EP1 DE MAC0328 ALGORITMOS EM GRAFOS
FLORESTA DE BLOCOS E VÉRTICES DE CORTE

Data de entrega: 12 de abril

Neste exercício-programa, sua tarefa é construir um programa que, dado um grafo $G = (V, E)$, calcula a floresta de blocos e vértices de corte de G. Ou seja, seu programa deve construir o grafo $G' = (V', E')$ sobre $V' := \mathcal{B} \cup V_c$, onde $\mathcal{B}$ é o conjunto de blocos de G e V_c é o conjunto de vértices de corte de G, e $E' := \{ Bv : B \in \mathcal{B},\ v \in V_c \cap B \}$. Para manter este texto autocontido, listamos as definições apropriadas na seção 3.

Seu código deve ser escrito na linguagem `C` e deve consumir tempo $O(n+m)$ quando alimentado com um grafo G com n vértices e m arestas. Seu programa passará por um corretor automático, sendo portanto essencial que ele obedeça rigidamente o formato de entrada e saída descritos na seção 1.

Além disso, seu código deve ser bem legível, organizado e modular. Leia atentamente a seção 2 acerca de detalhes de implementação antes de planejar a organização de seu programa.

1. Formatos da entrada e saída

1.1. **Entrada.** Seu programa deve ler da entrada padrão. A entrada terá o formato descrito a seguir; veja um exemplo mais adiante. A primeira linha conterá um único número inteiro n, indicando que o grafo G tem conjunto de vértices $V := \{0, \ldots, n-1\}$. Seguem-se n linhas correspondentes aos vértices $0, \ldots, n-1$, nesta ordem, cada uma descrevendo os vizinhos do vértice.

A linha correspondente ao vértice $u \in V$ é formada por inteiros separados por espaços. Ela começa com um inteiro $\deg(u)$, o grau de u em G. Seguem-se $\deg(u)$ inteiros nesta mesma linha, $v_{u,1} < \cdots < v_{u,\deg(u)} \in V \setminus \{u\}$, indicando que as arestas incidentes em u são $\{uv_{u,1}, \ldots, uv_{u,\deg(u)}\}$.

O exemplo abaixo representa o grafo descrito na figura 1.

Exemplo de entrada

```
6
2 1 2
2 0 2
3 0 1 3
3 2 4 5
2 3 5
2 3 4
```

Data: 11 de março de 2015.

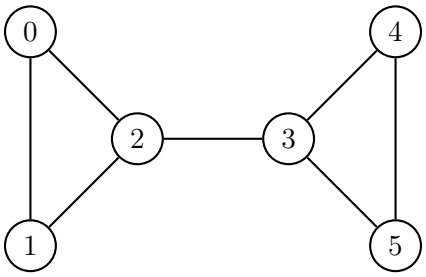

FIGURA 1. Exemplo de grafo G para entrada.

1.2. **Saída.** O conjunto de vértices V' de G' deverá ser representado da seguinte forma. Tome $n' \coloneqq |V'|$. Teremos $V' = \{0, \ldots, n' - 1\}$, onde os primeiros $|\mathcal{B}|$ vértices (de 0 a $|\mathcal{B}| - 1$) correspondem a blocos, e os demais correspondem a elementos de V_c, possivelmente renomeados.

A saída deve começar com uma única linha contendo o número n, seguida por n linhas, correspondentes aos vértices $0, \ldots, n - 1$, nesta ordem. A linha correspondente ao vértice $u \in V$ é formada por inteiros separados por espaços. Ela deve começar com o inteiro $\deg(u)$, seguido por $\deg(u)$ inteiros, $b_{u,1}, \ldots, b_{u,\deg(u)} \in \{0, \ldots, |\mathcal{B}| - 1\}$, onde $b_{u,i}$ é o número do bloco que induz a aresta $uv_{u,i}$; veja descrição da entrada.

Após estas $1+n$ linhas, deve ser impressa uma linha contendo o número $|V_c|$. A essa deverão se seguir $|V_c|$ linhas, cada uma contendo dois inteiros separados por espaços, correspondendo a cada elemento de V_c, em qualquer ordem. Cada linha terá como primeiro inteiro um elemento v de V_c, seguido pelo vértice de V' que corresponde a v; lembre-se que tal vértice é um inteiro em $\{|\mathcal{B}|, \ldots, n' - 1\}$.

Finalmente, você deverá imprimir o grafo G' no mesmo formato da entrada. Veja um exemplo a seguir, acompanhado do grafo G' correspondente na figura 2.

Exemplo de saída

```
6
2 0 0
2 0 0
3 0 0 1
3 1 2 2
2 2 2
2 2 2
2
2 3
3 4
5
1 3
2 3 4
1 4
2 1 0
2 1 2
```

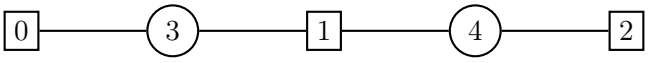

FIGURA 2. Exemplo de grafo G para entrada.

2. Detalhes de implementação e entrega

Sua implementação deve organizar as estruturas de dados e sua manipulação de forma modular e encapsulada. Apesar de a correção ser automática, a organização de seu código também será avaliada. Faz parte da sua tarefa encontrar um ponto de equilíbrio entre um programa totalmente desorganizado, que só tem variáveis globais, e um programa totalmente encapsulado, só com variáveis locais, mas incrivelmente difícil de ler e/ou muito ineficiente.

Seu programa também deve ser cuidadoso com o uso da memória. Você deve alocá-la usando a família `malloc` de funções e deve liberar todo espaço alocado no *heap* antes de terminar a execução. Seu programa será testado usando o Valgrind.

Você deverá fornecer um `Makefile` para gerar seu executável, com o nome `blocks`, obrigatoriamente pelo GCC (a versão usada será $\geq$ 4.8.2), a partir da invocação '`make`'. Dentre as opções de compilação, inclua os argumentos

```
-Wall -pedantic -std=c99 -g
```

Note que, no padrão `C99`, variáveis não precisam ser declaradas no início de um bloco de escopo; faça bom uso desse *feature*.

Para a entrega, comprima um diretório contendo todos os seus arquivos em um único arquivo no formato `tar.gz`. Os nomes tanto do diretório como do arquivo de entrega (quando a extensão é removida) devem ser seu número USP.

Programas que não seguirem rigorosamente os formatos de entrada e saída terão suas notas punidas severamente. O mesmo se aplica para programas que não seguirem as instruções desta seção.

3. Definições

Seja $G = (V, E)$ um grafo. Se $uv \in E$ é uma aresta de G, dizemos que v é um *vizinho* de u, e que a aresta uv *incide em* u. O *grau* de $u \in V$ é o número de arestas de G incidentes em u.

Dizemos que G é *conexo* se, para quaisquer $u, v \in V$, vale que $u \underset{G}{\rightsquigarrow} v$. Dizemos que $C \subseteq V$ é um *componente (conexo)* de G se $C \subseteq V$ é maximal tal que $G[C]$ é conexo.

Se $U \subseteq V$, o *subgrafo de G induzido por U* é $G[U] \coloneqq (U, E[U])$, onde $E[U] \coloneqq \{\, uv \in E : u, v \in U\}$. Para cada vértice $v \in V$, definimos $G - v \coloneqq G[V \setminus \{v\}]$. Dizemos que $v \in V$ é um *vértice de corte* de G se $G - v$ possui mais componentes que G.

Um subconjunto $B \subseteq V$ é um *bloco* de G se $B \subseteq V$ é maximal tal que $G[B]$ é conexo e não tem vértice de corte.