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

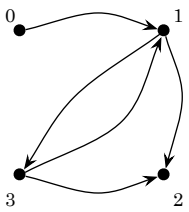
Esame scritto del 18 Luglio 2025 (Appello D)

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.*

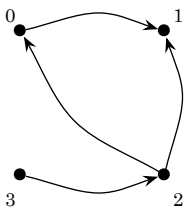
Esercizio n. 1

Leggere in input un grafo orientato con n vertici e rappresentarlo con liste di adiacenza. Leggere in input un intero $k > 0$ e generare una lista L di k numeri interi casuali tali che ogni elemento di L appartenga all'insieme $\{0, 1, \ldots, n-1\}$. Partendo dalla prima coppia, per ogni coppia di elementi consecutivi di L, $u \to v$, effettuare la seguente operazione: se v non è adiacente ad u in G, aggiungere v alla lista di adiacenza di u, altrimenti eliminare v dalla lista di adiacenza di u. Dopo aver eseguito le $k-1$ operazioni sul grafo in input, stampare le liste di adiacenza del grafo ottenuto.

Esempio Sia $n = 4$ e sia G il grafo riportato in fig. (a). Sia $k = 6$ e sia $L = 2 \to 1 \to 3 \to 1 \to 2 \to 0$ la sequenza di numeri interi casuali. Allora il grafo ottenuto modificando le liste di adiacenza di G in base alla sequenza L, è rappresentato in fig. (b).



(a)



(b)

Soluzione

```c
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#define MAX 100

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

int adiacente(struct nodo *G[], int u, int v) {
  struct nodo *p;
  int r;
```

```
  p = G[u];
  while (p != NULL && p->info != v)
    p = p->next;
  if (p == NULL)
    r = 0;
  else
    r = 1;
  return r;
}

struct nodo *aggiungi(struct nodo *p, int x) {
  struct nodo *q;
  q = malloc(sizeof(struct nodo));
  q->info = x;
  q->next = p;
  return q;
}

struct nodo *elimina(struct nodo *p, int x) {
  struct nodo *q, *primo;
  primo = p;
  if (p != NULL && p->info == x) {
    primo = p->next;
    free(p);
  } else {
    while (p != NULL && p->next != NULL) {
      if (p->next->info == x) {
        q = p->next;
        p->next = p->next->next;
        free(q);
      }
      p = p->next;
    }
  }
  return primo;
}

void modificaGrafo(struct nodo *G[], int n, struct nodo *p) {
  while (p != NULL && p->next != NULL) {
    if (p->info != p->next->info) {
      if (adiacente(G, p->info, p->next->info))
        G[p->info] = elimina(G[p->info], p->next->info);
      else
        G[p->info] = aggiungi(G[p->info], p->next->info);
    }
    p = p->next;
  }
  return;
}

struct nodo *generaLista(int n) {
  int i, k;
  struct nodo *p, *primo = NULL;
```

```
67    srand((unsigned)time(NULL));
68    printf("Numero di elementi della lista: ");
69    scanf("%d", &k);
70    for (i=0; i<k; i++) {
71      p = malloc(sizeof(struct nodo));
72      p->info = rand() % n;
73      p->next = primo;
74      primo = p;
75    }
76    return primo;
77  }
78
79  struct nodo *leggiLista(void) {
80    struct nodo *p, *primo = NULL;
81    int i, n;
82    printf("Numero di elementi: ");
83    scanf("%d", &n);
84    printf("Inserisci %d vertici: ", n);
85    for (i=0; i<n; i++) {
86      p = malloc(sizeof(struct nodo));
87      scanf("%d", &p->info);
88      p->next = primo;
89      primo = p;
90    }
91    return primo;
92  }
93
94  int leggiGrafo(struct nodo *G[]) {
95    int i, n;
96    printf("Numero di vertici: ");
97    scanf("%d", &n);
98    for (i=0; i<n; i++) {
99      printf("Lista di adiacenza di %d\n", i);
100     G[i] = leggiLista();
101   }
102   return n;
103 }
104
105 void stampaLista(struct nodo *p) {
106   while (p != NULL) {
107     printf("%d --> ", p->info);
108     p = p->next;
109   }
110   printf("NULL\n");
111   return;
112 }
113
114 void stampaGrafo(struct nodo *G[], int n) {
115   int i;
116   for (i=0; i<n; i++) {
117     printf("G[%d]: ", i);
118     stampaLista(G[i]);
119   }
```

```
  return;
}

int main(void) {
  struct nodo *L, *G[MAX];
  int n;
  n = leggiGrafo(G);
  L = generaLista(n);
  stampaLista(L);
  modificaGrafo(G, n, L);
  stampaGrafo(G, n);
  return 0;
}
```

Esercizio n. 2

Letti in input due interi positivi n e k, con $n > k$, costruire un array A composto da n numeri interi casuali compresi nell'insieme $\{1, 2, \ldots, 100\}$. Stampare l'array A e la sotto-sequenza di k elementi di A la cui media aritmetica sia minima.

Esempio Siano $n = 10$ e $k = 4$. Consideriamo il seguente array di numeri casuali

$$A = (a_0 = 8, a_1 = 6, a_2 = 20, a_3 = 4, a_4 = 5, a_5 = 12, a_6 = 3, a_7 = 17, a_8 = 6, a_9 = 1)$$

La sotto-sequenza la cui media aritmetica è minima è $(a_3 = 4, a_4 = 5, a_5 = 12, a_6 = 3)$.

Soluzione

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

void arrayCasuale(int A[], int n) {
  int i;
  srand((unsigned)time(NULL));
  for (i=0; i<n; i++)
    A[i] = rand() % 100 + 1;
  return;
}

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

float media(int A[], int k) {
  float m = 0;
  int i;
  for (i=0; i<k; i++)
    m = m + A[i];
```

```
  m = m / k;
  return m;
}

int main(void) {
  int n, k, i, A[MAX], iMin=0;
  float m, minMedia;
  printf("Inserisci due interi positivi n e k: ");
  scanf("%d %d", &n, &k);
  arrayCasuale(A, n);
  stampaArray(A, n);
  minMedia = media(A, k);
  for (i=1; i<=n-k; i++) {
    m = media(&A[i], k);
    printf("m = %4.2f min = %4.2f\n", m, minMedia);
    if (m < minMedia) {
      minMedia = m;
      iMin = i;
    }
  }
  printf("il sotto-array di media minima e':\n");
  for (i=iMin; i<iMin+k; i++)
    printf("%2d ", A[i]);
  printf("\n");
  return 0;
}
```