Instituto Tecnológico de Aeronáutica
CES-12 - Algoritmos e Estruturas de dados II
Aluno: Bernardo Hoffmann da Silva

Relatório do Laboratório 4 - parte 2: Paradigmas da programação

1. Descrição da implementação.

O algoritmo utilizado para a implementação A2 foi o Branch and Bound recursivo. Ele se trata em procurar as soluções do problema da mochila resolvido colocando cada input em um galho, sendo que o input de maior valor está no galho superior. Caso esse nó seja utilizado na solução, segue para o ramo da esquerda dele. Caso contrário, segue-se para o ramo da direita. Assim, testa-se até chegar às folhas, as quais podem também ou não ser parte da solução.

A árvore em si, nesse caso, não foi implementada, mas a recursão foi feita de forma análoga, em que inicia-se no input máximo e desce para a "esquerda", subtraindo-se da capacidade da mochila o valor do nó atual e para a "direita" passando a capacidade atual da mochila.

Para as regras de poda, fez-se as seguintes considerações: (1) caso o nó seja maior do que a capacidade atual da mochila, não é testada a condição em que esse nó faz parte da solução. (2) É testada a condição de não utilizar o nó somente se o teste da solução com esse nó retornar falsa. (3) Caso o nó analisado seja uma folha, checa-se se atingiu-se o objetivo de valor total da mochila com ou sem esse nó. Caso não, esse ramo é descartado.

Percebe-se a semelhança dessa implementação recursiva com *backtracking*, porém, as regras de poda permitem uma solução menos bruta.

O algoritmo em programação dinâmica (PD) implementado faz uso de uma matriz de tamanhos (input.size() + 1) e (value + 1). Essa matriz funciona guardando, em cada posição, uma variável booleana que representa se para aquele tamanho de mochila específico e para aquela lista de objetos específica, de cima para baixo, da esquerda para a direita, pode-se ter uma solução que satisfaça o problema fornecido.

Para montar a matriz, verifica o caso do objeto anterior ou o caso em que, para a mesma lista de objetos, um tamanho de mochila exatamente menor satisfaz. Após a montagem da matriz, a verificação do problema se dá pela posição inferior direita. Isso é feito pois o tamanho da mochila desejado é tal que é a última coluna da matriz. E se houver solução, o set de objetos que contém esse último certamente contém o subset que é a solução em si.

2. Análise dos algoritmos.

A fim de comparar os algoritmos implementados, pode-se traçar os gráficos de tempo de execução por tamanho da entrada. Eles estão presentes nas figuras 1, 2, 3, 4, 5 e 6.

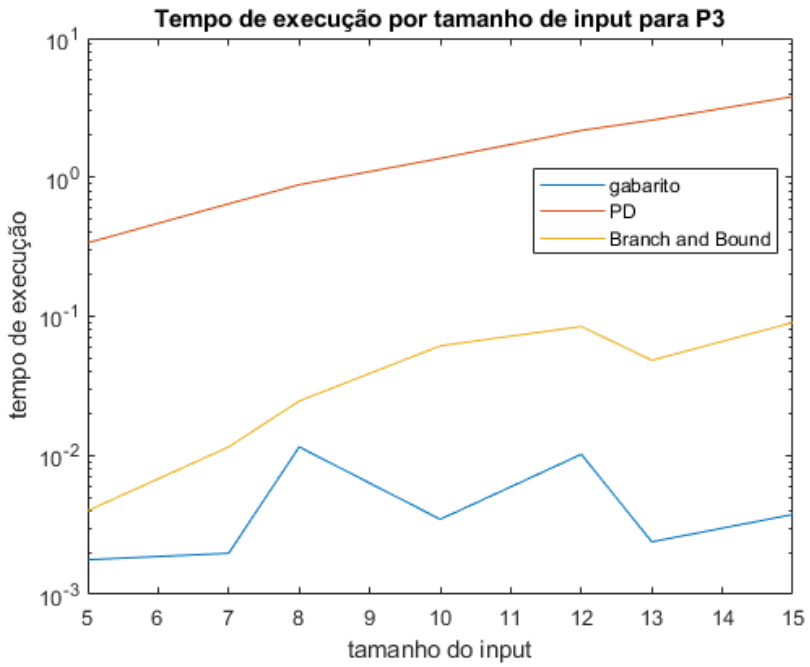

Figura 1: Comparação do tempo de execução para diferentes tamanhos de entrada para P3

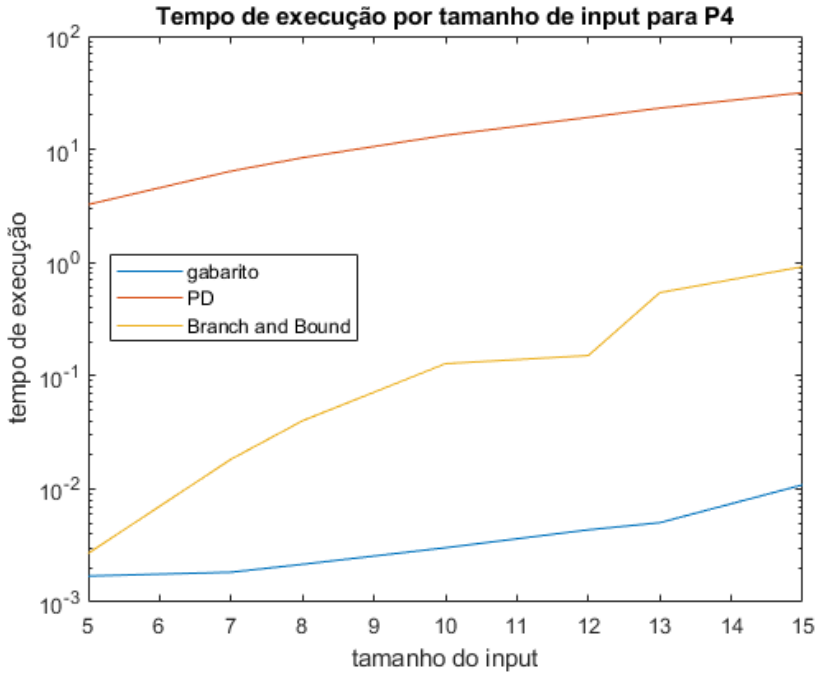

Figura 2: Comparação do tempo de execução para diferentes tamanhos de entrada para P4

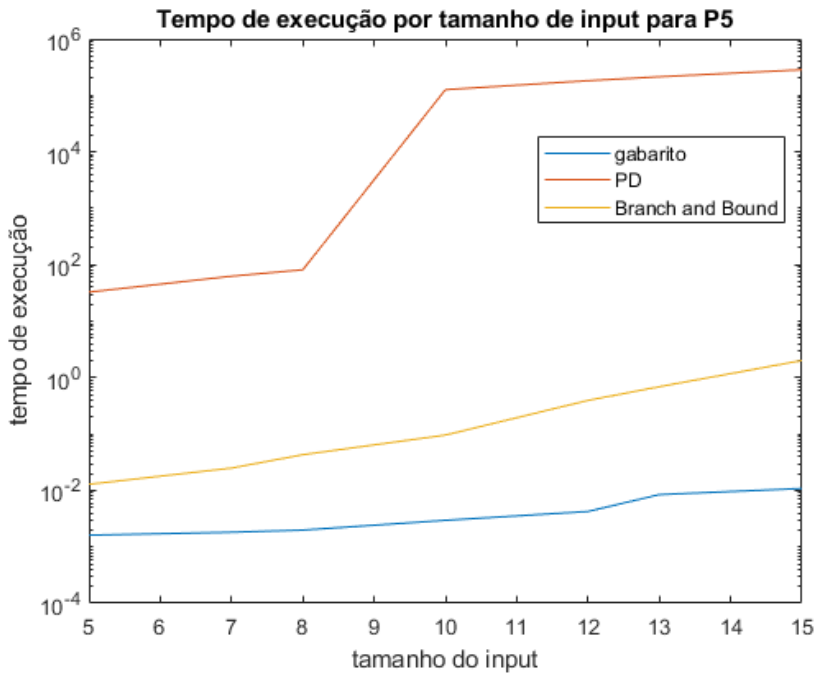

Figura 3: Comparação do tempo de execução para diferentes tamanhos de entrada para P5

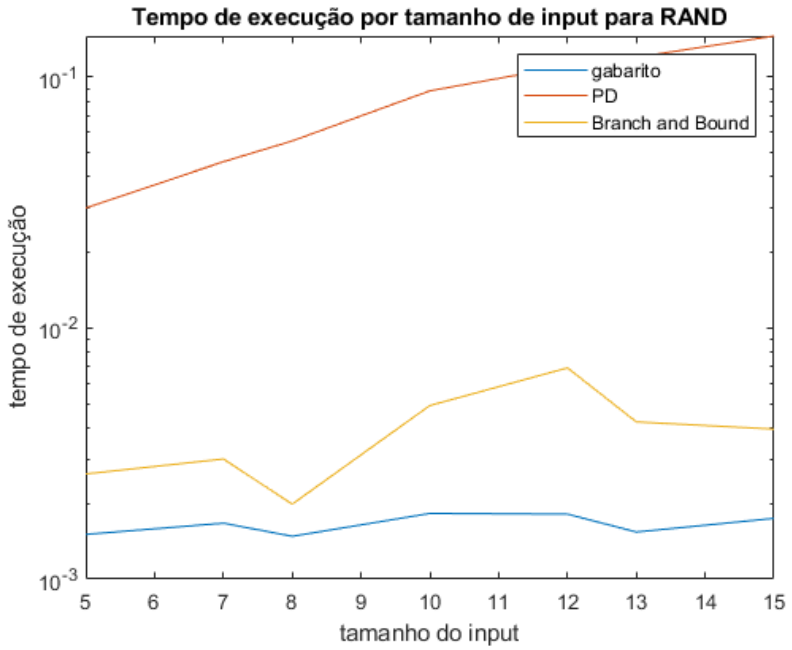

Figura 4: Comparação do tempo de execução para diferentes tamanhos de entrada para RAND.

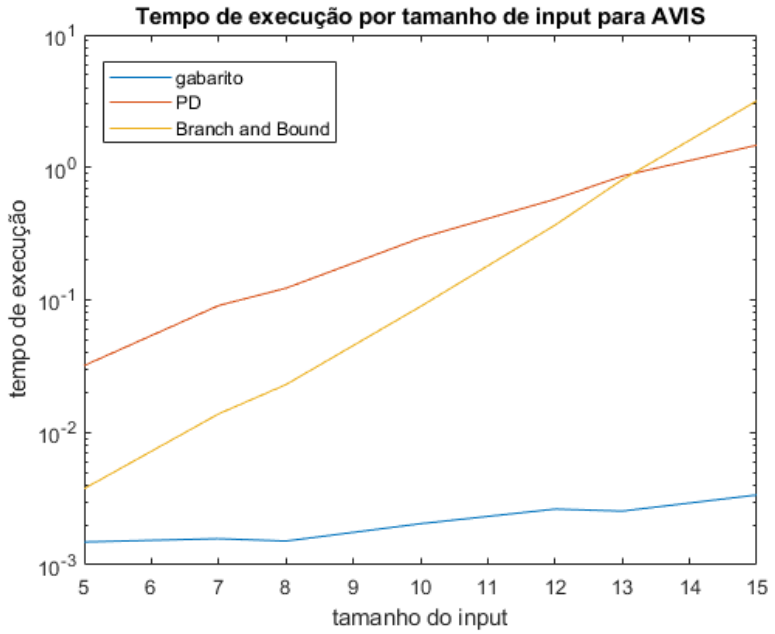

Figura 5: Comparação do tempo de execução para diferentes tamanhos de entrada para AVIS.

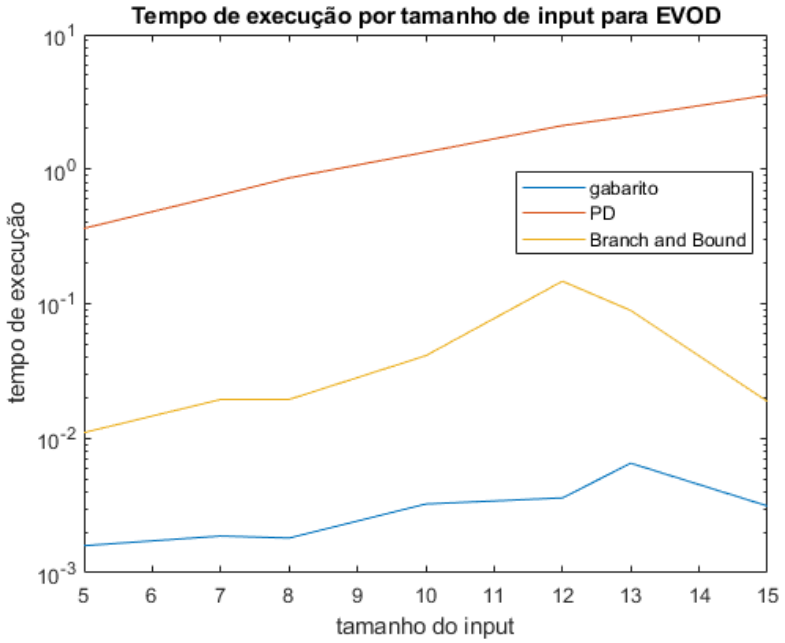

Figura 6: Comparação do tempo de execução para diferentes tamanhos de entrada para Even Odd

Realizando comparações entre os algoritmos, percebe-se que o aumento do tamanho da mochila em P3, P4 e P5 provocou aumento considerável no método da programação dinâmica, enquanto o Branch and Bound e o gabarito não apresentaram crescimentos tão acentuados. Isso se deve ao fato de que o aumento exponencial da mochila necessita de um preenchimento de matriz muito maior por esse algoritmo, o que o torna lento para casos muito grandes. Branch and bound não percebe muito esse crescimento pois sua execução

depende mais fortemente do tamanho do vetor de inputs, pois isso o fará testar muito mais ramos, de forma geral.

Para os valores aleatórios RAND, percebe-se que o Branch and Bound se torna uma opção rápida, em geral, se aproximando do gabarito. A programação dinâmica também apresenta tempo de execução relativamente rápido, por esses valores aleatórios são da mesma ordem.

Para AVIS, percebe-se que não há nenhum teste com resposta positiva. Por esse motivo, o Branch and Bound se torna o pior algoritmo para inputs grandes. A falta de respostas positivas faz com que esse algoritmo procure em todos os ramos da árvore até chegar nas folhas. Isso o força a testar muitos casos e aumentar o tempo de execução. A programação dinâmica não é muito afetada por essas entradas, apesar de ter um crescimento de tempo grande conforme aumentam-se os inputs.

Para entradas EVOD, nota-se que se segue o padrão em que a programação dinâmica é o algoritmo mais lento, seguido do Branch and Bound e o gabarito. Comparados a entradas de ordens maiores, as Even Odd geram resultados mais rápidos em geral.