Lista de Exercícios - Grafos

UNIMONTES PPGMCS
Projeto e Análise de Algoritmos 1º semestre de 2015
Professor: Renê Rodrigues Veloso Data: 04/05/2015

1 Exercícios

1. Dado um DAG (V, E) e dois vértices $u, v \in V$, se há aresta entre u e v, podemos dizer que uEv, pois E é uma relação em V^2. Suponha um subconjunto S de V, pede-se:
   (a) Dê o conjunto $in(S)$, como sendo o conjunto de todas as arestas que chegam em S.
   (b) Dê o conjunto $out(S)$, como sendo o conjunto de todas as arestas que saem de S.
2. Dado um DAG G como o par ordenado (V, E), indique como é o par ordenado G^{-1}, i. e., o DAG onde as arestas são invertidas.
3. Dado um dígrafo (V, E) e um vértice $u \in V$, o componente fortemente conectado que inclui u, denotado por $SCC(u) = \{v \in V \mid ...\}$. Complete.
4. Verifique se é possível obter a ordenação topológica de um DAG qualquer, usando o método remoção de raizes. Nesse método, a cada iteração os vértices que não possuem arestas incidentes são removidos primeiro. Uma vez removido um vértice, todas as arestas que saem dele também são removidas, surgindo novas raizes.
5. Interprete o enunciado abaixo e complete:

   Qualquer vértice em um componente pode ser escolhido como o "representante" do componente. Formalmente, temos um subconjunto V_{DAG} de V e um mapeamento $r : V \to V_{\text{DAG}}$ tal que para todo $(u, v) \in V^2, SCC(u) = SCC(v) \Rightarrow \exists w \in SCC(u) \mid r(u) = r(v) = w$. Dado tal mapeamento r, o DAG (associado com o dígrafo), como exemplificado na Figura 1, é definido como:

   DAG Associado: Dado um dígrafo (V, E), o DAG associado (V, E_{DAG}) é $(V, ...)$

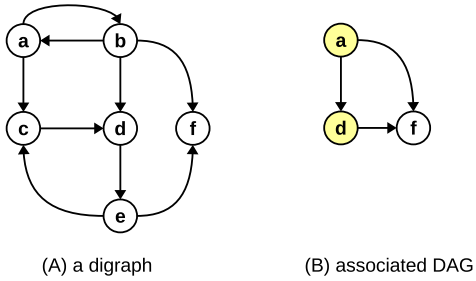

Figure 1: Em (A) um digrafo e, em (B), o DAG associado onde r(a) = r(b) = a, r(c)= r(d)= r(e) = d e r(f) = f.

6. Explique: Dado um DAG (V, E_{DAG}), uma ordenação topológica $<$ dele é uma ordem total de V tal que para todo $(r(u), r(v)) \in E_{\text{DAG}}, r(u) < r(v)$. Uma consequencia dessa definição é que, dado um DAG (V, E_{DAG}), sua ordenação topológica $<$ e dois vértices $u, v \in V^2, r(u)\ E^*_{\text{DAG}}\ r(v) \Rightarrow r(u) = r(v)$ ou existe $(r(u), r(k)) \in E_{\text{DAG}}$ tal que $r(k) \leq r(v)$ e $r(k)\ E^*_{\text{DAG}}\ r(v)$. (o operador $*$ é *estrela de kleene*, procure saber o que é isso)

7. Dados dois vértices $(r(u), r(v))$ s.t. $u, v \in V^2$, o conjunto $\{r(w) \mid w \in V$ and $r(u)\ E^*_{\text{DAG}}\ r(w)$ and $r(w)\ E^*_{\text{DAG}}\ r(v)\}$ é denotado por $P(r(u), r(v))$. Com suas palavras, o que é $P(r(u), r(v))$?

8. Escreva um algoritmo para, dado um dígrafo qualquer (V, E), seu conjunto de componentes fortemente conectados $SCCS$ (conseguido pelo algoritmo de Tarjan, por exemplo) e uma nova aresta $(u, v) \in E$, insira (u, v) no grafo e verifique se essa nova aresta faz a junção de dois componentes. Se sim, o conjunto $SCCS$ deve ser atualizado.

9. Escreva um algoritmo para, dado um dígrafo $G = (V, E)$ qualquer, transforme G em $G_{\text{DAG}} = (V, E_{\text{DAG}})$ de forma que:

   - identifique os componentes fortemente conectados.
   - toda aresta que sai de algum componente em G, deve agora sair do representante desse componente em G_{DAG}.
   - toda aresta que chega em algum componente em G, deve agora chegar no representante desse componente em G_{DAG}.
   - se já houver aresta $(r(u), r(v))$ em E_{DAG}, outras arestas u, v em E não deve aparecer em E_{DAG}.

   A Figura 2 representa essa transformação. Tente escrever um algoritmo que use o máximo de definições dos exercícios anteriores.

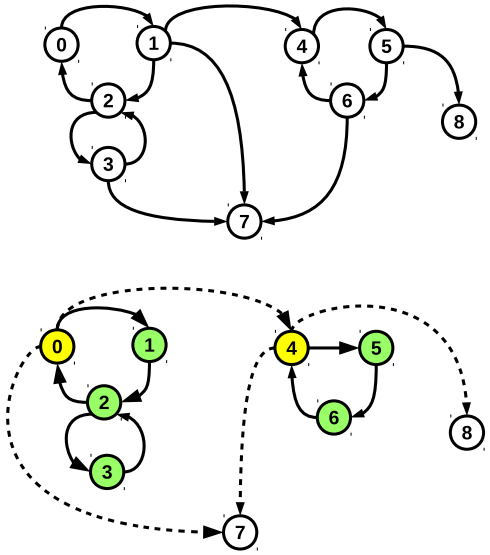

Figure 2: Um dígrafo e o seu DAG. Os vértices em amarelo são os representantes de cada componente. Os vértices 7 e 8 também são representantes de componentes de tamanho 1. As arestas tracejadas são as novas arestas de E_{DAG}.

10. Escreva um algoritmo que, dado um dígrafo transformado pelo exercício anterior, os conjuntos de vértices, arestas, representantes e uma aresta (u, v) qualquer, verifique se u e v estão no mesmo componente. Se sim, remova a aresta (u, v) e retorne *true* se u ainda alcançar v dentro do componente ou *false*, caso contrário.

11. É possível resolver o problema a seguir representando-o como um grafo. Dê um algoritmo que faça isso.

    Charlie acquired airline transport company and to stay in business he needs to lower the expenses by any means possible. There are N pilots working for his company (N is even) and N/2 plane crews needs to be made. A plane crew consists of two pilots - a captain and his assistant. A captain must be older than his assistant. Each pilot has a contract granting him two possible salaries - one as a captain and the other as an assistant. A captain's salary is larger than assistant's for the same pilot. However, it is possible that an assistant has larger salary than his captain. Write a program that will compute the minimal amount of money Charlie needs to give for the pilots' salaries if he decides to spend some time to make the optimal (i.e. the cheapest) arrangement of pilots in crews.

    Input

The first line of input contains integer N, $2 \leq N \leq 10,000$, N is even, the number of pilots working for the Charlie's company. The next N lines of input contain pilots' salaries. The lines are sorted by pilot's age, the salaries of the youngest pilot are given the first. Each of those N lines contains two integers separated by a space character, XY, $1 \leq Y < X \leq 100,000$, a salary as a captain (X) and a salary as an assistant (Y).

Output

The first and only line of output should contain the minimal amount of money Charlie needs to give for the pilots' salaries.

Sample

```
input
4
5000 3000
6000 2000
8000 1000
9000 6000

output
19000

input
6
10000 7000
9000 3000
6000 4000
5000 1000
9000 3000
8000 6000

output
32000
```

12. Dizemos que o algoritmo de Tarjan é mais eficiente do que o algoritmo de Kosaraju. De maneira informal (pois ainda não estudamos complexidade de algoritmos), explique o motivo.

13. O que é a fase de relaxamento do algoritmo de Dijkstra?

14. Escreva o algoritmo de Bellman-Ford. Em que ele difere do algoritmo de Dijkstra?

15. Incremente a Union-Find inserindo uma nova função $disunion(x)$, que separa x do conjunto a qual pertence. O novo conjunto representado

por x deve conter todos os elementos ligados a x (que não foram comprimidos pela função Find).