UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL
Faculdade de Computação

Implementação Algorítmica

Atividade 4 — Problema da mochila

1 Descrição

Um ladrão está roubando uma loja e encontra n objetos. Nessa loja, o i-ésimo objeto tem valor v_i e peso w_i, onde v_i e w_i são números inteiros. O ladrão quer roubar a carga mais valiosa possível, mas ele pode carregar no máximo W quilos em sua mochila, para algum número inteiro W. Quais objetos ele deve levar? Esse é um problema de otimização clássico e é chamado de *problema da mochila binária (ou 0-1)*, porque cada objeto pode ser levado ou deixado para trás.

Um variação desse problema é aquela em que o ladrão pode levar frações dos produtos, ao invés de ter de fazer uma escolha binária para cada objeto. Esse problema, por sua vez, é chamado de *problema da mochila fracionária*, porque o ladrão pode levar uma fração de cada material/produto.

Ambos problemas exibem a propriedade da subestrutura ótima, o que permite que a estratégia da programação dinâmica e o método guloso possam ser usados. Mas apesar dos problemas serem similares, o problema da mochila fracionária é solucionado pela estratégia gulosa, enquanto que o problema da mochila binária é solucionado por programação dinâmica.

No problema da mochila fracionária, o ladrão deve primeiro computar o valor por quilo v_i/w_i de cada material. Obedecendo a escolha gulosa, o ladrão começa levando tanto quanto possível do material com maior valor por quilo. Se termina o suprimento desse material e o ladrão ainda pode roubar mais, ele leva tanto quanto possível do material com o próximo maior valor por quilo e assim por diante, até que ele não possa mais levar nenhum material. Assim, pela ordenação dos materiais pelo valor por quilo, o algoritmo tem tempo de execução $O(n \lg n)$. O algoritmo que implementa a estratégia gulosa para o problema da mochila fracionária é mostrado a seguir.

```
FractionalKnapsack(v, w, W)
    ordene v e w por v_i/w_i
    para i ← 1 até v.length
        x_i ← 0
    i ← 1
    enquanto w_i ≤ W
        x_i ← 1
        W ← W − w_i
        i ← i + 1
    se W > 0
        x_i ← W/w_i
    devolva x
```

Por outro lado, no problema da mochila binária temos de usar a estratégia da programação dinâmica. Suponha que um objeto i de peso w_i faz parte da solução. Como vimos em sala, devemos então resolver o subproblema contendo $n-1$ objetos com peso máximo $W - w_i$ (subestrutura ótima). Note que precisamos resolver o problema da mochila binária para todos os objetos e pesos possíveis menores que W. Construimos uma matriz K de dimensões $(n+1)\times(W+1)$ de valores em que as linhas são indexadas pelos objetos e as colunas são indexadas pelos pesos.

Para uma linha i e coluna j, precisamos decidir o que é mais vantajoso: incluir o item i na mochila, comparando o valor total da mochila incluindo os objetos de 1 a $i-1$ com o peso máximo j *ou* o valor da inclusão dos objetos de 1 a $i-1$ com peso máximo $j-w_i$, mais o objeto i. A solução, isto é, o valor da mochila mais valiosa, está armazenado na posição n, W da matriz K. Assim, a recorrência para a solução é obtida pela seguinte fórmula, para todo para i, j, com $0 \le i \le n$ e $0 \le j \le W$:

$$K[i,j] = \begin{cases} 0\,, & \text{se } i = 0 \text{ ou } j = 0\,, \\ K[i-1,j]\,, & \text{se } j < w_i\,, \\ \max(K[i-1,j], K[i-1,j-w_i]+v_i)\,, & \text{caso contrário}\,. \end{cases}$$

Um algoritmo de programação dinâmica usando a estratégia *bottom-up* e que implementa essa fórmula é descrito a seguir.

```
BINARY-KNAPSACK (v, w, W)
    n ← v.length
    seja K uma matriz de dimensões (n + 1) × (W + 1)
    para i ← 0 até n
        K[i, 0] ← 0
    para j ← 1 até W
        K[0, j] ← 0
    para i ← 1 até n
        para j ← 1 até W
            se j < w_i
                K[i, j] ← K[i − 1, j]
            senão
                K[i, j] ← max(K[i − 1, j], K[i − 1, j − w_i] + v_i)
    devolva K
```

O tempo de execução do algoritmo BINARY-KNAPSACK é $\Theta(nW)$. Note que isso significa que esse algoritmo tem tempo de execução **pseudo-**polinomial.

Nesta atividade, você tem de implementar os dois algoritmos para solução do problema da mochila.

2 Programa, entrada e saída

Você deve implementar os algoritmos conforme a descrição da seção 1, desenvolvendo um ou mais programas.

Você deve construir conjuntos de dados de entrada (pseudo)aleatórios da seguinte forma. Primeiro, determine um valor da capacidade da mochila W. Por exemplo, você pode supor que as capacidades variam como $W = 10, 15, 20, 25, \ldots, 1000$. Para cada valor W de capacidade de uma mochila, você deve determinar um número de itens. Por exemplo, se a capacidade da mochila é $W = 10$ quilos, você deve gerar quantidades de itens n variando de 5 a 20, isto é, $n = 5, 10, 15, 20$. Você pode determinar quantidades razoáveis de itens n para uma mochila de capacidade W. De modo geral, você pode fazer n variar de $W/2$ a $2 \cdot W$, por exemplo. Depois, dado que você tem uma capacidade W e um número de itens n, você deve sortear n números inteiros que representam os valores v_i dos itens da loja e n números inteiros que representam os pesos w_i desses itens. Também é necessário ser razoável na determinação desses valores. Determine um valor máximo inteiro de um item da loja, por exemplo \$200, e sorteie valores no intervalo de 1 a 200 para os valores v_i desses n itens. Além disso, faça com que os pesos sorteados w_i sejam menores ou iguais à capacidade da mochila W, isto é, estejam no intervalo de 1 a W.

Para cada caso de teste, ou seja, uma capacidade da mochila W, um número de itens da loja n, valores v_i e pesos w_i dos n itens, a saída consiste de uma primeira linha contendo os rótulos da capacidade da mochila W, a quantidade de itens n, o valor da solução da mochila fracionária (**`frac`**), o valor da solução da mochila binária (**`bin`**) e o peso dos itens escolhidos na solução da mochila binária (**`w`**), o tempo de execução do algoritmo da mochila fracionária (**`tf`**) e o tempo de execução do algoritmo da mochila binária (**`tb`**) para esse caso de teste.

2.1 Exemplo de entrada e saída

Um pequeno exemplo, para casos de teste com $10 \leq W \leq 25$, é mostrado a seguir. Os tempos de execução dos algoritmos são mostrados em segundos.

```
  W    n    frac  bin   w       tf       tb
-------------------------------------------
 10    5  489.22  474   9 0.000005 0.000003
 10   10  577.88  560   9 0.000004 0.000004
 10   15  558.75  487  10 0.000005 0.000005
 10   20  565.25  537   9 0.000007 0.000007
 15    7  659.38  639  14 0.000002 0.000004
 15   12  379.43  320  14 0.000004 0.000005
 15   17  611.00  611  15 0.000006 0.000006
 15   22  651.00  651  15 0.000007 0.000010
 15   27  653.00  653  15 0.000008 0.000010
 20   10  500.00  491  19 0.000003 0.000006
 20   15  632.53  623  19 0.000005 0.000008
 20   20  744.86  728  20 0.000006 0.000009
 20   25  836.23  780  16 0.000007 0.000010
 20   30  617.86  599  20 0.000009 0.000014
 20   35  951.75  928  19 0.000011 0.000014
 20   40  652.80  640  20 0.000011 0.000016

(continua...)
```

25	12	526.00	486	24	0.000004	0.000007
25	17	575.00	575	25	0.000005	0.000009
25	22	797.78	746	24	0.000006	0.000012
25	27	1047.43	1041	25	0.000008	0.000015
25	32	745.00	745	25	0.000009	0.000016
25	37	910.38	900	25	0.000010	0.000018
25	42	1210.00	1169	23	0.000012	0.000022
25	47	1151.00	1111	22	0.000014	0.000022
30	15	583.80	549	27	0.000004	0.000010
30	20	459.00	459	30	0.000005	0.000012
30	25	872.38	854	29	0.000007	0.000016
30	30	838.64	803	28	0.000009	0.000018
30	35	866.00	788	30	0.000010	0.000020
30	40	903.50	891	29	0.000011	0.000022
30	45	987.44	985	30	0.000012	0.000025
30	50	762.00	724	30	0.000014	0.000027
30	55	1207.67	1187	30	0.000017	0.000029
30	60	1199.43	1199	30	0.000018	0.000036
35	17	421.21	372	32	0.000005	0.000012
35	22	553.25	544	34	0.000007	0.000014
35	27	876.75	821	35	0.000007	0.000018
35	32	815.14	806	35	0.000009	0.000022
35	37	1134.00	1128	35	0.000011	0.000026
35	42	627.85	589	33	0.000013	0.000024
35	47	875.00	860	34	0.000029	0.000029
35	52	890.50	889	35	0.000015	0.000064
35	57	1548.38	1537	35	0.000007	0.000015
35	62	958.08	956	35	0.000007	0.000014
35	67	778.73	760	35	0.000007	0.000015
40	20	577.28	533	40	0.000002	0.000006
40	25	980.50	905	36	0.000002	0.000007
40	30	848.89	794	39	0.000003	0.000008
40	35	1124.08	1106	40	0.000003	0.000010
40	40	644.00	644	40	0.000005	0.000010
40	45	1115.89	1084	40	0.000005	0.000012
40	50	1170.75	1147	39	0.000004	0.000013
40	55	889.47	874	40	0.000006	0.000013
40	60	1092.88	1075	40	0.000006	0.000017
40	65	1597.33	1555	40	0.000008	0.000018
40	70	1183.89	1144	40	0.000010	0.000018
40	75	1266.36	1257	40	0.000008	0.000020
40	80	1487.00	1487	40	0.000009	0.000021
45	22	814.76	808	44	0.000002	0.000008
45	27	710.26	669	41	0.000003	0.000009
45	32	855.60	772	44	0.000004	0.000010
45	37	852.24	802	45	0.000005	0.000011
45	42	678.00	678	45	0.000004	0.000012
45	47	905.00	893	44	0.000005	0.000014
45	52	981.75	960	45	0.000006	0.000015
45	57	1306.40	1267	42	0.000007	0.000016
45	62	1153.59	1116	42	0.000007	0.000018
45	67	1344.40	1330	45	0.000008	0.000020
45	72	1117.83	1090	45	0.000007	0.000021
45	77	1155.75	1146	44	0.000007	0.000022
45	82	1370.89	1367	45	0.000009	0.000024
45	87	1072.00	1072	45	0.000010	0.000024

3 Entrega

Instruções para entrega da sua atividade:

1. **O que entregar?**

   O arquivo a ser entregue deve conter o seguinte:

   - programa(s) desenvolvidos (e um arquivo **`Makefile`**, se for o caso),
   - tabela descrita na seção 2.1, e
   - pelo menos dois gráficos no formato (**`.pdf`**) gerados a partir da tabela de saída: um comparando os resultados das soluções dos dois problemas para um caso de teste (como no exemplo do mostrado na Figura 1) e outro com os valores dos tempos de execução dos algoritmos. O gráfico foi construído usando o software **`gnuplot`**.

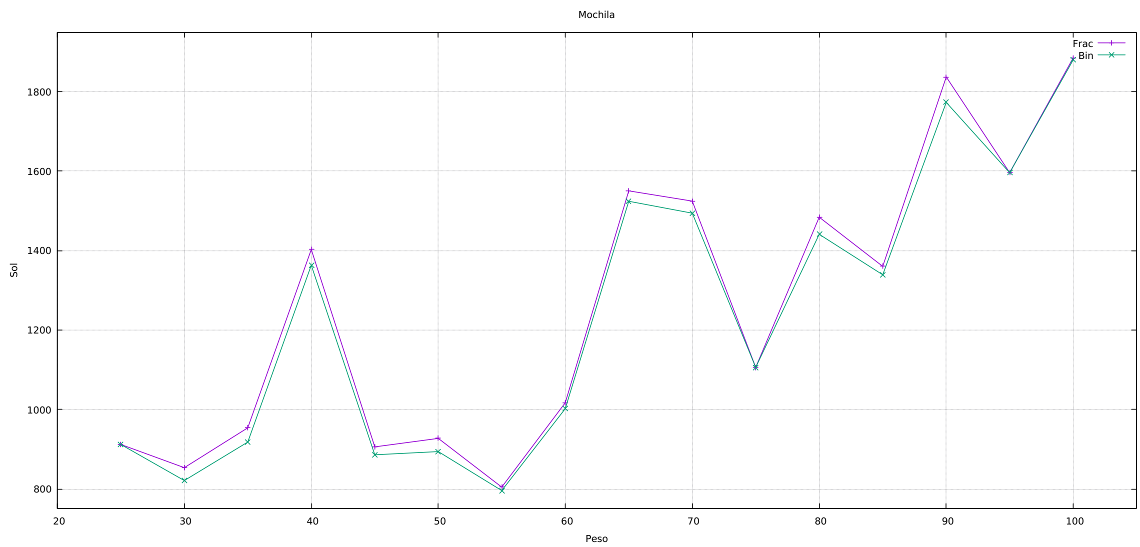

Figura 1: Execução dos algoritmos para o caso de teste com $W = 50$ e $n = 25, 30, \ldots, 95, 100$.

   Compacte todos esses arquivos com o compactador de sua preferência e entregue um único arquivo (com extensão **`.tgz`**, **`.bz2`**, **`.zip`**, **`.rar`**, ...).

2. **Forma de entrega**

   A entrega será realizada diretamente no Sistema (AVA/UFMS), na disciplina Implementação Algorítmica – T01. Após abrir uma sessão digitando seu *login* e sua senha, vá até a sessão "Atividades" e escolha o tópico "Atividade 4 – Problema da mochila", onde você poderá entregar sua atividade. Um fórum de discussão deste trabalho já se encontra aberto, além de outras informações adicionais. Você pode entregar o trabalho quantas vezes quiser até às

23 horas e 59 minutos do dia **28 de outubro de 2021**. A última versão entregue é aquela que será corrigida. Encerrado o prazo, não serão mais aceitos trabalhos.

3. **Atrasos**

   Trabalhos atrasados não serão aceitos. Não deixe para entregar seu trabalho na última hora. Para prevenir imprevistos como queda de energia, problemas com o sistema, e/ou falha de conexão com a internet, sugerimos que a entrega do trabalho seja feita pelo menos um dia antes do prazo determinado.

4. **Erros**

   Trabalhos com erros de compilação/interpretação receberão nota ZERO. Faça todos os testes necessários para garantir que seu programa está livre de erros de compilação.

5. **Linguagem de programação e arquivo com o(s) programa(s)-fonte**

   Você pode escolher a sua linguagem de programação preferida para implementar esta atividade, mas não use uma linguagem dependende de plataforma como `dotNet`. Adicionalmente, fique atento(a) para que seu(s) arquivo(s) contendo o(s) programa(s)-fonte esteja(m) bem organizado(s). Um programa tem de ser muito bem compreendido por uma pessoa. Verifique se seu programa tem a indentação adequada, se não tem linhas muito longas, se tem variáveis com nomes significativos, entre outros. Não esqueça que um programa bem descrito e bem organizado é a chave de seu sucesso. E não esqueça da documentação de seu programa.

6. **Conduta Ética**

   O trabalho deve ser feito INDIVIDUALMENTE por cada grupo. Cada grupo tem responsabilidade sobre cópias de seu trabalho, mesmo que parciais. Não compartilhe seu programa ou trechos de seu programa. Você pode consultar seu(sua) colega de dupla para esclarecer dúvidas e discutir idéias sobre o trabalho, pode consultar o professor em uma chamada virtual ou no fórum de discussão da disciplina, mas NÃO copie a atividade!

   Trabalhos considerados plagiados terão nota ZERO.