Projetos e Análise de Algoritmos – 2024/2 – Prova II (20 pontos)
Prof. Marcio Costa Santos

INSTRUÇÕES

(a) As respostas devem estar na prova no espaço designado para tal.

(b) A interpretação das questões faz parte da prova. Explique as suposições que fizer.

(c) Não é permitido o uso de material de consulta.

Nome:

Matrícula:

Questão 1 (4 pontos) (Grafos) Considere a seguinte sugestão para um algoritmo de busca: Vamos ordenar todos os vértices em ordem crescente de grau e visitamos os vértices seguindo esta ordem. Este algoritmo é apresentado abaixo:

Algoritmo 1: Visita Maior

Data: Grafo $G = (V, E)$
para todo $v \in V$ **faça**
  calcule $d(v)$;
  $visitados[v] = BRANCO$;
Ordene os vértices de G em ordem crescente de grau;
para todo $v_i \in V$ *segundo a ordem de graus* **faça**
  $visitados[v] = PRETO$;

Considere que a complexidade de ordenar os vértices segundo seu grau é $O(n \log n)$

a) Qual a complexidade desse algoritmo quando o grafo é representado como uma lista de adjacências.
Na lista de adjacências podemos calcular os graus com apenas uma passada pela lista encadeada onde armazenamos os vizinhos de cada vértice, dessa forma precisamos de $O(d(v))$ para calcular o grau de cada vértice e logo $O(m)$ para calcular todos os graus de todos os vértices. A ordenação é realizada em tempo $O(n \log n)$, logo a complexidade total do método é $O(n \log n + m)$

b) Qual a complexidade desse algoritmo quando o grafo é representado como uma matriz de adjacências.
Na matriz de adjacências podemos calcular os graus passando pela linha que representa sua vizinhança, dessa forma precisamos de $O(n)$ para calcular o grau de cada vértice e logo $O(n^2)$ para calcular todos os graus de todos os vértices. A ordenação é realizada em tempo $O(n \log n)$, logo a complexidade total do método é $O(n \log n + n^2) = O(n^2)$.

Questão 2 (4 pontos) (DFS e BFS) Um aluno da UFMG desenvolveu uma nova estrutura de dados para representar um grafo. Ele não explicou como a estrutura é implementa, mas garante que podemos fazer todas as operações usuais em grafos e que elas tem as seguintes complexidades:

Verificar se uma aresta está ou não no grafo : $\Theta(\log n)$;

Recuperar uma lista com a vizinhaça de um vértice : $\Theta(\log n)$;

a) Qual a complexidade do algoritmo de busca em largura com essa nova estrutura usada para representar o grafo de entrada?
Vamos analisar

Algoritmo 2: BFS(G,s)

Entrada: Grafo $G = (V, s)$, vértice inicial s.
para v *até* $V(G) - \{s\} O(n)$ **faça**
  $color[v] \leftarrow BRANCO O(1)$;
  $\pi[v] = u O(1)$;
  $[v] = \infty O(1)$;
$color[s] \leftarrow CINZA O(1)$;
$\pi[s] = s O(1)$;
$[s] = 0 O(1)$;
Fila $Q \leftarrow \emptyset O(1)$;
`Enfila(`Q`,`s`)`$O(1)$;
enquanto $Q \neq \emptyset$ **faça**
  $u \leftarrow$ `Desenfila(Q)`$O(1)$;
  Calcular $N(u) O(\log n)$;
  para todo $v \in N(u) O(d(v))$ **faça**
    se $color[v] = BRANCO$ **então**
      $color[v] = CINZA O(1)$;
      $d[v] = d[u] + 1 O(1)$;
      $\pi[v] = u O(1)$;
      `Enfila(`Q`,`v`)`$O(1)$;
  $color[u] \leftarrow PRETO O(1)$;

Com isso a complexidade fica:

$$O(n)(O(1)) + O(1) + \sum_{v \in V(G)} (O(1) + O(\log n) + O(d(v)).O(1) + O(1)) =$$

$$O(n) + \sum_{v \in V(G)} (O(\log n) + O(d(v))) =$$

$$O(n) + \sum_{v \in V(G)} O(\log n) + \sum_{v \in V(G)} O(d(v)) =$$

$$O(n) + O(n \log n) + \sum_{v \in V(G)} O(d(v)) =$$

$$O(n) + O(n \log n) + O(m) = O(m + n \log n)$$

b) Qual a complexidade do algoritmo de busca em profundidade com essa nova estrutura usada para representar o grafo de entrada?
Vamos analisar

Algoritmo 3: DFS(G,s)

Entrada: Grafo $G = (V, s)$, vértice inicial s.
para v *até* $V(G) O(n)$ **faça**
  $color[v] \leftarrow BRANCO$ $O(1)$;
  $\pi[u] = \lambda$ $O(1)$;
$time \leftarrow 0$ $O(1)$;
para $v \in V(G)$ **faça**
  se $color[v] == BRANCO O(1)$ **então**
    `DFS-VISIT(G,u)` $O(1)$;

Algoritmo 4: DFS-VISIT(G,v)

$time \leftarrow time + 1 O(1)$;
$color[v] = CINZA O(1)$;
$i[v] = time O(1)$;
Calcular $N(v) O(\log n)$;
para $u \in N(v) O(d(v))$ **faça**
  se $color[u] = BRANCO$ **então**
    $\pi[u] = v O(1)$;
    `DFS-VISIT(G,u)` chamada recursiva;
$time \leftarrow time + 1 O(1)$;
$color[v] = PRETO O(1)$;
$f[v] = time O(1)$;

$$O(n).(O(1) + O(1)) + O(1) + \sum_{v \in V(G)} (O(1) + O(1) + O(1) + O(\log n) + O(d(v))) =$$

$$O(n).O(1) + O(1) + \sum_{v \in V(G)} (O(\log n) + O(d(v))) =$$

$$O(n) + \sum_{v \in V(G)} (O(\log n) + O(d(v))) =$$

$$O(n) + O(n \log n) + \sum_{v \in V(G)} O(d(v)) = O(n \log n + m)$$

Questão 3 (4 pontos) (Caminho Mínimo) Assuma que nós modificamos a maneira como calculamos o peso de um caminho. Ao invés de somar os pesos das arestas do caminho, nós consideramos agora apenas a arestas mais pesada daquele caminho. Ou seja, o peso de uma caminho é dado pela aresta mais pesada do caminho. Neste contexto, dada uma **ÁRVORE** ponderada $T = (V, E)$ com pesos $w(u, v)$ positivos em suas arestas. Apresente um algoritmo para determinar os caminhos mínimos entre todos vértices e um vértice fixo s com essa nova regra de cálculo de peso.

Observe que em uma árvore nós temos apenas um caminho entre quaisquer par de vértices. Assim podemos enraizar a árvore no vértice s e determinar a aresta mais pesada do caminho até todos os demais simplesmente com uma busca (largura ou profundidade). Na busca em largura por exemplo, podemos armazenar a aresta mais pesada do caminho em cada um dos vértices da árvore. O algoritmo abaixo exemplifica isso.

Algoritmo 5: BFS-modificado(T,s)

Entrada: Árvore $T = (V, s)$, vértice inicial s, pesos $w[uv]$ para cada aresta $uv \in E(T)$.
para v *até* $V(T) - \{s\}$ **faça**
  $color[v] \leftarrow BRANCO$;
  $d[v] = -\infty$;
$color[s] \leftarrow CINZA$;
$d[s] = -\infty$;
Fila $Q \leftarrow \emptyset$;
`Enfila(`Q`,`s`)`;
enquanto $Q \neq \emptyset$ **faça**
  $u \leftarrow$ `Desenfila(Q)`;
  Calcular $N(u)$;
  para todo $v \in N(u)$ **faça**
    se $color[v] = BRANCO$ **então**
      $color[v] = CINZA$;
      se $w[u, v] \geq d[v]$ **então**
        $d[v] \leftarrow w[u, v]$;
      `Enfila(`Q`,`v`)`;
  $color[u] \leftarrow PRETO$;

Questão 4 (4 pontos) (Árvore Geradora Mínima) Considere o seguinte problema: Um construtor de rodovias deve construir vias para interligar n cidades. Ele dispõe da distância entre cada par de cidades $d(v, u)$ em quilômetros e possui também uma função $f(u, v)$ que representa o preço de pavimentar cada quilômetro entre as cidades u e v. O construtor deseja garantir que exista um caminho entre quaisquer duas cidades e que as rodovias construídas tenham o menor custo possível.

a) Como podemos modelar esse problema como um problema em grafos?
Vamos utilizar os vértices para representar as cidades, assim temos n vértices. Temos todas as arestas possíveis e estas são ponderadas. Os pesos das arestas representam o valor gasto para construir aquela via ligado as duas cidades. Dessa forma, a aresta (u, v) custa $d(v, u) * f(v, u)$. Assumindo que as rodovias são todas de mãos duplas, o que desejamos é determinar uma árvore geradora de custo mínimo desse grafo.

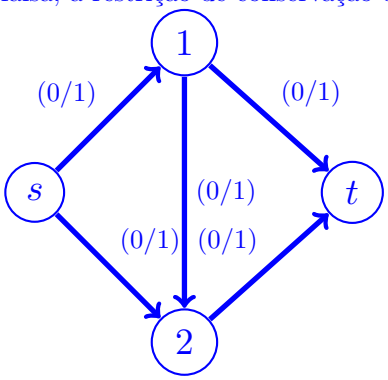
b) Apresente um algoritmo para resolver esse problema. Lembre-se que você pode usar os algoritmos vistos em sala e apenas indicar as possíveis modificações neles.
Podemos usar o algoritmo de Prim ou de Kruskal para determinar a árvore geradora mínima, apenas com a correção dos pesos para os valores que calculamos.

Questão 5 (4 pontos) (Fluxo) Considere uma rede $G = (V, A)$ com capacidades $c(u, v)$ para todos os arcos. Sejam f_1 e f_2 dois fluxos sobre esta rede e defina a *subtração* de dois fluxos $f_1 - f_2$ como sendo

$$(f_1 - f_2)(u, v) = \max\{0, f_1(u, v) - f_2(u, v)\}$$

Prove ou refute: A subtração de dois fluxos é um fluxo válido. Caso a afirmação seja falsa, explique qual a restrição é violada.

A afirmação é falsa, a restrição de conservação de fluxo pode ser violada nesta operação. Para isso observe a rede abaixo:



Considere agora os dois fluxos seguintes:

f_1 :

$$\begin{aligned} f_1(s, v_1) &= 1 \\ f_1(s, v_2) &= 0 \\ f_1(v_1, v_2) &= 1 \\ f_1(v_1, t) &= 0 \\ f_1(v_2, t) &= 1 \end{aligned}$$

.

f_2 :

$$\begin{aligned} f_2(s, v_1) &= 1 \\ f_2(s, v_2) &= 0 \\ f_2(v_1, v_2) &= 0 \\ f_2(v_1, t) &= 0 \\ f_2(v_2, t) &= 1 \end{aligned}$$

.