3ª prova de Algoritmos e Teoria dos Grafos

2° semestre de 2005

---

(1) LEIA COM ATENÇÃO TODA A PROVA. Entender o enunciado das questões faz parte da prova.

(2) EXPLIQUE e JUSTIFIQUE todas as suas respostas. Você pode usar os teoremas e algoritmos vistos nas aulas. Nesses casos, escreva o que você está usando.

(3) A prova é sem consulta, termina 17:20hs. A nota da prova é no máximo cem.

---

1) (**20**) Dê as respostas dos *find*'s (linhas 10 e 11) no seguinte algoritmo supondo que a representação usada para conjuntos disjuntos é árvore usando compressão de caminhos e união por tamanho.

1: **para** i de 1 até 16
2:     $make(x_i)$
3: **para** i de 1 até 15 passo 2
4:     $union(x_i, x_{i+1})$
5: **para** i de 1 até 13 passo 4
6:     $union(x_i, x_{i+2})$
7: $union(x_1, x_5)$
8: $union(x_1, x_{13})$
9: $union(x_1, x_{10})$
10: $find(x_2)$
11: $find(x_9)$

2) (**30**) Lembrando que quando usamos representação de conjuntos disjuntos por árvores com união-por-tamanho e compressão de caminhos o custo de m operações *make/find/union*, sendo n makes, é no pior caso $O(m \lg^* n)$, determine a complexidade do algoritmo de Kruskal:

$M \leftarrow \emptyset$;
$F \leftarrow$ arestas em ordem não-decrescente de peso;
para $(v \in V)$
    $make(v)$;
para $(uv \in F)$
    se $(find(u) \neq find(v))$
        $S = S \cup \{uv\}$;
        $union(u, v)$;
Devolva M

3) (**20**) Dado que os pesos das arestas num grafo conexo são inteiros de 1 até W, com W uma constante qualquer, o tempo de pior caso do algoritmo de Prim pode ser melhorado para $O(|V|+|E|)$. Descreva como isso pode ser feito (não precisa escrever o algoritmo, basta uma descrição detalhada da idéia) e argumente justificando a complexidade $O(|V|+|E|)$.

4) (**50**) O *centro* de um grafo G é um vértice $u \in V(G)$ tal que

$$\max_{v \in V(G)} \operatorname{dist}(u, v)$$

é mínimo.

(a) (**10**) Mostre que toda árvore T com pelo menos 3 vértices tem os mesmos centros que a árvore T' obtida de T removendo-se todas as folhas.

(b) (**15**) Deduza de (a) que toda árvore tem ou exatamente um centro ou dois centros adjacentes.

(c) (**25**) Dscreva a idéia de um algoritmo com complexidade linear no número de vértices para determinar o(s) centro(s) de uma árvore. Justifique a complexidade.

4)a) Para cada vértice v os vértices mais distantes de v sao as folhas da árvore. Removendo as folhas as maiores distências ficam diminuidas de 1, poratno os centros são os mesmos.

b) Por indução no numero de vértices

Se a árvore tem um vértice, então tem um centro.
Se a árvore tem 2 vertices, então tem 2 centrso.
Se a árvore tem mais que dois vértices, removemos as folhas e, por indução, a árvore resultante tem um ou dois centros. Devolvendo as folhas e usando o item a) concluímos que a árvore original tem 1 ou 2 centros.

c)

(1)Primeiro construímos uma lista com todas as folhas.
(2)Removemos as folhas construindo uma lista das novas folhas criadas
quando remover uma folha verifique se o vértice adjacente tem grau 1
se sim coloque-o na lista das novas folhas
(3) Repita até restar 1 ou 2 folhas.

Cada etapa é linear no número de folhas portanto o algoritmo é linear no número de vértices.

3) Implementamos um fila de prioridade usando um vetor de listas ligadas indexado por 1..W.
Na posição i estão os vértices de peso (chave) i.
Percorremos a lista ligada até acha a primeira lista não vazia, que temo vértice de menor peso. **O(W) que é constante**.
Esse vértice entra pra árvore e se for o caso atualiza-se as chaves de cada vizinho, se for o caso

O tempo do Prim (veja as notas de aula) é O(a|V|+b|E|) onde a é o tempo do extraimin -- O(1) nesse caso -- e b o tempo de uma atualização -- O(1) nesse caso.

Portanto, O(|V|+|E|)

2) O número de operções é m=
|V| makes + 2|E| finds + |V|-1 unions = 2|E|+2|V|-1 que custa
O((|V| + 2|E| + |V| - 1)log*|V|)= O((|V|+|E|)log*|V|)=O(|E|log*|V|) pois
o grafo é conexo (|E|>=|V|-1)
mais o O(|E|log|E|) da ordenação resulta em O(|E|log|E|).

1) A esposta é qualquer Xi desde que i =1 ..8