Daniela dos Santos Tomas
Pontuação: 11.5/20 (total: 11.6/20)

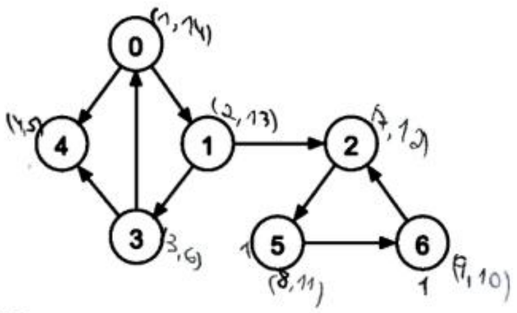
Design of Algorithms L.EIC016

First Test - Mar. 31, 2023

- Duration: 120 minutes.
- The test consists of 9 questions for a total of 20 points.
- The use of a single, double-sided and typed or hand-written A4 sheet is allowed.
- The use of electronic devices such as cell phones or smart-watches is not allowed.
- Answers must be given EXCLUSIVELY on the answer pages 6 through 12.
- Different questions may have different grade marks. In case there are more then one item in a question, the distribution of the question grade is random. As such, your best strategy is to answer as best as you can in all of them.
- Use a B or HB pencil for your answers or pen if you would like, but if you use pencil, erasing to change your answer is usually easier.
- Do not redraw any answer boxes and do not fill in any boxes in the *Reserved* shaded sections of the answer sheets. These are obviously reserved for grading.

Name: Daniela dos Santos Tomas (202004946)

Question 1
[1 point] Consider the directed graph G below with 7 nodes and a representation that uses an adjacency list where neighbors are list in ascending order of node numbering.

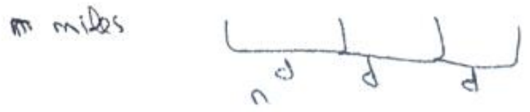


For this graph answer the following:

- Show the reverse graph G^R and its strongly-connected components (SCCs).
- Determine a possible topological sorting of G.
- Discuss the time complexity of the topologic sorting in terms of the number of vertices (nodes) and number of edges.

Please write the answer on the separate answer sheet.

Question 2
[3 points] Ralph wants to drive from Los Angeles to Chicago along route 66. His car's fuel tank, when full, holds enough fuel to travel n miles, and his map gives the distances between gas stations on this route. Ralph wishes to make as few stops for fuel as possible along the way. Develop an efficient algorithm by which he can determine which gas stations he should stop at, and prove that your strategy yields an optimal solution.
Please write the answer on the separate answer sheet.



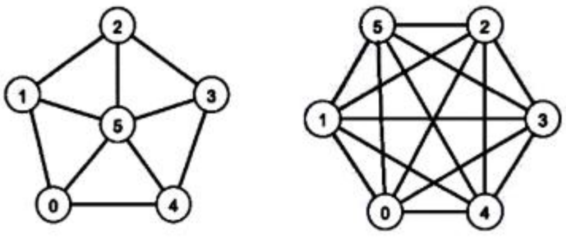
Question 3

[2 points] Using the Brute-Force algorithmic approach devise an algorithm that determines a maximal matching for a generic undirected bipartite graph. In addition, develop a lazy strategy that can find a maximal matching as soon as possible. ~~To clarify, we illustrate a sample instance of a bipartite undirected graph below~~. Your algorithm should be generic and work correctly for a generic bipartite undirected graph.

Please write the answer on the separate answer sheet.

Question 4

[2 points] Consider the two weighted undirected graphs shown below as illustrative examples of a class of undirected and connected graphs with `N` nodes where all edges have unit weight. The specific graphs below are particular instances of this class of graphs for `N=6` and for a particular arrangement of nodes.



For this graph class of graph adapt Prim's algorithm, to find a minimum-cost spanning tree possibly simplifying it and arriving at a more efficient (from a time complexity standpoint) algorithmic variant. Describe your algorithm using *reasonable* pseudo-code and argue for its improved time complexity.

Please write the answer on the separate answer sheet.

Question 5

[2 points] In class we examined the 0/1 Knapsack problem and its fractional variant where fractions of selected items can be included in the Knapsack with maximum capacity `W`. We also discussed a greedy algorithm where items were inserted in the Knapsack by descending order of their ratio of value per weight `(v/w)`. We showed that this greedy approach was optimal.

In this problem we explore another variant of the fractional Knapsack problem in which all the items have the same value, but possibly different weights. Note that you can include a fraction of any item in the Knapsack.

For this particular class of problem instances, derive and discuss the complexity of an efficient algorithm that maximizes the value of the items in a Knapsack of capacity `W` and argue for the optimality of your solution.

Please write the answer on the separate answer sheet.

Question 6

[2 points] Consider the recursive function G depicted below with one integer argument and other arguments omitted here for simplicity. The G function makes use of another function, F which does some work that is either polynomial, logarithmic or even constant time with respect to the input instance's size, n.

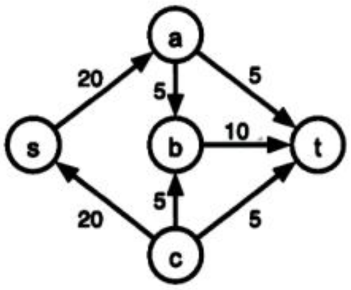
```
public int G (int n, ...) {
   if (n < 1) {   }
     return 1;    } +2
   }
                                    +n+3
   F(n, ...); } +n
                                    +3x T(n/9)
   int x = G(n / 9, ...);  }
   int y = G(n / 9, ...);  }
   int z = G(n / 9, ...);  }

   return x + y * z; } +3
}
```

Derive a recurrence that models the execution time complexity of G and derive its asymptotic time complexity using the Master Theorem as discussed in class. Clearly, you need to examine various complexity scenarios regarding F itself. Discuss the overall complexity when F(n, ...) is both linear and quadratic with respect to n.

Would it make sense to invest in a more efficient implementation of F to be, say, logarithmic with respect to n? Why or why not?
Please write the answer on the separate answer sheet.

Question 7

[4 points] Consider the network flow below and nodes s and t as source and sink, respectively.

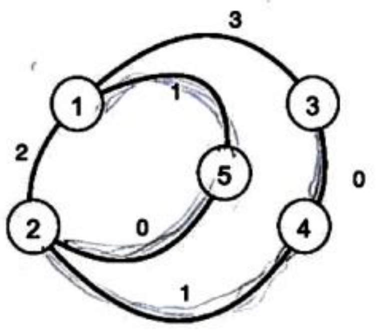


For this network, derive:

- A minimal cut indicating the corresponding capacity.
- List the augmenting paths using Edmonds-Karp algorithm strategy, indicating the order in which the Edmonds-Karp would explore them.
- The maximal flow between s and t.
- Identify the edges whose capacity can be reduced, and to which minimum value, while not reducing the overall maximal flow between s and t. Explain your choice.

Please write the answer on the separate answer sheet.

Question 8
[2 points] Consider the undirected and weighted graph depicted below.

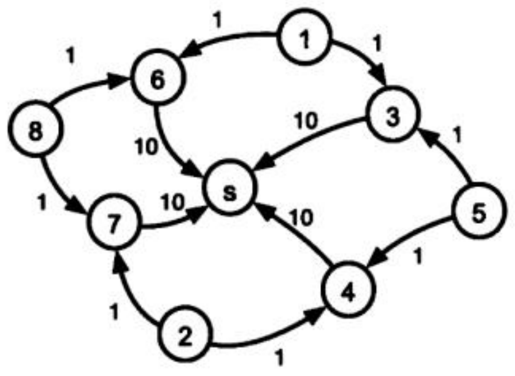


Using Dijkstra's greedy shortest-path algorithm determine the values of the shortest-distance between node 1 and the other nodes. Show the distance values this algorithm produces after one iteration after the initialization step.

Describe the order in which the nodes are examined and considered as part of the resulting tree of shortest-paths.

Show the resulting MST using Prim's algorithm rooted at node 1 and why is this tree the same as the tree of shortest paths derived using Dijkstra's algorithm? Discuss when are they different.
Please write the answer on the separate answer sheet.

Question 9
[2 points] You are asked to develop an efficient algorithm that determines the maximum flow to a given node of a network flow graph, say node s, from all the other graph nodes. A simple algorithm could consider adding the capacity of all the edges directly connected to s, but as the sample network flow graph below shows, this can be a gross over estimation of this maximum flow.



You may describe your solution using pseudo-code, if you would like, but a description of your approach alongside a simple, but not simplistic example with a network flow graph example, would suffice. Also, related the complexity of your implementation with the complexity of the implementation of the max-flow algorithms discussed in class.
Please write the answer on the separate answer sheet.

End of questions; answers on the next pages.

Answers

Daniela dos Santos Tomas (202004946)

- Answers must be given exclusively on pages 6–12; answers given on the other pages will be ignored.
- Use a B or HB pencil for your answers or pen if you would like, but if you use pencil, erasing to change your answer is usually easier.

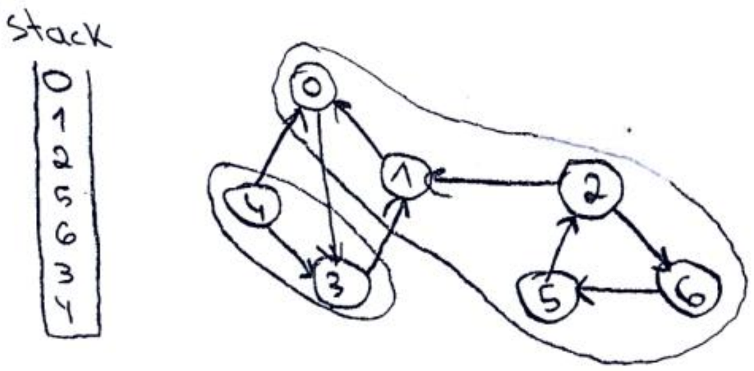
Question 1 0.6/1

Q1 ☐ 0 ☐ 0.2 ☐ 0.4 [✓ 0.6] ☐ 0.8 ☐ 1 RESERVED

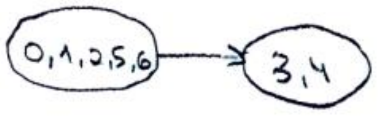
a) ✓ ✗

b) ✗

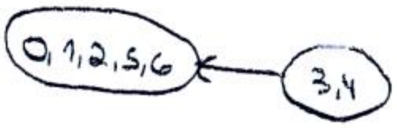
c) ✓ ✓



Graph of SCCs:

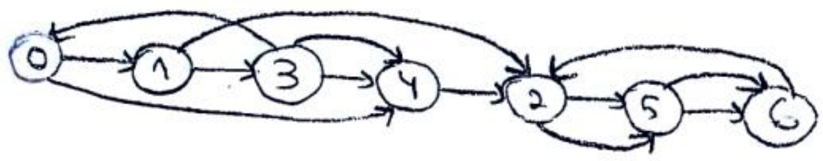


SCCs in G^R



Possible Topological Sorting:

$(0 \to 1 \to 3 \to 4 \to 2 \to 5 \to 6)$



Time Complexity:

$O(|E|+|V|)$, porque tem de percorrer todos os vértices e arestas no pior dos casos.

+115/7/6+

Question 2 1.5/3

Q2 0 0.5 1.0 [✓ 1.5] 2.0 2.5 3 RESERVED

```
Solution[]=0 ; D=[];
Sum =0;
    foreach $d_i \in D$ do
        if (miles+($d_{i+1}$- $d_i$) > miles) then
            solution[i]= i;

            miles =n;
        else
            miles = miles - $d_i$;
```

Why does it work?
complexity ?

Question 3 1/2

Q3 0 0.5 [✓ 1.0] 1.5 2 RESERVED

Algoritmo: Explorar exaustivamente todas as ligações possíveis, explorando todos os nós e arestas.

good start of an idea... but very incomplete.

+115/8/5+

Question 4 1.5/2

Q4 [ ] 0 [ ] 0.5 [ ] 1.0 [✓ 1.5] [ ] 2 RESERVED

```
function Prim(G,w,r)
  Q = V[G]
  foreach u ∈ Q do
    key[u] = ∞
  key [r] = 0
  prod[r] = NIL
  while Q ≠ {} do
    u = extract(Q)   // ⊛
    foreach v ∈ Adj[u] do
      if(v ∈ Q and w(u,v) < key(v)) then
        prod[v] = u
        key[v] = w(u,v)
```

Para este algoritmo, como todas as arestas têm o mesmo peso, não é necessário extrair a aresta mínima e mínima adjacente, podemos extrair qualquer aresta, tornando o algoritmo mais eficiente.

⊛ Antes era O(log V) e agora é O(1)

But you still need to check if edge is "safe"...

see solution.

Question 5 2/2

Q5 ☐0 ☐0.5 ☐1.0 ☐1.5 [✓ 2] RESERVED

```
function Knapsack (v, w, W)
  weight = 0
  while weight < W do
    select element i with minimal $w_i$
    if ($w_i$ + weight ≤ W) then
      $x_i$ = 1; weight += $w_i$
    else
      $x_i$ = (W - weight) / $w_i$; weight = W
```

Como todos os valores são iguais, deve-se escolher sempre o item com menor peso. Podemos dividir o problema em subproblemas, considerando por exemplo, que os pesos começam pelo segundo, w_{i+1} e selociendo sempre o peso menor. A solução ótima do problema original, pode ser obtida adicionando o primeiro peso, w_i, ao subproblema.

Question 6 1.5/2

Q6 ☐0 ☐0.5 ☐1.0 [✓ 1.5] ☐2 RESERVED

$$T(n) = \begin{cases} 2 & \text{if } n<1 \\ 3T(\frac{n}{9}) + n + 3 & \text{otherwise} \end{cases}$$

Base case: +2

Non-recursive work: +n+3

Recursive work: $+3 \times T(n/9)$

<u>Se f é linear</u>

$$T(n) = \begin{cases} 2 & \text{if } n<1 \\ 3T(\frac{n}{9}) + n^2 + 3 & \text{otherwise} \end{cases}$$

<u>se f é quadratica</u>

$\log_3 9 = \frac{1}{2}$ $\frac{1}{2} > 1$

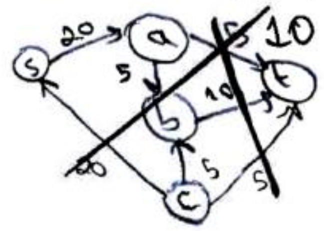
$T(n) \in \Theta(n^2)$

$$\left.\begin{aligned} f(n) &= n+3 \\ f(n) &= n^2+3 \end{aligned}\right\} \Theta(n^c)$$

$c = 1, a = 2, b = 9$

Question 7 3/4 Q7 ☐0 ☐0.5 ☐1.0 ☐1.5 ☐2.0 ☐2.5 [✓ 3.0] ☐3.5 ☐4 RESERVED

Minimal cut capacity: 20



Edmonds-Karp paths list:

- s, a, t
- s, a, b, t

Maximal flow between s and t: 10 ✓

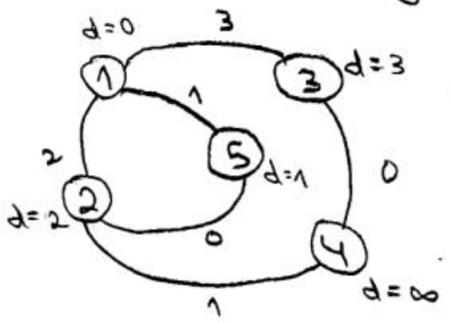
As arestas c→s, c→b e c→t podem ser reduzidas até ao seu valor mínimo, pois nunca são explorados pelo algoritmo. Não existe nenhum caminho entre s e t que seja possível passando por essas arestas.

✓

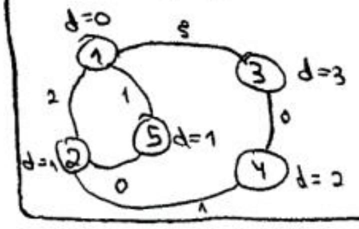
Question 8 0.5/2

Q8 □0 [✓ 0.5] □1.0 □1.5 □2 RESERVED

Dijkstra after initialization step:



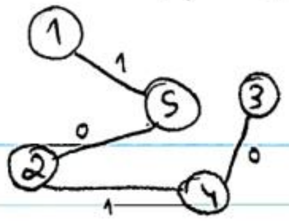
Final graph (dijkstra):



a) ✓ x

b) x

shortest paths tree:

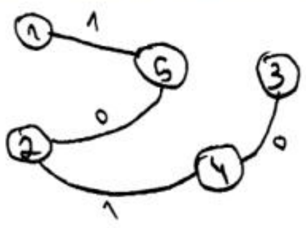


Order in wich the nodes are examined:

$1 \to 3 \to 5 \to 2 \to 4$

c) ✓ ++

Resulting MST (Prim)



Os algoritmos do Dijkstra e Prim são iguais, mas para encontrar coisas diferentes. Dijkstra encontra o caminho curto e Prim a MST. Contudo, Dijkstra funciona com grafos direcionados e não direcionados e Prim não. Além disso, Prim pode lidar com pesos negativos, ao contrário do Dijkstra.

+115/12/1+

Question 9 0/2

Q9 [✓ 0] 0.5 1.0 1.5 2 RESERVED

[illegible]