Revisão - Prova 2

Algoritmos e Estrutura de Dados - IF672

Andreywid Yago Lima de Souza <ayls>
João Victor Nascimento <jvsn2>

Centro de Informática UFPE
Universidade Federal de Pernambuco

O que veremos hoje:

- Encaminhamento em Grafos
  - BFS e DFS
  - MST
    - Kruskal
    - Prim
  - Menor caminho:
    - Dijkstra
    - Bellman-Ford
- Problemas NP-completo
  - Backtracking
- Programação Dinâmica
  - Knapsack

Busca em largura

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 vértcie 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.

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

		Cores								
Passo	Fila	0	1	2	3	4	5	6	7	8
0	∅	U	U	U	U	U	U	U	U	U
1	0	R	U	U	U	U	U	U	U	U
⋮	⋮					⋮				

Indique claramente ao final se o grafo é bipartido.

Busca em profundidade:

Considere o seguinte grafo representado por listas de adjacências

$0 \rightarrow (1,3),\ (2,5),\ (3,9)$
$1 \rightarrow (0,3),\ (2,3),\ (3,5),\ (4,7)$
$2 \rightarrow (0,5),\ (1,3),\ (3,2),\ (4,6),\ (5,8)$
$3 \rightarrow (0,9),\ (1,5),\ (2,2),\ (4,2),\ (5,3)$
$4 \rightarrow (1,7),\ (2,6),\ (3,2),\ (5,5)$
$5 \rightarrow (2,8),\ (3,3),\ (4,5)$

onde cada elemento numa dessas listas $u \rightarrow \ldots\ (v,w)\ \ldots$ indica uma aresta de u para v com peso w.

a) Forneça a enumeração em ***profundidade*** dos nós desse grafo.

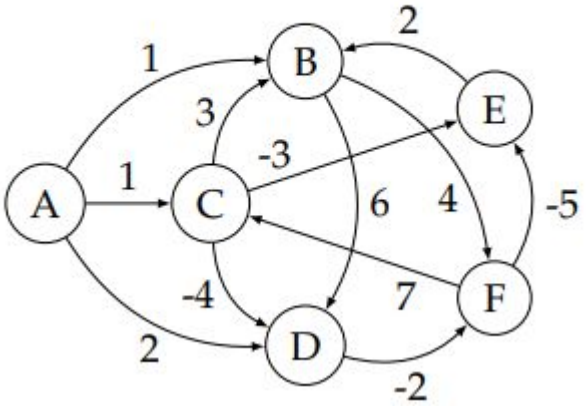
Algoritmo Prim

b) Complete o diagrama a seguir correspondente à execução do ***Algoritmo Prim*** sobre esse grafo.

Iter. #	Peso, Precursor					
	0	1	2	3	4	5
0	0, –	∞, ?	∞, ?	∞, ?	∞, ?	∞, ?
1	0, –	3, 0	...	...	...	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮

Bellman-Ford

Considere o grafo

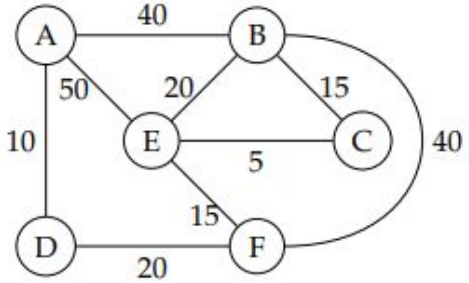


Preencha o diagrama a seguir, correspondente à execução do ***Algoritmo Bellman-Ford*** sobre o grafo acima a partir do vértice de origem A. Indique se o grafo possui ciclos negativos com base na execução do algoritmo.

Iter. #	Distância, Precursor					
	A	B	C	D	E	F
0	0, –	∞, ?	∞, ?	∞, ?	∞, ?	∞, ?
1	0, –	1, A	···	···	···	···
⋮	⋮	⋮	⋮	⋮	⋮	⋮

Dijkstra

Considere o grafo



Preencha o diagrama a seguir, correspondente à execução do ***Algoritmo Dijkstra*** sobre o grafo acima a partir do vértice de origem *A*.

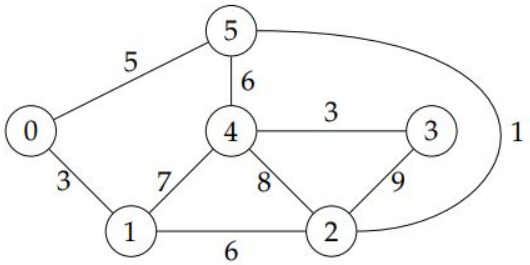
Iter. #	Distância, Precursor					
	A	B	C	D	E	F
0	0, –	∞, ?	∞, ?	∞, ?	∞, ?	∞, ?
1	0, –	40, *A*	...	...	...	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮

Kruskal

O Algoritmo de Kruskal é um algoritmo guloso para encontrar uma árvore geradora de custo mínimo (MST) de um grafo ponderado que faz uso da estrutura de dados *union-find* da seguinte maneira:

1. Inicie com a MST vazia e com cada vértice u numa componente isolada da union-find, `make_set(`u`);`
2. Ordene as arestas por 1º. peso, 2º. menor vértice, e 3º. maior vértice;
3. Para cada aresta (u, v) com peso w na ordem acima, se u e v não estão na mesma componente, una-as `union(`u`,`v`)`, e adicione a aresta à MST.

Considerando a execução do Algoritmo Kruskal sobre o grafo



represente graficamente a estrutura *union-find* C ao final da execução do algoritmo, considerando o emprego das heurísticas de *união ponderada* e *compressão de caminhos*.

PD - Knapsack

Escreva a Matriz de Programação Dinâmica correspondente à execução do Algoritmo ***Knapsack-0/1*** sobre os itens a seguir

Item (i)	0	1	2	3	4
Peso (w_i)	4	3	1	2	2
Valor (v_i)	40	25	10	20	15

com capacidade $K = 7$. Indique os itens escolhidos através da correspondente sequência de entradas da matriz de PD.

Dijkstra

Considere o diagrama a seguir correspondente a uma execução do Algoritmo Dijkstra.

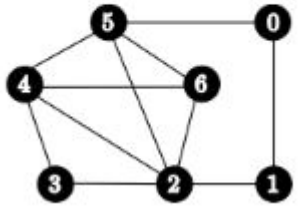
	A	B	C	D	E	F
0	(0)	∞	∞	∞	∞	∞
1	(0)	(15)	∞	30	20	∞
2	(0)	(15)	55	25	(20)	∞
3	(0)	(15)	45	(25)	(20)	35
4	(0)	(15)	45	(25)	(20)	(35)
5	(0)	(15)	(40)	(25)	(20)	(35)

a) Represente o grafo dirigido com 6 vértices e 8 arestas correspondente a essa execução do algoritmo.
b) Represente também o vetor de precursores correspondentes a essa execução.

Backtracking

Um algoritmo baseado em ***backtracking*** para o problema de encontrar um k-clique num grafo G consiste em iniciar com um conjunto vazio de vértices C e ir acrescentando progressivamente vértices ao conjunto, interrompendo o processo assim que um vértice adicionado não esteja ligado a algum outro vértice de C. O primeiro vértice é escolhido na ordem $0, 1, 2, \ldots$ Uma vez acrescentado um vértice v, os próximos a serem tentados são os vizinhos ainda não escolhidos de v, enquanto houver. Se, em algum momento, o algoritmo consegue escolher k vértices com sucesso, ele pára e retorna o clique C.

Ilustre a árvore de execução desse algoritmo backtracking para a o grafo a seguir para $k = 4$.



Busca em profundidade

Considere a codificação do mapa do Brasil a seguir como um grafo não dirigido nos qual cada estado é representado por um nó e dois estados vizinhos são ligados por uma aresta.

Qual o percurso adequado nesse grafo para determinar a menor quantidade de estados $\delta(s,t)$ a serem visitados numa viagem de Pernambuco ($s = 14$) a cada um dos demais estados, $\delta(s = 14, t)$ para $t = 1, \ldots, 27$?

a) Ilustre a execução desse percurso exibindo a árvore correspondente. b) Enumere os estados de acordo com esse percurso e c) Indique o menor número de estados visitados numa viagem de Pernambuco ($s = 14$) ao Paraná ($t = 25$).

Centro de Informática UFPE

Universidade Federal de Pernambuco

Boa Prova !