Disciplina: Projeto e Otimização de Algoritmos

Professor: Rafael Scopel

Prova P1

Instruções:

- Prova sem consulta e individual.
- É permitido o uso de calculadora científica.
- A prova é composta por 5 questões, todas com o peso de 2 pontos.
- Responda as questões justificando os seus resultados e apresentado a memória de cálculo.
- Responda as questões na folha pautada de respostas. Respostas nesta folha não serão consideradas.

1. (2 pontos) Verifique a veracidade das afirmativas sobre algoritmos Greedy abaixo e selecione a alternativa correta:

   I. Um algoritmo é greedy se ele constrói uma solução com pequenos passos, fazendo uma escolha local (míope) a cada passo para otimizar um determinado critério ou função objetivo. V

   II. Para garantir que um algoritmo greedy resolve um determinado problema precisamos procurar por dois ingredientes chave: a propriedade da escolha greedy e a subestrutura ótima do problema. V

   III. O algoritmo greedy do troco em moedas (Cashiers-Algorithm) funciona quando temos apenas as seguintes moedas 1 centavo, 10 centavos e 25 centavos. F

   - O algoritmo greedy sempre produz uma solução ótima para todas as denominações de moedas?
   - Imagine se o Real só possui-se as seguintes moedas 1 centavo, 10 centavos e 25 centavos.
   - Use o algoritmo do Cashier para dar troco para 30 centavos?
   - 1 moeda de 25 centavos
   - 5 moedas de 1 centavo
   - Total de 6 moedas. Essa solução é ótima?
   - Claramente não, pois 3 moedas de 10 centavos entregam os mesmos 30 centavos de troco!!!

( )	Apenas I é verdadeira.
( )	I, II e III são verdadeiras.
( )	Apenas II é verdadeira.
(x)	Apenas I e II são verdadeiras.
( )	Todas as alternativas são falsas.

2. (2 pontos) Verifique a veracidade das afirmativas sobre algoritmos de Divisão e Conquista abaixo e selecione a alternativa correta:

   I. O valor da recorrência $T(n) = 4T\left(\frac{n}{16}\right) + n$ é $\Theta(\sqrt{n})$. F

TEOREMA T.1 O método mestre

- **Caso 1: Se $f(n) = O(n^{\log_b a - \epsilon})$ para alguma constante $\epsilon > 0$, então $T(n) = \Theta(n^{\log_b a})$.**
- **Caso 2: Se $f(n) = \Theta(n^{\log_b a})$, então $T(n) = \Theta(n^{\log_b a} \log_2 n)$.**
- **Caso 3: Se $f(n) = \Omega(n^{\log_b a + \epsilon})$ para alguma constante $\epsilon > 0$, e se $af(n/b) \leq cf(n)$ para alguma constante $c < 1$ e todos os n suficientemente grandes, então $T(n) = \Theta(f(n))$.**

$T(n) = aT(n/b) + f(n) = 4T\left(\frac{n}{16}\right) + n$

onde $a = 4$, $b = 16$, $f(n) = n$

Então temos que $n^{log_b a} = n^{log_{16} 4} = \sqrt{n} = \Theta(\sqrt{n})$

Caso 3:

Como $f(n) = O(n^{\log_b a+\epsilon})$, onde $\epsilon = 0.5$, pois $n^{\log_b a+\epsilon} = n^{log_{16} 4+0.5} = n^{0.5+0.5} = n$.

$af(n/b) \leq cf(n) \rightarrow 4(n/16) \quad \leq cn \rightarrow \frac{1}{4}n \leq cn$
onde $c = \frac{1}{4} < 1$.

Portanto, $n^{log_b a} < f(n)$, a solução é $T(n) = \Theta(n) = \Theta(f(n))$.

II. A recorrência $T(n) = 16T\left(\frac{n}{32}\right) + n^{\frac{1}{3}}$ é resolvida pelo Caso 2 do Teorema Mestre. F

$T(n) = aT(n/b) + f(n) = 16T\left(\frac{n}{32}\right) + n^{\frac{1}{3}}$

onde $a = 16$, $b = 32$, $f(n) = n^{\frac{1}{3}}$

Então temos que $n^{log_b a} = n^{log_{32} 16} = n^{\frac{4}{5}} = \Theta\left(n^{\frac{4}{5}}\right)$

Caso 1:

Como $f(n) = O(n^{\log_b a-\epsilon})$, onde $\epsilon = \frac{7}{15} = 0.4667$, pois $n^{\log_b a-\epsilon} = n^{log_{32} 16-\frac{7}{15}} = n^{\frac{4}{5}-\frac{7}{15}} = n^{\frac{1}{3}}$.

Portanto, $n^{log_b a} > f(n)$, a solução é $T(n) = \Theta\left(n^{\frac{4}{5}}\right)$.

III. O método da divisão e conquista é composto pelas etapas de dividir, conquistar e combinar. Onde o custo para dividir e combinar é representado pela função $f(n)$ na estrutura de recorrência $T(n) = aT(n/b) + f(n)$. V

( )	Apenas I é verdadeira.
( )	I, II e III são verdadeiras.
(x)	Apenas III é verdadeira.
( )	Apenas I e II são verdadeiras.
( )	Todas as alternativas são falsas.

3. (2 pontos) Verifique a veracidade das afirmativas sobre algoritmos de Programação Dinâmica abaixo e selecione a alternativa correta:

I. A Programação Dinâmica explora um conjunto exponencialmente grande de possíveis soluções para o problema, ela faz isso examinando explicitamente todas as soluções. F

Apesar de estar explorando um conjunto exponencialmente grande de possíveis soluções para o problema, ela faz isso sem nunca examinar explicitamente todas as soluções.

II. O termo $\text{OPT}(j) = v_j + \text{OPT}\big(p(j)\big)$ no problema de escalonamento de tarefas ponderadas significa que a requisição j está no conjunto solução. V

III. Memoização é a técnica de armazenar valores computados previamente para uso futuro. V

(x)	Apenas I é falsa.
( )	I, II e III são verdadeiras.
( )	Apenas III é verdadeira.
( )	Apenas I e II são verdadeiras.
( )	Todas as alternativas são falsas.

4. (2 pontos) Com base no algoritmo de Huffman codes responda as questões abaixo:

```
HUFFMAN(S)
1  n = |S|
2  Q = S
3  for i = 1 to n − 1
4       allocate a new node w|
5       y = EXTRACT-MIN(Q)
6       z = EXTRACT-MIN(Q)
7       w|left = z
8       w|right = y
9       w|freq = z.freq + y.freq
10      INSERT(Q w|)
11 return EXTRACT-MIN(Q)      // the root of the tree is the only node left
```

a. O Huffman codes utiliza que abordagem algorítmica para compressão de dados?
b. Qual o tempo de execução do algoritmo?
c. Defina a árvore final para o seguinte dicionário (letra:frequência): {[a:12], [b:11], [c:8], [d:17], [e:24], [f:5]}.
d. Descreva os códigos definidos pela árvore do item c.

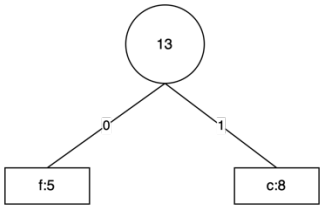
a) O algoritmo Huffman codes utiliza uma abordagem greedy para compressão de dados.

b) Utilizando uma implementação de filas de prioridade via heaps, podemos fazer cada inserção e extração da execução mínima no tempo $O(\log n)$. Somando todas as n iterações, obtemos um tempo total de execução de $O(n \log n)$;

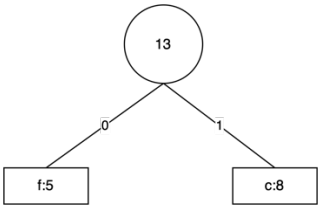
c) Resposta:

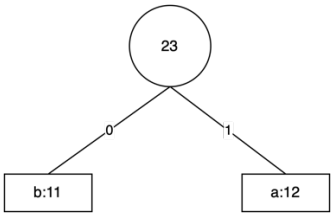
1) {[f:5] ,[c:8], [b:11], [a:12], [d:17], [e:24]}

2) { [b:11], [a:12], 13, [d:17], [e:24]}

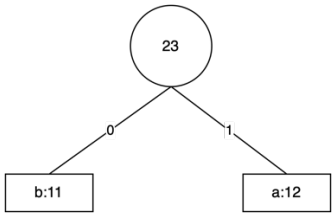


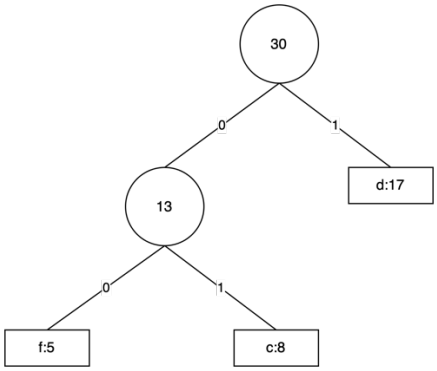
3) {13, [d:17], 23, [e:24]}



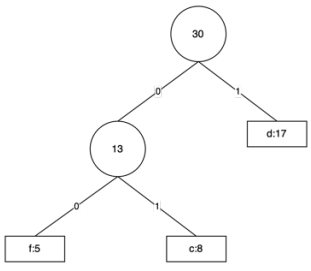


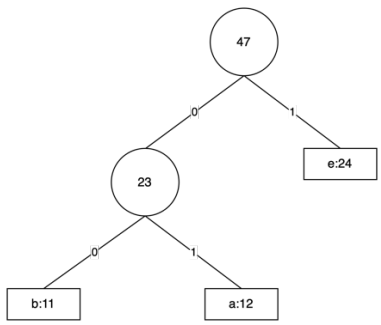
4) {23, [e:24], 30}



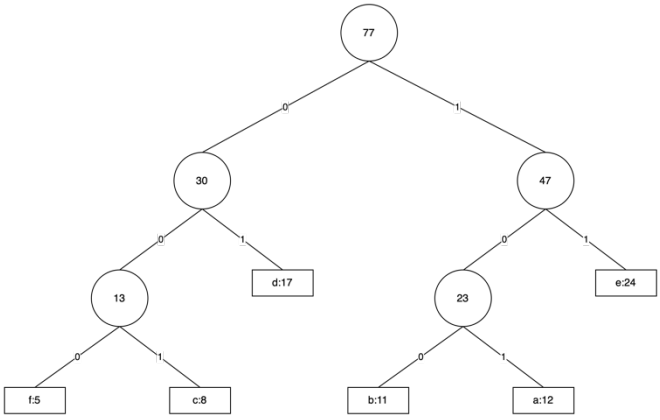


5) {30, 47}





5) {77}



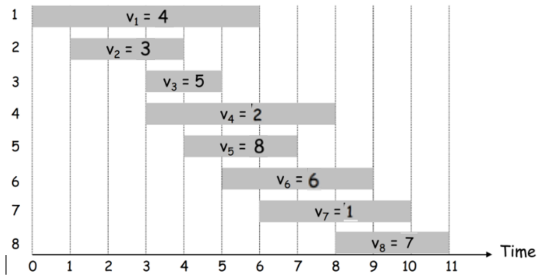
d:01
e:11
f:000
c:001
b:100
a:101

d)

a:101
b:100
c:001
d:01
e:11
f:000

5. (2 pontos) Utilize a abordagem de programação dinâmica com memoização para resolver a alocação de tarefas ponderadas abaixo.

$$\boldsymbol{OPT(j)} = \begin{cases} \mathbf{0} & \boldsymbol{, se\ j = 0} \\ \boldsymbol{max\left(v_j + OPT(p(j)), OPT(j-1)\right)} & \boldsymbol{, se\ j > 0} \end{cases}$$



Responda as seguintes perguntas:

a. O que é memoização?
b. Calcule p(j) para todas as requisições.
c. Calcule $OPT(8)$.
d. Indique quais requisições fazem parte da solução ótima dada por $OPT(8)$.

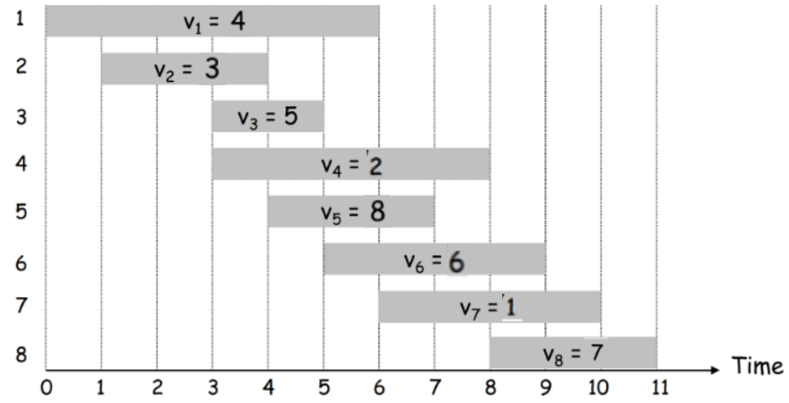
a)

A ideia básica para Programação Dinâmica é similar a intuição dos algoritmos de divisão e conquista, mas é essencialmente o oposto dos algoritmos greedy. Na programação dinâmica implicitamente procuramos o espaço de todas as soluções possíveis. De certa maneira a programação dinâmica opera muito próximo da busca via força bruta. Apesar de estar explorando um conjunto exponencialmente grande de possíveis soluções para o problema, ela faz isso sem nunca examinar explicitamente todas as soluções. Este efeito é obtido por meio da memoização.

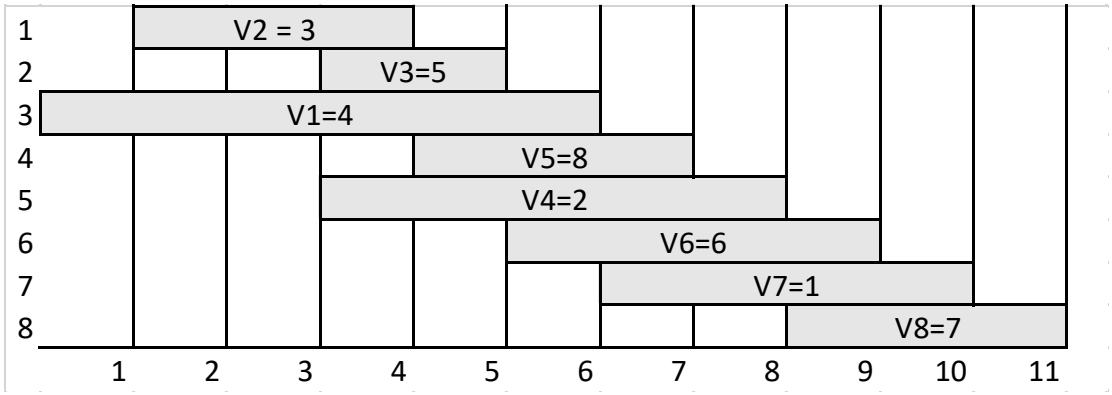
Poderíamos armazenar o valor de chamadas recursivas anteriores numa variável global na primeira vez que computarmos o valor e depois continuar a utilizar o valor pré-computado nas chamadas recursivas subsequentes. Essa técnica de salvar valores que já foram computados é chamada de MEMOIZAÇÃO!

b)

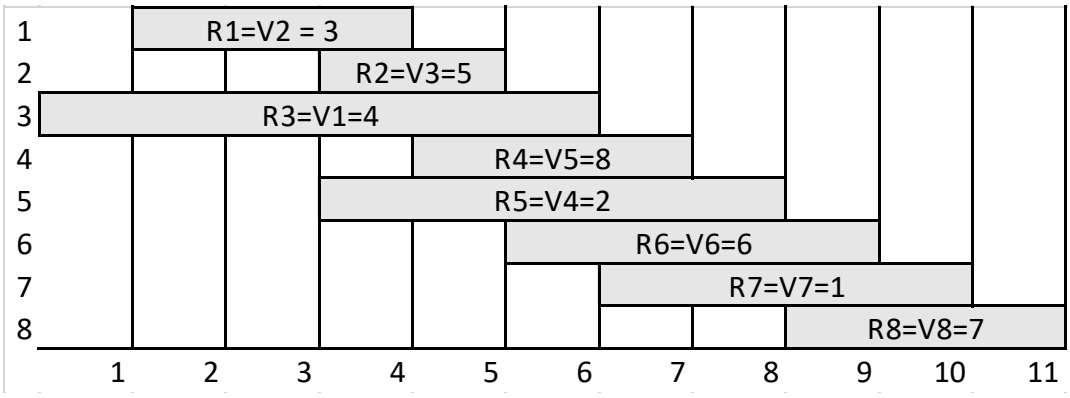
i. Paço, ordenar as requisições por tempo de finalização:



Nova visão com as requisições ordenadas por tempo de finalização:



Renomear as requisições:



Valores de $\boldsymbol{p(j)}$

$$p(0) = 0$$
$$p(R1) = 0$$
$$p(R2) = 0$$
$$p(R3) = 0$$
$$p(R4) = 1$$
$$p(R5) = 0$$

$p(R6) = 2$
$p(R7) = 3$
$p(R8) = 5$

c)

M-Compute-Opt(R8)

$$\boldsymbol{OPT(R8) = M[8] = max\left(v_8 + \mathrm{M-Compute-Opt}(p(8)), \mathrm{M-Compute-Opt}(7)\right)}$$
$$\boldsymbol{= max\left(v_8 + \mathrm{M-Compute-Opt}(5), \mathrm{M-Compute-Opt}(7)\right)}$$

$$\boldsymbol{OPT(R7) = M[7] = max\left(v_7 + \mathrm{M-Compute-Opt}(p(7)), \mathrm{M-Compute-Opt}(6)\right)}$$
$$\boldsymbol{= max\left(v_7 + \mathrm{M-Compute-Opt}(3), \mathrm{M-Compute-Opt}(6)\right)}$$

$$\boldsymbol{OPT(R6) = M[6] = max\left(v_6 + \mathrm{M-Compute-Opt}(p(6)), \mathrm{M-Compute-Opt}(5)\right)}$$
$$\boldsymbol{= max\left(v_6 + \mathrm{M-Compute-Opt}(2), \mathrm{M-Compute-Opt}(5)\right)}$$

$$\boldsymbol{OPT(R5) = M[5] = max\left(v_5 + \mathrm{M-Compute-Opt}(p(5)), \mathrm{M-Compute-Opt}(4)\right)}$$
$$\boldsymbol{= max\left(v_5 + \mathrm{M-Compute-Opt}(0), \mathrm{M-Compute-Opt}(4)\right)}$$

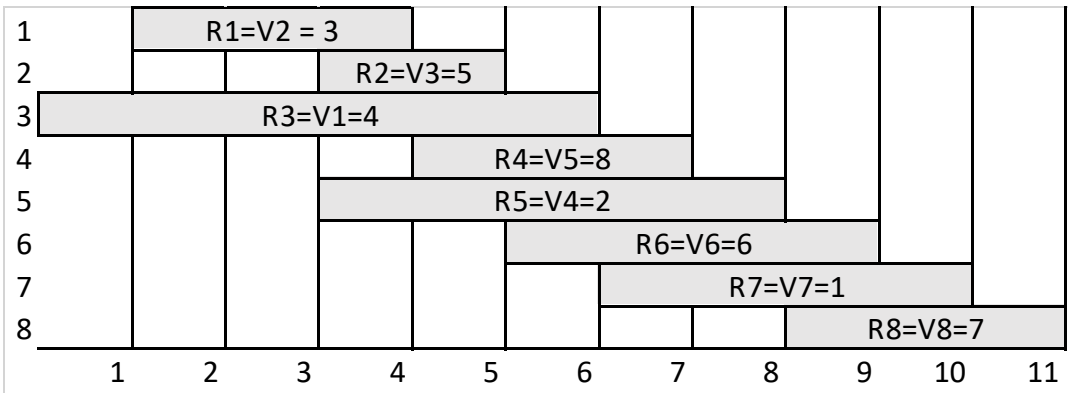
$$\boldsymbol{OPT(R4) = M[4] = max\left(v_4 + \mathrm{M-Compute-Opt}(p(4)), \mathrm{M-Compute-Opt}(3)\right)}$$
$$\boldsymbol{= max\left(v_4 + \mathrm{M-Compute-Opt}(1), \mathrm{M-Compute-Opt}(3)\right)}$$

$$\boldsymbol{OPT(R3) = M[3] = max\left(v_3 + \mathrm{M-Compute-Opt}(p(3)), \mathrm{M-Compute-Opt}(2)\right)}$$
$$\boldsymbol{= max\left(v_3 + \mathrm{M-Compute-Opt}(0), \mathrm{M-Compute-Opt}(2)\right)}$$

$$\boldsymbol{OPT(R2) = M[2] = max\left(v_2 + \mathrm{M-Compute-Opt}(p(2)), \mathrm{M-Compute-Opt}(1)\right)}$$
$$\boldsymbol{= max\left(v_2 + \mathrm{M-Compute-Opt}(0), \mathrm{M-Compute-Opt}(1)\right)}$$

$$\boldsymbol{OPT(R1) = M[1] = max\left(v_1 + \mathrm{M-Compute-Opt}(p(1)), \mathrm{M-Compute-Opt}(0)\right)}$$
$$\boldsymbol{= max\left(v_1 + \mathrm{M-Compute-Opt}(0), \mathrm{M-Compute-Opt}(0)\right)}$$

$$\boldsymbol{OPT(0) = M[0] = 0}$$



j	0	1	2	3	4	5	6	7	8
M[]	0								

$$\boldsymbol{OPT(0) = M[0] = 0}$$

$$\boldsymbol{OPT(1) = M[1] = max}\left(\boldsymbol{v_1} + \text{M} - \text{Compute} - \text{Opt}(\boldsymbol{p(1)}), \text{M} - \text{Compute} - \text{Opt}(\boldsymbol{0})\right)$$
$$= \boldsymbol{max}(\boldsymbol{v_1} + \text{M} - \text{Compute} - \text{Opt}(\boldsymbol{0}), \text{M} - \text{Compute} - \text{Opt}(\boldsymbol{0}))$$
$$= \boldsymbol{max}(\boldsymbol{v_1} + \text{M}[0], \text{M}[0]) = \boldsymbol{max}(\boldsymbol{3} + 0,0) = \boldsymbol{3}$$

j	0	1	2	3	4	5	6	7	8
M[]	0	3							

$$\boldsymbol{OPT(R2) = M[2] = max}\left(\boldsymbol{v_2} + \text{M} - \text{Compute} - \text{Opt}(\boldsymbol{p(2)}), \text{M} - \text{Compute} - \text{Opt}(\boldsymbol{1})\right)$$
$$= \boldsymbol{max}(\boldsymbol{v_2} + \text{M} - \text{Compute} - \text{Opt}(\boldsymbol{0}), \text{M} - \text{Compute} - \text{Opt}(\boldsymbol{1}))$$
$$= \boldsymbol{max}(\boldsymbol{v_2} + \text{M}[0], \text{M}[1]) = \boldsymbol{max}(\boldsymbol{5} + 0,3) = \boldsymbol{5}$$

j	0	1	2	3	4	5	6	7	8
M[]	0	3	5						

$$\boldsymbol{OPT(R3) = M[3] = max}\left(\boldsymbol{v_3} + \text{M} - \text{Compute} - \text{Opt}(\boldsymbol{p(3)}), \text{M} - \text{Compute} - \text{Opt}(\boldsymbol{2})\right)$$
$$= \boldsymbol{max}(\boldsymbol{v_3} + \text{M} - \text{Compute} - \text{Opt}(\boldsymbol{0}), \text{M} - \text{Compute} - \text{Opt}(\boldsymbol{2}))$$
$$= \boldsymbol{max}(\boldsymbol{v_3} + \text{M}[0], \text{M}[2]) = \boldsymbol{max}(\boldsymbol{4} + 0,5) = \boldsymbol{5}$$

j	0	1	2	3	4	5	6	7	8
M[]	0	3	5	5					

$$\boldsymbol{OPT(R4) = M[4] = max}\left(\boldsymbol{v_4} + \text{M} - \text{Compute} - \text{Opt}(\boldsymbol{p(4)}), \text{M} - \text{Compute} - \text{Opt}(\boldsymbol{3})\right)$$
$$= \boldsymbol{max}(\boldsymbol{v_4} + \text{M} - \text{Compute} - \text{Opt}(\boldsymbol{1}), \text{M} - \text{Compute} - \text{Opt}(\boldsymbol{3}))$$
$$= \boldsymbol{max}(\boldsymbol{v_4} + \text{M}[1], \text{M}[3]) = \boldsymbol{max}(\boldsymbol{8} + 3,5) = \boldsymbol{11}$$

j	0	1	2	3	4	5	6	7	8
M[]	0	3	5	5	11				

j	0	1	2	3	4	5	6	7	8
M[]	0	3	5	5	11				

$$\boldsymbol{OPT(R5) = M[5] = max\left(v_5 + \text{M} - \text{Compute} - \text{Opt}(\boldsymbol{p(5)}), \text{M} - \text{Compute} - \text{Opt}(\boldsymbol{4})\right)}$$
$$\boldsymbol{= max\left(v_5 + \text{M} - \text{Compute} - \text{Opt}(0), \text{M} - \text{Compute} - \text{Opt}(4)\right)}$$
$$\boldsymbol{= max(v_5 + \text{M}[0], \text{M}[4]) = max(2 + 0, 11) = 11}$$

j	0	1	2	3	4	5	6	7	8
M[]	0	3	5	5	11	11			

$$\boldsymbol{OPT(R6) = M[6] = max\left(v_6 + \text{M} - \text{Compute} - \text{Opt}(\boldsymbol{p(6)}), \text{M} - \text{Compute} - \text{Opt}(\boldsymbol{5})\right)}$$
$$\boldsymbol{= max\left(v_6 + \text{M} - \text{Compute} - \text{Opt}(2), \text{M} - \text{Compute} - \text{Opt}(5)\right)}$$
$$\boldsymbol{= max(v_6 + \text{M}[2], \text{M}[5]) = max(6 + 5, 11) = 11}$$

j	0	1	2	3	4	5	6	7	8
M[]	0	3	5	5	11	11	11		

$$\boldsymbol{OPT(R7) = M[7] = max\left(v_7 + \text{M} - \text{Compute} - \text{Opt}(\boldsymbol{p(7)}), \text{M} - \text{Compute} - \text{Opt}(\boldsymbol{6})\right)}$$
$$\boldsymbol{= max\left(v_7 + \text{M} - \text{Compute} - \text{Opt}(3), \text{M} - \text{Compute} - \text{Opt}(6)\right)}$$
$$\boldsymbol{= max(v_7 + \text{M}[3], \text{M}[6]) = max(6 + 1, 11) = 11}$$

j	0	1	2	3	4	5	6	7	8
M[]	0	3	5	5	11	11	11	11	

$$\boldsymbol{OPT(R8) = M[8] = max\left(v_8 + \text{M} - \text{Compute} - \text{Opt}(\boldsymbol{p(8)}), \text{M} - \text{Compute} - \text{Opt}(\boldsymbol{7})\right)}$$
$$\boldsymbol{= max\left(v_8 + \text{M} - \text{Compute} - \text{Opt}(5), \text{M} - \text{Compute} - \text{Opt}(7)\right)}$$
$$\boldsymbol{= max(v_8 + \text{M}[5], \text{M}[7]) = max(7 + 11, 11) = 18}$$

j	0	1	2	3	4	5	6	7	8
M[]	0	3	5	5	11	11	11	11	18

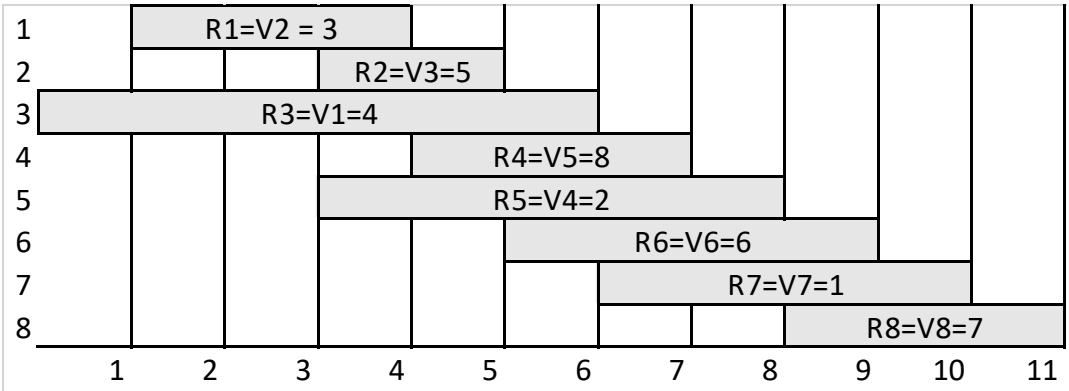
d)

```
FIND-SOLUTION(j)

IF (j = 0)
   RETURN ∅.
ELSE IF (w_j + M[p[j]] > M[j − 1])
   RETURN { j } ∪ FIND-SOLUTION(p[j]).
ELSE
   RETURN FIND-SOLUTION(j − 1).
```

$$\boldsymbol{S = \{\quad\}}$$

$\boldsymbol{Find - Solution(R8) = M[8] > M[7]}$	$\boldsymbol{S = \{R8\}}$
$\boldsymbol{Find - Solution(p(R8)) = M[5] > M[4]}$	$\boldsymbol{S = \{R4, R8\}}$
$\boldsymbol{Find - Solution(p(R4)) = M[1] > M[0]}$	$\boldsymbol{S = \{R1, R4, R8\}}$



As requisições que compõem a solução final são: $\boldsymbol{S = \{R1, R4, R8\}}$, e traduzindo para a instância original do problema: $\boldsymbol{S = \{V2, V5, V8\}}$