ASD Laboratorio 04

The A(SD)-Team

UniTN

2026-11-19

Calendario

24/09	Introduzione
22/10	Ad-hoc
29/10	Grafi 1
19/11	Grafi 2
26/11	Grafi 3 + presentazione progetto
03/12	Lab Progetto 1

Progetto grafi

- Da giovedì 26 novembre a venerdì 04 dicembre (consegna ore 18:00);
- Iscrizione dei gruppi al progetto entro **mercoledì 25 novembre**: `https://bit.ly/ASDprog_2026-2027` (dovete essere loggati con l'account UniTN)

SOLUZIONI: VISITA-ALBERO

- Fate riferimento alla slide del prof. Montresor sulle visite di alberi: `https://disi.unitn.it/~montreso/asd/slides/05-alberi.pdf`.

FUNZIONI RICORSIVE **PRE-, IN-, POST-**ORDER

```
static void preorder(Node *n) {
    if (!n) return;
    out << n->data << ' ';
    preorder(n->left);
    preorder(n->right);
}
static void inorder(Node *n) {
    if (!n) return;
    inorder(n->left);
    out << n->data << ' ';
    inorder(n->right);
}
static void postorder(Node *n) {
    if (!n) return;
    postorder(n->left);
    postorder(n->right);
    out << n->data << ' ';
}
```

SOLUZIONI: VISITA

- Basta una qualunque visita (in ampiezza, in profondità, ecc...)
- Implementazioni sul sito:

  `https://asdlab.disi.unitn.it/slides/asd26/soluzioni_lab03.zip`.

Implementazione DFS ricorsiva - input

```
struct nodo{
  vector<int> vic;
  bool visited = false;
};
...
vector<nodo> grafo;
...
in >> N;
grafo.resize(N);
for(int i=0;i<M;i++){
  int from, to;
  in >> from >> to;
  grafo[from].vic.push_back(to);
}
```

Implementazione DFS ricorsiva

Con grafo variabile globale (*caveat emptor*):

```
void visit(int n){
  grafo[n].visited=true;
  count++;
  for(int v:grafo[n].vic)
    if(!grafo[v].visited)
       visit(v);
}
```

Passando grafo per reference:

```
void visit(int n, vector<nodo>& grafo, int& count) {
  grafo[n].visited = true;
  count++;
  for(int v:grafo[n].vic)
    if(!grafo[v].visited)
       visit(v, grafo);
}
```

IMPLEMENTAZIONE DFS CON STACK

```
stack<int> s;
s.push(S);
while(!s.empty()){
  int n = s.top();
  s.pop();
  // se un nodo è già stato visitato lo ignoriamo!
  if(grafo[n].visited) { continue; }
  grafo[n].visited = true;
  count++;
  // visita tutti i vicini
  for(int v: grafo[n].adj) {
    s.push(v);
  }
}
```

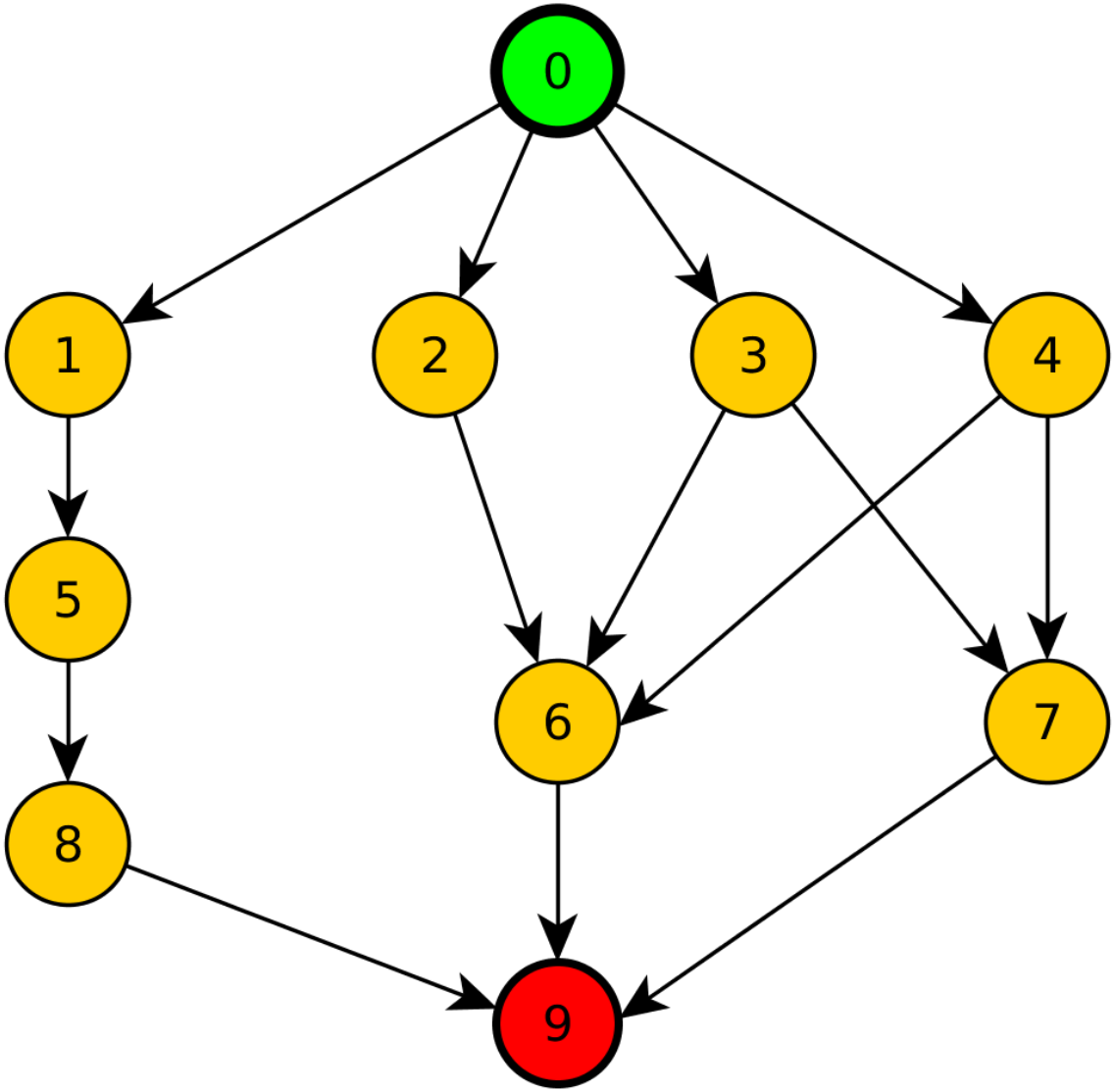
IMPLEMENTAZIONE BFS CON CODA

```
queue<int> q;
q.push(S);
while(!q.empty()) {
  int n = q.front();
  q.pop();
  // se un nodo è già stato visitato lo ignoriamo!
  if(grafo[n].visited) { continue; }
  grafo[n].visited = true;
  count++;
  // visita tutti i vicini
  for(int v: grafo[n].adj) {
    q.push(v);
  }
}
```

SOLUZIONI: DIAMETRO

- Per ogni nodo di partenza, trovare il nodo più lontano.
- La distanza in un grafo non pesato si può calcolare utilizzando una `BFS`.

Esempio: cammini minimi



Esercizi (I)

Alberi Red-Black (red-black)

Dati N valori interi positivi distinti, costruire un albero Red-Black inserendo gli N valori dati.

Individuazione di un ciclo in un grafo orientato (ciclo)

Dato un grafo orientato, connesso e che ha al più un ciclo, determinare se il grafo contiene un ciclo oppure no

Numero di cammini minimi (numcammini)

Dato un grafo diretto ed una coppia di nodi S e T, trovare quanti **diversi cammini di lunghezza minima** ci sono fra S e T.

Esercizi (II)

Progetto grafi a. a. 2017/2018

007: Pioggia di laser (`laser`)