SEGUNDA PROVA DE ALGORITMOS EM GRAFOS
BCC, 1o. SEMESTRE DE 2005

1. [2 pontos] Seja $G = (V, E)$ um grafo 2-aresta-conexo, isto é, um grafo conexo, com pelo menos dois vértices, e sem arestas de corte. Suponha que executamos uma busca em profundidade (BeP) a partir de um vértice r de G. Suponha que obtemos a árvore de BeP $T = (V, A)$. Claramente, r pode ser considerado uma 'raiz' natural para T. Oriente todas as arestas em A de r para as folhas, e oriente todas as arestas em $E \setminus A$ de forma 'ascendente'. Obtemos assim um grafo dirigido $\vec{G}$. Decida se as seguintes asserções são corretas. **Justifique suas respostas.**
   (i) Em $\vec{G}$, todo vértice $x \in V$ é acessível a partir de r.
   (ii) Em $\vec{G}$, o vértice r é acessível a partir de todo vértice $x \in V$.
   (iii) Em $\vec{G}$, para quaisquer vértices x e $y \in V$, o vértice y é acessível a partir de x.
2. [2 pontos] Considere o algoritmo de Warshall para a determinação do fecho transitivo de um grafo dirigido, visto em sala:

```
for (i = 0; i < V; i++)
  for (s = 0; s < V; s++)
    for (t = 0; t < V; t++)
      if (A[s][i] && A[i][t]) A[s][t] = 1;
```

   (Seguindo a convenção usual, a matriz de adjacência `A` tem `1`s na diagonal.) Suponha que executamos este algoritmo no circuito dirigido $\vec{C}^n$ com n vértices; suponha que os vértices de $\vec{C}^n$ são $0, \dots, n-1$ e os arcos de $\vec{C}^n$ são $(0, 1), \dots, (n-2, n-1), (n-1, 0)$.
   (i) Suponha que $n = 10$. Desenhe os grafos dirigidos correspondentes à matriz `A` imediatamente antes da iteração em que $i = 0$, $i = 1$, $i = 2$, e $i = 5$ (ignore os laços, isto é, arcos da forma (v, v)).
   (ii) Descreva a matriz `A` após o término da execução do algoritmo de Warshall no caso geral (isto é, para n genérico).
3. [2 pontos] Considere a função recursiva vista em sala dada abaixo, que determina uma ordenação topológica do DAG `D`:

```
void TSdfsR(Dag D, int v, int ts[])
  { int w;
    pre[v] = 0;
    for (w = 0; w < D->V; w++)
      if (D->adj[w][v] != 0)
        if (pre[w] == -1) TSdfsR(D, w, ts);
    ts[cnt0++] = v;
  }
```

   (i) Escreva um pseudocódigo que usa a função `TSdfsR()` acima e determina o comprimento máximo de um caminho dirigido em `D`.
   (ii) Traduza seu pseudocódigo em um trecho de código em `C`.
   Você deve fazer tanto (i) quanto (ii), pois sua resposta para (i) pode ser mais fácil de entender e você perderá pouco por eventuais pequenos erros de implementação em (ii), desde que sua resposta para (i) esteja correta.
4. [2 pontos] Considere o Programa 19.10 visto em sala, que implementa o algoritmo de Kosaraju:

```
static int post[maxV], postR[maxV];
static int cnt0, cnt1;
void SCdfsR(Graph G, int w)
  { link t;
    G->sc[w] = cnt1;
    for (t = G->adj[w]; t != NULL; t = t->next)
      if (G->sc[t->v] == -1) SCdfsR(G, t->v);
    post[cnt0++] = w;
  }
int GRAPHsc(Graph G)
  { int v; Graph R = GRAPHreverse(G);
    cnt0 = 0; cnt1 = 0;
    for (v = 0; v < G->V; v++) R->sc[v] = -1;
    for (v = 0; v < G->V; v++) if (R->sc[v] == -1) SCdfsR(R, v);
    cnt0 = 0; cnt1 = 0;
    for (v = 0; v < G->V; v++) G->sc[v] = -1;
    for (v = 0; v < G->V; v++) postR[v] = post[v];
    for (v = G->V-1; v >= 0; v--)
      if (G->sc[postR[v]] == -1)
        { SCdfsR(G, postR[v]); cnt1++; }
    GRAPHdestroy(R);
    return cnt1;
  }
int GRAPHstrongreach(Graph G, int s, int t)
  { return G->sc[s] == G->sc[t]; }
```

Seja G o grafo dirigido com 12 vértices e 19 arcos dado por

```
 0:  1  3      |       6:  7  8
 1:  2         |       7:  5
 2:  0  5      |       8:  9 11
 3:  4  6      |       9: 10
 4:  3  9      |      10:  8
 5:  6  8      |      11: 10
```

(i) Desenhe G.

(ii) Determine as componentes fortemente conexas de G.

(iii) Qual é o valor inteiro devolvido por `GRAPHsc()`, quando executado sobre G? O que é esse valor em geral (isto é, para um grafo dirigido genérico G)?

(iv) Determine os vetores `postR[]` e `G->sc[]` computados pelo Programa 19.10 para o grafo G acima. Como uma simulação detalhada pode tomar muito tempo, você pode simplesmente determinar um possível vetor `postR[]`. (De fato, como `GRAPHreverse()` não está especificada acima, o vetor `postR[]` não está univocamente determinado.)

5. [2 pontos] Seja $G = (V, E)$ um grafo conexo e $c\colon E \to \mathbb{R}$ uma função custo nas arestas de G. Suponha que c seja injetora, isto é, que todas as arestas de G tenham pesos diferentes. Prove que há uma única árvore geradora de peso mínimo em (G, c). [*Sugestão.* Sejam $T = (V, A)$ e $T' = (V, A')$ duas AGMs de peso mínimo. Seja e a aresta de peso mínimo em $(A \setminus A') \cup (A' \setminus A)$ e suponha que $e \in A$. Considere o circuito fundamental $C(e, T')$.]