Dodatna nastava iz programiranja 2008/2009
Prirodno Matematički Fakultet, Niš
datum: 11. april 2009. godine
predavač: Nikola Milosavljević
e-mail: nikola5000@gmail.com

Osnovni grafovski algoritmi - teorija

1 Predstavljanje grafova u računaru

1.1 Matrica susedstva

U matrici susedstva je adj[u, v] = 1 (true) ako postoji grana između u i v, inače adj[u, v] = 0 (false).

```
readln(n, m);
fillchar(adj, sizeof(adj), false);
for i := 1 to m do begin
   readln(a, b, w1);
   adj[a, b] = true; w[a, b] = w1;
   adj[b, a] = true; w[b, a] = w1;        { ako je graf neorjentisan }
end;
```

1.2 Simulacija liste

Susedi čvora u se nalaze u nizu **adj** na pozicijama od p[u] do p[u + 1] - 1.

```
readln(n, m);
fillchar(deg, sizeof(deg), 0);        { stepeni čvorova, na početku su 0 }
for i := 1 to m do begin
   readln(a[i], b[i], w[i]);
   deg[a[i]] := deg[a[i]] + 1;
   a[i + m] = b[i]; b[i + m] = a[i]; w[i + m] = w[i];     { ako je graf neorjentisan }
   deg[a[i + m]] := deg[a[i + m]] + 1;
end;

QSort(1, 2*m);
for i := 1 to 2*m do adj[i] := b[i];
p[1] := 1;
for i := 2 to n + 1 do
   p[i] = p[i - 1] + deg[i - 1];

{ QSort sortira niz a, dok prilikom razmene menja a[i], b[i] i w[i] }
```

Umesto QSorta, može se iskoristiti Counting Sort:

```
for i := 1 to 2*m do begin
   adj[p[a[i]]] := b[i];
   w1[p[a[i]]] := w[i];
   p[a[i]] := p[a[i]] + 1;
end;
```

2 Obilazak grafa

Pretpostavlja se da je graf zadat preko liste suseda (susedi čvora u se nalaze u nizu **adj** na pozicijama od p[u] do p[$u + 1$] - 1). Niz **mark** označava da li je dati čvor obiđen ili još ne, dok **anc**[v] predstavlja prethodnika čvora v u obilasku grafa (čvor iz koga smo došli u v). **queue** predstavlja red u kome smeštamo čvorove prilikom BFS obilaska, dok promenljive **first** i **last** ukazuju na početak, odnosno kraj reda.

2.1 DFS

```
procedure DFS(u :  longint)
var
   i, v :  longint
begin

   mark[u] := true;

   for i := p[u] to p[u + 1] - 1 do begin
      v = adj[i];
      if (not mark[v]) then begin
         anc[v] := u;
         DFS(v);
      end;
   end;

end;
```

2.2 BFS

```
procedure BFS(start :  longint)
var
   i, u, v, first, last :  longint
begin

   queue[1] := start;
   mark[start] := true;
   first := 0; last := 1;

   while (first < last) do begin
      first := first + 1;
      u := queue[first];
      for i := p[u] to p[u + 1] - 1 do begin
         v := adj[i];
         if (not mark[v]) then begin
            mark[v] := true;
            last := last + 1;
            queue[last] := v;
            anc[v] := u;
         end;
      end;
   end;

end;
```