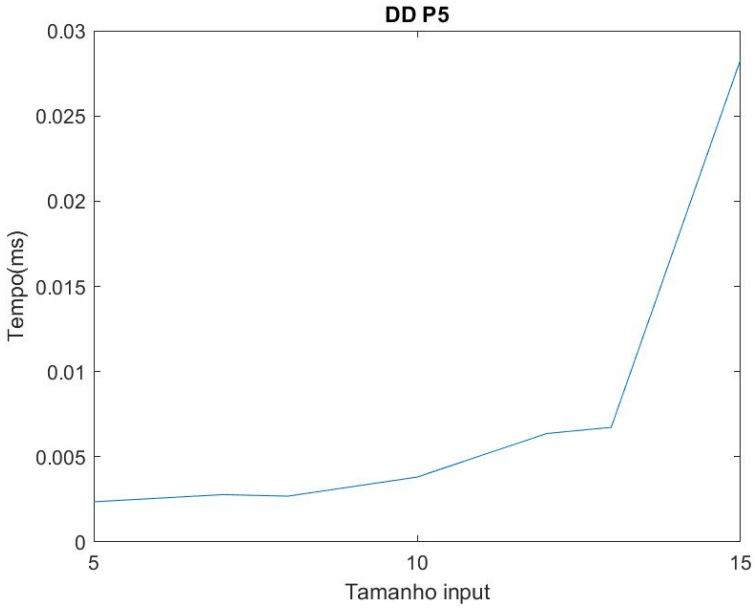


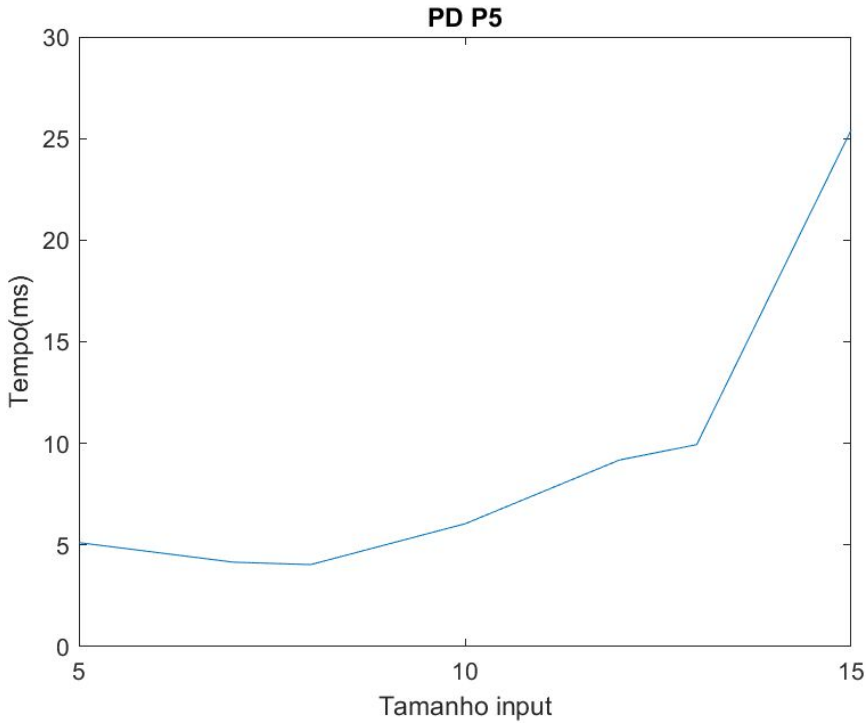




Para o gerador P5:







De acordo com os gráficos obtidos, comparando os algoritmos, observamos que:

- Os algoritmos de PD (Bellman) e BB (Meet in Middle) apresentam na média mesma ordem de tempo de funcionamento. Isso é esperado teoricamente, pois ambos algoritmos possuem mesma ordem de grandeza de tempo.

Comparando os geradores de instância:

- À medida que o valor dos pesos aumenta, o tempo de execução também aumenta de forma pseudo-polinomial.