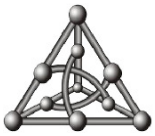


UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL
Faculdade de Computação

Implementação Algorítmica

Atividade 2 — Problema do corte da tora

1 Descrição

Uma empresa compra longas toras de madeira e as corta em pedaços menores para revenda. Os cortes são realizados de tal modo que os pedaços resultantes têm sempre comprimento inteiro. Um corte não tem valor e não é cobrado pela empresa. Para cada $i = 1, 2, \ldots$, a empresa cobra o preço p_i por uma tora de comprimento i.

Como exemplo, suponha que temos uma tora de comprimento 10 metros com os valores dos seus pedaços de 1 a 10 metros dados pela seguinte tabela:

comprimento i	1	2	3	4	5	6	7	8	9	10
preço p_i	1	5	8	9	10	17	17	20	24	30

Dessa forma, temos o seguinte:

> **Problema do corte da tora de madeira**: Dadas uma tora de n metros de comprimento e uma tabela de preços p_i para $i = 1, 2, \ldots, n$, determine o valor máximo de venda r_n que é possível obter pelo corte da tora e venda de seus pedaços.

Um exemplo para uma tora de $n = 4$ metros é mostrado na Figura 1.

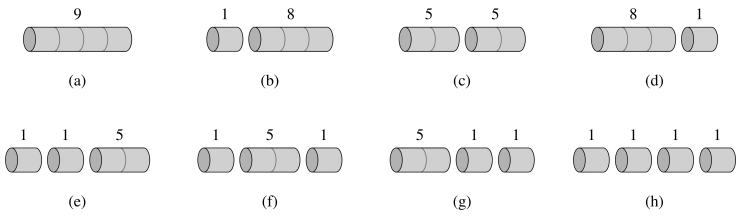


Figura 1: Exemplo com todos os cortes possíveis para uma tora de comprimento $n = 4$ e preços de venda dados pela tabela acima. Observe que o valor máximo de venda é obtido quando cortamos a tora em dois pedaços de 2 metros cada (letra (c)) e obtemos o valor de venda $p_2 + p_2 = 5 + 5 = 10$.

Nesta atividade, você tem de implementar dois algoritmos para o problema do corte das toras de madeira:

- um usando **programação dinâmica** (um dos algoritmos visto em aula - *bottom up* ou *top down with memoization*) e;
- outro usando uma **estratégia gulosa** (*greedy*) descrita a seguir.

Estratégia gulosa para o problema do curte da tora de madeira: Defina a **densidade** de uma tora de comprimento i como p_i/i, isto é, seu valor por metro. A estratégia gulosa para uma tora de comprimento n corta um primeiro pedaço de comprimento i, onde $1 \leq i \leq n$, com densidade máxima. Continua-se então aplicando a estratégia gulosa ao pedaço remascente de comprimento $n - i$.

A estratégia gulosa nem sempre encontra a solução ótima para este problema. De fato, o Exercício 15.1-2 do livro CLRS[1] pede para mostrar um contra-exemplo em que esta estratégia falha ao tentar encontrar uma solução ótima.

2 Programa, entrada e saída

Você deve desenvolver e implementar os algoritmos conforme a descrição da Seção 1.

Considere os seguintes parâmetros para execução dos experimentos, que são fornecidos como entrada:

- **`inc`** é o tamanho inicial da entrada;
- **`fim`** é o tamanho final;
- **`stp`** é o intervalo entre dois tamanhos; e

Você deve construir conjuntos de dados de entrada (pseudo)aleatórios da seguinte forma. Primeiro, um valor do comprimento da tora n. Por exemplo, para **`inc`** $= 10$, **`fim`** $= 100$ e **`stp`** $= 10$ os comprimentos devem ser $n = 10, 20, 30, \ldots, 100$. Para cada valor n do comprimento de uma tora, você deve determinar os preços de venda dos pedaços da tora de comprimentos inteiros de 1 a n. Limite os preços dos pedaços de uma tora no intervalo de 1 a 10. Depois disso, você deve executar os algoritmos propostos, mostrando o valor total de venda obtidos por cada um e registrar seus tempos de execução. Ao final, mostre a porcentagem de acerto que a solução gulosa obtém em relação à solução da programação dinâmica.

2.1 Exemplo de entrada e saída

Um pequeno exemplo, para casos de teste com $n = 100, 200, \ldots, 2000$ é mostrado a seguir. Os tempos de execução dos algoritmos são mostrados em segundos. Os valores `vDP` e `tDP` indicam o valor total da venda obtido pelo algoritmo de programação dinâmica e o tempo de execução do algoritmo em segundos, respectivamente. Os valores `vGreedy` e `tGreedy` fazem as mesmas referências ao algoritmo guloso.

[1] *Introduction to Algorithms 3rd Edition*, Cormen *et al.*, MIT Press 2009.

n	vDP	tDP	vGreedy	tGreedy	%
100	300	0.000016	300	0.000013	100.00
200	1000	0.000057	1000	0.000046	100.00
300	600	0.000128	300	0.000055	50.00
400	1800	0.000263	800	0.000121	44.44
500	1500	0.000344	1500	0.000279	100.00
600	1800	0.000504	1800	0.000400	100.00
700	3150	0.000680	1400	0.000273	44.44
800	4800	0.000890	4800	0.000734	100.00
900	3600	0.001138	3600	0.000890	100.00
1000	5000	0.001435	5000	0.001398	100.00
1100	3300	0.001802	3300	0.001454	100.00
1200	7200	0.002149	7200	0.001744	100.00
1300	5200	0.002296	5200	0.001828	100.00
1400	2800	0.002875	1400	0.001445	50.00
1500	10500	0.003582	10500	0.002428	100.00
1600	6400	0.003477	3200	0.001619	50.00
1700	5100	0.004034	5100	0.003378	100.00
1800	4800	0.004634	1200	0.001195	25.00
1900	4750	0.005101	1267	0.001493	26.67
2000	16000	0.005844	16000	0.004757	100.00

3 Entrega

A atividade pode ser feita, preferencialmente, em duplas. Na primeira linha do código-fonte, coloque o nome completo de ambos como comentário. Apenas um dos integrantes da dupla submete a atividade no AVA.

Instruções para entrega da sua atividade:

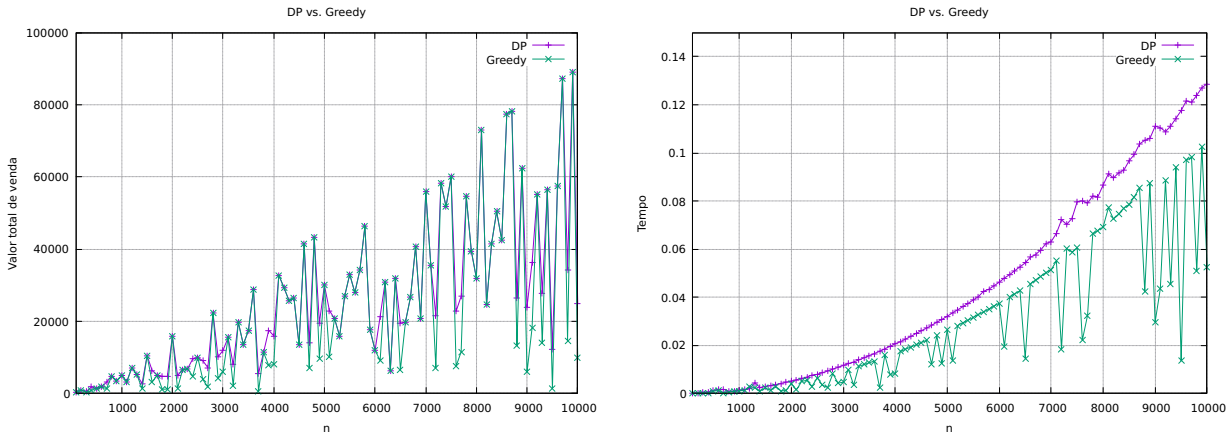
1. **O que entregar?**

   Um arquivo compactado a ser entregue deve conter o seguinte:

   - Programa desenvolvido;
   - Tabela dos valores dos cortes das toras e tempos de execução dos algoritmos (em texto), de acordo com a formatação apresentada na Seção 2.1, e
   - Um relatório em `.pdf` contendo gráficos gerados a partir das tabelas dos valores e dos tempos de execução dos algoritmos. Exemplos relativos à tabela da Seção 2.1 usando o software `gnuplot` são mostrados na Figura 2. Você pode utilizar qualquer outro software de sua preferência para gerar os gráficos. No relatório, faça uma discussão sobre os resultados obtidos.

   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) Valor total de venda.

(b) Tempo.

Figura 2: Gráficos da execução dos algortimos do corte a tora de madeira para o caso de teste com **`inc`** $= 100$, **`fim`** $= 10000$ e **`stp`** $= 100$.

A entrega será realizada diretamente no Sistema (AVA/UFMS), na disciplina de Implementação Algorítmica – T01. Você pode entregar a atividade quantas vezes quiser até às **23 horas e 59 minutos** do dia **16 de outubro de 2023**. A última versão entregue é aquela que será corrigida. Encerrado o prazo, trabalhos não serão mais aceitos.

3. **Linguagem de programação**

   O programa deve ser implementado em uma das seguintes linguagens: C/C++, Python ou Java.

4. **Erros**

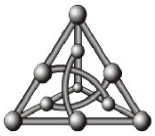
   Trabalhos com erros de compilação/interpretação receberão nota ZERO. Faça todos os testes necessários para garantir que seu programa está livre de erros de compilação/interpretação.

5. **Arquivo com o programa fonte**

   Seu(s) arquivo(s) contendo o(s) fonte(s) do(s) programa(s) na linguagem escolhida deve(m) estar 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.

6. **Conduta Ética**

   O trabalho deve ser feito INDIVIDUALMENTE/COM SEU GRUPO. Cada estudante tem responsabilidade sobre cópias de seu trabalho, mesmo que parciais. Não compartilhe seu programa ou trechos de seu programa. Você pode consultar seus colegas para esclarecer dúvidas e discutir idéias sobre o trabalho, ao vivo ou no fórum de discussão da disciplina, mas NÃO copie o programa!



Trabalhos considerados plagiados terão nota ZERO.