Universidade Federal de Pernambuco
Centro de Informática
Bach. em Ciência da Computação

IF672cc Algoritmos e Estr. de Dados 2019.2
Prof. Paulo Fonseca

Segunda Prova — 21 de Novembro de 2019

■ QUESTÃO 1 (2,0pt)

Um grafo $G = (V, E)$ é dito bipartido se seus vértices podem ser coloridos com duas cores, vermelho (R) e verde (G), de forma que nenhuma aresta ligue dois vértices com a mesma cor. Uma BFS pode receber as seguintes modificações para testar se um grafo é bipartido.

- Comece com todos os vértices sem cor (U).
- A visita a um vértice u só é iniciada se ele ainda não tiver cor. Nesse caso, a visita começa pintando-o de vermelho (R).
- Ao desenfileirar um vértice u e considerar as arestas (u, v):
  - Se v ainda não tem cor, pinte-o com a cor oposta à cor de u e enfileire-o.
  - Se v já estiver pintado com uma cor oposta a de u, ignore-o.
  - Se v já estiver colorido com a mesma cor de u, pára e retorna `FALSE` (o grafo não é bipartido).
- Se o precurso chegar até o final com todos os vértices coloridos normalmente, encerra e retorna `TRUE` (o grafo é bipartido).

Ilustre a execução desse algoritmo sobre o grafo dado pelas listas de adjacências a seguir, completando o quadro abaixo.

$0 \to 2, 3$	$3 \to 0$	$6 \to 4, 7, 8$
$1 \to 4, 7$	$4 \to 1, 6$	$7 \to 1, 5, 6$
$2 \to 0$	$5 \to 7$	$8 \to 6$

Passo	Fila	Cores								
		0	1	2	3	4	5	6	7	8
0	$\varnothing$	U	U	U	U	U	U	U	U	U
1	0	R	U	U	U	U	U	U	U	U
2	$2, \cdots$	R	U	G	$\cdots$					
$\vdots$	$\vdots$					$\vdots$				

Indique claramente ao final se o grafo é bipartido.

■ QUESTÃO 2 (2,0pt)

A estrutura *union-find* também pode ser empregada para determinar se um grafo $G = (V, E)$ com vértices $0, 1, \ldots, n-1$ é bipartido através do seguinte algoritmo.

Algoritmo *is_bipartite*
Entrada $G = (V = (0, \ldots, n-1), E)$

```
1  S ← union-find com cada vértice isolado numa componente.
2  F = (f_0, ..., f_{n-1}) ← (⊥, ..., ⊥)
3  para cada aresta (u, v) ∈ E faça
4      r_u ← find(S, u)
5      r_v ← find(S, v)
6      se r_u = r_v então
7          devolva FALSE
8      se F[u] = ⊥ então
9          F[u] ← v
10     senão
11         union(S, F[u], v)
12     se F[v] = ⊥ então
13         F[v] ← u
14     senão
15         union(S, u, F[v])
16 devolva TRUE
```

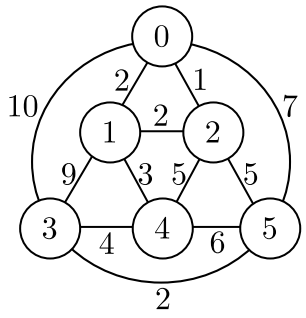
Ilustre a execução desse algoritmo sobre o grafo da Questão 1, exibindo a union-find e o vetor F ao final da execução, considerando as heurísticas da *união ponderada* e *compressão de caminhos*.

■ QUESTÃO 3 (2,0pt)

Complete o quadro abaixo

Iteração	Distância/Precursor					
	0	1	2	3	4	5
0 (início)	(0/?)	∞/?	∞/?	∞/?	∞/?	∞/?
$\vdots$			$\vdots$			

correspondente à execução do *Algoritmo Dijkstra* sobre o grafo a seguir, considerando o vértice 0 como a origem.

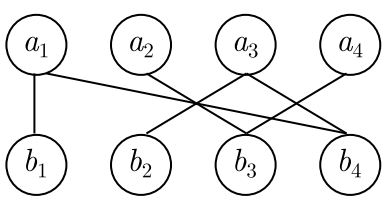


■ **QUESTÃO 4** (2,0pt)

Um grafo $G = (V, E)$ é dito *tripartido* se seus vértices puderem ser coloridos com três cores, vermelho (R), verde (G), e azul (B), de forma que nenhuma aresta ligue dois vértices com a mesma cor. Infelizmente, determinar se um grafo é tripartido é um problema NP-completo. Podemos usar um algoritmo de *backtracking* para atribuir progressivamente uma cor a cada vértice, retrocedendo sempre que vértices vizinhos tenham a mesma cor. Ilustre a execução desse algoritmo sobre o grafo da Questão 3 (desconsiderando os pesos das arestas), exibindo a árvore de execução correspondente ao percurso do espaço de soluções. Indique claramente se o grafo é tripartido e, em caso positivo, indique a coloração obtida na árvore.

■ **QUESTÃO 5** (2,0pt)

Considere um grafo bipartido $G = (V, E)$, com $2n$ vértices, sendo $V = A \cup B$, onde $A = (a_1, \ldots, a_n)$ e $B = (b_1, \ldots, b_n)$. Cada aresta do grafo é da forma (a_i, b_j), ou seja, cada vértice de A está ligado apenas a 0 ou mais vértices de B e vice versa, como neste exemplo para $n = 4$.



Dizemos que duas arestas (a_i, b_j) e (a_p, b_q) são *cruzadas* quando ($i < p$ e $j > q$), ou ($i > p$ e $j < q$). Por exemplo, no grafo acima, as arestas (a_1, b_4) e (a_3, b_2) são cruzadas.

Um *emparelhamento* é um subconjunto de arestas disjuntas de G. Considere o problema de encontrar o tamanho do maior emparelhamento de G sem arestas cruzadas. Defina $S(i, j)$ como a solução do problema considerando o subgrafo restrito aos vértices $(a_1, \ldots, a_i) \cup (b_1, \ldots, b_j)$, ou seja, apenas os i primeiros vértices de A e j primeiros vértices de B. Assim, temos

$S(0, j) = S(i, 0) =$ _____(a).

Para $i, j > 0$, temos três casos:

1) Se o emparelhamento máximo não contém nenhuma aresta incidente ao vértice a_i, temos
   $S(i, j) =$ _____(b);
2) Se o emparelhamento máximo não contém nenhuma aresta incidente ao vértice b_j, temos
   $S(i, j) =$ _____(c);
3) Se $(a_i, b_j) \in E$ e o emparelhamento máximo contém essa aresta, temos
   $S(i, j) =$ _____(d).

Combinando esses três casos, que cobrem todas as possibilidades, temos

$S(i, j) =$ _____(e).

Dessa forma, podemos resolver esse problema através de um algoritmo usando a técnica de _____(f), preenchendo uma matriz $D_{(n+1)\times(n+1)}$ uma linha por vez, onde $D[i, j] = S(i, j)$.

(g) Ilustre a execução desse algoritmo sobre o grafo acima exibindo a matriz S computada e indicando claramente a solução final.

■ ■ ■

BOAS FESTAS E FELIZ 2020!