Corso di Algoritmi e Strutture Dati (IN110) – Prof. Marco Liverani – a.a. 2024/2025

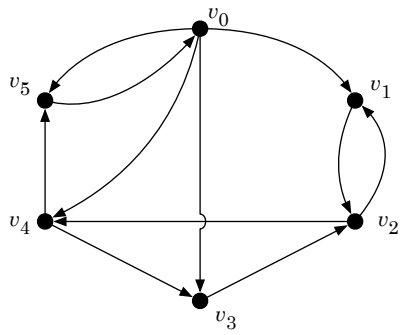
Esame scritto del 22 Gennaio 2025 (Appello A)

Si richiede di risolvere entrambi gli esercizi riportando una codifica in linguaggio C completa dei due programmi. Nel caso in cui non si riesca a completare entrambi gli esercizi si suggerisce di riportare almeno la codifica in C delle funzioni principali o una loro pseudo-codifica. È possibile consultare libri e appunti personali, ma non scambiare libri o appunti con altri studenti. I compiti che presenteranno evidenti ed anomale «similitudini» saranno annullati. La prova scritta ha una durata di tre ore, durante le quali non è consentito allontanarsi dall'aula, se non dopo aver consegnato il compito.
Deve essere consegnata solo la «bella copia» del compito scritto; su ciascun foglio deve essere riportato il ***nome****, il* ***cognome*** *e il* ***numero di matricola*** *dello studente.*

Esercizio n. 1

Leggere in input le liste di adiacenza di un grafo $G = (V, E)$ con $|V| = n$ vertici. Letto in input un intero $k > 0$, con $0 < k < n$, costruire una lista L di k interi casuali compresi nell'insieme $\{0, 1, \dots, n-1\}$. Stampare in output la lista. Verificare che, per ogni spigolo $(u, v) \in E(G)$ risulti $u \in L$ o $v \in L$.

Esempio Si consideri il grafo $G = (V, E)$ in figura.



La lista $L = 2 \to 0 \to 4$ è composta da vertici che verificano proprietà richiesta dall'esercizio (copertura di vertici), mentre la lista $L' = 0 \to 1 \to 5 \to 4$ non soddisfa tale proprietà perché entrambi gli estremi dello spigolo $(2, 3) \in E(G)$ non appartengono alla lista.

Soluzione

```c
#include <stdlib.h>
#include <stdio.h>
#define MAX 50

struct nodo {
  int info;
  struct nodo *next;
};

struct nodo *leggiLista(void) {
  int i, n;
  struct nodo *p, *primo = NULL;
  printf("numero di elementi della lista: ");
  scanf("%d", &n);
  printf("inserisci gli elementi della lista: ");
  for (i=0; i<n; i++) {
```

```
    p = malloc(sizeof(struct nodo));
    scanf("%d", &p->info);
    p->next = primo;
    primo =p;
  }
  return primo;
}

int leggiGrafo(struct nodo *V[]) {
  int i, n;
  printf("numero di vertici del grafo: ");
  scanf("%d", &n);
  for (i=0; i<n; i++) {
    printf("lista di adiacenza del vertice %d\n", i);
    V[i] = leggiLista();
  }
  return n;
}

void stampaLista(struct nodo *p) {
  while (p != NULL) {
    printf("%d --> ", p->info);
    p = p->next;
  }
  printf("Null\n");
  return;
}

void stampaGrafo(struct nodo *V[], int n) {
  for (int i=0; i<n; i++) {
    printf("%2d: ", i);
    stampaLista(V[i]);
  }
  return;
}

int isNotIn(struct nodo *p, int i) {
  int rc = 0;
  while (p != NULL && p->info != i)
    p = p->next;
  if (p == NULL)
    rc = 1;
  return rc;
}

int verificaCopertura(struct nodo *V[], int n, struct nodo *L) {
  struct nodo *p;
  int i, rc = 1;
  for (i = 0; i < n && rc == 1; i++) {
    if (isNotIn(L, i)) {
      p = V[i];
      while (p != NULL && isNotIn(L, p->info))
        p = p->next;
      if (p == NULL)
        rc = 0;
    }
  }
  return rc;
```

```
}

int main(void) {
  int n;
  struct nodo *V[MAX], *L;
  n = leggiGrafo(V);
  L = leggiLista();
  stampaGrafo(V, n);
  stampaLista(L);
  if (verificaCopertura(V, n, L) == 1)
    printf("La lista L e' una copertura di vertici di G\n");
  else
    printf("La lista L NON e' una copertura di vertici di G\n");
  return 0;
}
```

Esercizio n. 2

Letti in input due numeri interi $n, m > 0$, costruire una matrice A di n righe ed m colonne con numeri interi casuali compresi in $\{0, 1, 2, \ldots, 20\}$. Stampare in output la matrice A. Stampare poi tutti i numeri dell'insieme $\{0, 1, 2, \ldots, 20\}$ che non sono presenti nella matrice A.

Esempio Siano $n = 4$ e $m = 6$. Si consideri la seguente matrice A:

$$A = \begin{pmatrix} 17 & 12 & 9 & 3 & 8 & 20 \\ 9 & 14 & 8 & 0 & 12 & 12 \\ 13 & 15 & 15 & 6 & 17 & 8 \\ 12 & 11 & 8 & 4 & 3 & 5 \end{pmatrix}$$

I numeri dell'insieme $\{0, 1, 2, \ldots, 20\}$ che non sono contenuti in A sono i seguenti: $1, 2, 7, 10, 16, 18, 19$.

Soluzione

```
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#define MAX 50

void numeriNonPresenti(int A[MAX][MAX], int n, int m) {
  int i, j, k, trovato;
  printf("Nella matrice non sono presenti i seguenti elementi: ");
  for (k=0; k<=20; k++) {
    trovato = 0;
    for (i=0; i < n && trovato == 0; i++)
      for (j=0; j < m && trovato == 0; j++)
        if (A[i][j] == k)
          trovato = 1;
    if (trovato == 0)
      printf("%d ", k);
  }
  printf("\n");
  return;
}

void matriceCasuale(int A[MAX][MAX], int n, int m) {
  srand((unsigned)time(NULL));
  for (int i=0; i<n; i++)
```

```
    for (int j=0; j<m; j++)
      A[i][j] = rand() % 21;
  return;
}

void stampaMatrice(int A[MAX][MAX], int n, int m) {
  for (int i=0; i<n; i++) {
    for (int j=0; j<m; j++)
      printf("%2d ", A[i][j]);
    printf("\n");
  }
  return;
}

int main(void) {
  int m, n, A[MAX][MAX];
  printf("Inserisci due interi positivi n e m: ");
  scanf("%d %d", &n, &m);
  matriceCasuale(A, n, m);
  stampaMatrice(A, n, m);
  numeriNonPresenti(A, n, m);
  return 0;
}
```