Disciplina: Projeto e Otimização de Algoritmos

Professor: Rafael Scopel

Trabalho T2

- Este trabalho consiste no desenho, análise e implementação de algoritmos de Programação Dinâmica e Branch-and-bound.
- O trabalho pode ser realizado grupos de até 5 alunos.
- O trabalho deve ser implementado em Java.

Objetivos do trabalho:

Problema 1

Suponha que você esteja gerenciando uma squad de DEVs e, a cada semana, tenha que escolher um trabalho para eles realizarem (Isso mesmo "crazy one week sprints"). Agora, como você pode imaginar, o conjunto de tarefas possíveis está dividido entre aqueles que são de baixa dificuldade (por exemplo, criar uma API) e aquelas que são de alta dificuldade (por exemplo, criar um detector de textos elaborados pelo ChatGPT). A questão básica, a cada semana, é se devemos aceitar uma tarefa de baixa ou alta dificuldade.

Se você selecionar uma tarefa de baixa dificuldade para sua equipe na semana i, obterá uma receita de $l_i > 0$ dinheiros; se você selecionar uma tarefa de alta dificuldade, obterá uma receita de $h_i > 0$ dinheiros. O problema, porém, é que para que a equipe assuma uma tarefa de alta dificuldade na semana i, é necessário que ela não faça nenhum trabalho (de qualquer tipo) na semana $i - 1$; eles precisam de uma semana inteira (W*F?!) de preparação para este tipo de tarefa. Por outro lado, não há problema em aceitar uma atividade de baixa dificuldade na semana i, mesmo que tenham feito um trabalho (de qualquer tipo) na semana $i - 1$.

Assim, dada uma sequência de n semanas, um plano é especificado por uma escolha de "baixa dificuldade", "alta dificuldade" ou "fazer nada" para cada uma das n semanas, com a propriedade de que se uma tarefa de "alta dificuldade" é escolhida para a semana $i > 1$, então "fazer nada" deve ser escolhido para a semana $i - 1$ **(Não há problema em escolher uma tarefa de alta dificuldade na semana 1).** O valor do plano é determinado de forma natural: para a cada i, você adiciona l_i ao valor se escolher "baixa dificuldade" na semana i, e adiciona h_i ao valor se escolher "alto dificuldade" na semana i (Você adiciona 0 se escolher "fazer nada" na semana i.). Cada tarefa só pode ser executada na sua respectiva semana.

O problema: Dados conjuntos de valores $l_1, l_2, \ldots, l_n$ e $h_1, h_2, \ldots, h_n$, encontre um plano de valor máximo.

Exemplo:
Suponha $n = 4$, e os valores de l_i e h_i são dados pela tabela a seguir. Então o plano de valor máximo seria escolher "fazer nada" na semana 1, uma tarefa de alta dificuldade na semana 2 e tarefas de baixa complexidade nas semanas 3 e 4. O valor deste plano seria 0 + 50 + 10 + 10 = 70.

	Week 1	Week 2	Week 3	Week 4
ℓ	10	1	10	10
h	5	50	5	1

Considerando o exposto acima, responda as seguintes perguntas:

1. Mostre que o algoritmo a seguir não resolve este problema corretamente, fornecendo uma instância na qual ele não retorna a resposta correta. Para evitar problemas com limites dos vetores, definimos $h_i = l_i = 0$ quando $i > n$. No seu exemplo, diga qual é a resposta correta e o que o algoritmo abaixo encontra.

```
For iterations i = 1 to n
  If h_{i+1} > ℓ_i + ℓ_{i+1} then
    Output "Choose no job in week i"
    Output "Choose a high-stress job in week i+1"
    Continue with iteration i+2
  Else
    Output "Choose a low-stress job in week i"
    Continue with iteration i+1
  Endif
End
```

2. Forneça um algoritmo eficiente (tempo polinomial baixo) que assuma valores para $l_1, l_2, \ldots, l_n$ e $h_1, h_2, \ldots, h_n$ que:
    a. Retorna o valor de um plano ideal;
    b. Lista quais tarefas foram executadas em cada semana.

<u>O algoritmo do item 2, deve possuir a seguinte interface principal:</u>

void solveP1(int l[], int h[])

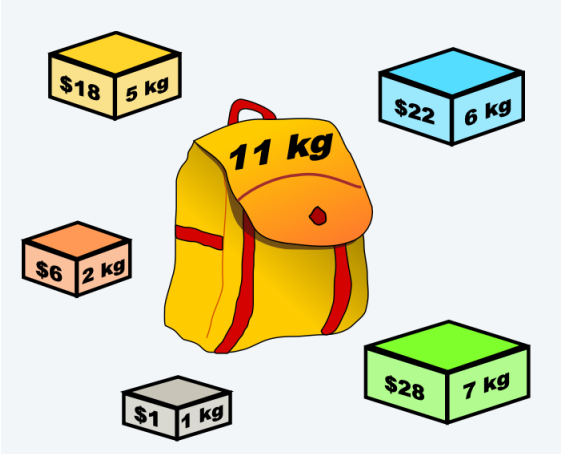
A função recebe dois vetores como parâmetros e imprime na tela as repostas para os itens 2.a e 2.b.

Estruture a resposta no formato de um relatório com as seguintes sessões:

1. O Problema;
2. Resposta do item 1.
3. Resposta do item 2:
    a. O Algoritmo;
    b. Análise do Algoritmo;
    c. Implementação e Tempo de Execução;

Problema 2

Resolva o problema da mochila abaixo utilizando Branch-and-Bound:



Estruture a resposta no formato de um relatório com as seguintes sessões:

1. O Problema;
2. O Algoritmo;
    a. Retorna o valor ótimo para colocar na mochila;
    b. Lista quais itens foram colocados na mochila.
3. Análise do Algoritmo;
4. Implementação e Tempo de Execução;

O algoritmo do item 2, deve possuir a seguinte interface principal:

void solveP2(int n, int wi[], int vi[], int W)

A função recebe a quantidade de itens n, um vetor de peso *wi,* um vetor de valores *vi* e a capacidade total da mochila *W* como parâmetros e imprime na tela as repostas para os itens 2.a e 2.b.

Entregáveis do Trabalho:

1. Relatório em Word com:
    a. Capa com título e nome dos integrantes;
    b. Resolução do problema 1;
    c. Resolução do problema 2;

2. Código fonte comentado.

Critérios de Avaliação:

- Código fonte compilável sem erros;
- Códigos que não seguirem as interfaces solveP1 e solveP2 não serão considerados;
- Qualidade e profundidade das análises do relatório;
- Qualidade e documentação do código fonte;