Esame di Metodi Computazionali della Fisica I
30 novembre 2009

C++

Nell'esercizio di C^{++} è prevista l'implementazione di una classe *Network* per simulazioni su grafi aleatori.

Un network (o grafo) è definito a partire da un insieme di nodi V ed un insieme di edge E. Diremo dunque che il nodo $i \in V$ è connesso al nodo $j \in V$ se $(i, j) \in E$. Il modo più semplice per rappresentare un network è attraverso la sua matrice di adiacenza $\mathcal{A}$ di dimensioni $N \times N$ tale che $\mathcal{A}_{ij} = 1$ solo se il nodo i è connesso al nodo j, e zero altrimenti.

Nella prima parte dell'esame sarà dunque necessario implementare una classe che rappresenti una generica matrice di adiacenza $\mathcal{A}$ e tutte le funzioni di supporto (aggiunta di un edge, rimozione di un edge, ecc ..) a fissato numero di nodi (ma definibile dall'utente con allocazione dinamica). Nella seconda parte verrà chiesto di implementare funzioni in grado di manipolare i network creati precedentemente.

Ogni parte è suddivisa in punti i quali possono essere obbligatori o facoltativi. I punti obbligatori debbono essere svolti in sequenza per il corretto funzionamento della libreria mentre i punti facoltativi (se non esplicitamente indicato) possono essere svolti in qualsiasi ordine. Il codice deve essere **correttamente compilabile** e accompagnato dal rispettivo **Makefile** funzionante (se il Makefile è assente saranno decurtati dei punti). Non saranno accettati codici non compilabili (abbiate cura di commentare la parte di codice che non riuscite sviluppare. In caso di indecisione sul voto tale parte sarà comunque parzialmente valutata). Il codice deve rispondere correttamente alle richieste previste nel main. In sede di valutazione il codice sarà comunque testato per verificarne l'affidabilità.

I punti obbligatori sono:

I Parte (a.1), (a.2), (a.3), (a.4)

I Parte (b.1), (b.2), (b.3), (b.4), (b.5), (b.6)

I Parte (c.1), (c.2)

II Parte (d.1), (d.2), (d.3)

III Parte (e.1)

1 I Parte: classe Network

1.1 Definizione della classe Network

(a) Costruttori/Distruttori:

(1 - Obb) Implementare il costruttore per la classe Network che accetti in ingresso un intero (**int nodes**) e che allochi la memoria corretta per tutta la matrice di adiacenza.

(2 - Obb) Implementare il costruttore di copia per la classe Network.

(3 - Obb) Implementare il distruttore con la relativa liberazione di memoria.

(4 - Obb) La variabile **nodes** non può essere modificata direttamente dall'esterno. Implementare una funzione che restituisca all'esterno il numero dei nodi.

TIP: Poiché la matrice può assumere solo due valori $\{0, 1\}$, è possibile implementarla come matrice di **bool** (per risparmiare spazio). Ma possono essere anche usati gli **int**.

TIP: Usare **type **_net** per la matrice di adiacenza, dove **type** è il tipo che avete scelto.

TIP: Nel costruttore non inserire alcun controllo sulla validità di nodes.

TIP: ALLOCARE e DEALLOCARE correttamente tutta la memoria, in qualsiasi costruttore e distruttore.

(b) Modifica del Network: Il numero di edge (**int edges**) è modificato ogni qual volta si aggiunge o si elimina un edge.

(1 - Obb) La variabile **edges** non può essere modificata direttamente dall'esterno. Implementare una funzione che restituisca all'esterno il numero di edge.

(2 - Obb) Quando si chiama il costruttore standard quanti sono gli edge? Settare nel costruttore implementato in (a.1) il numero corretto di edge.

(3 - Obb) Quando si chiama il costruttore di copia quanti sono gli edge? Settare nel costruttore implementato in (a.2) il numero corretto di edge.

(4 - Obb) Implementare una funzione **Check_Edge** che accetta in ingresso due interi i e j e mi restituisce **true** se il nodo i è connesso al nodo j, altrimenti **false**.

(5 - Obb) Implementare una funzione **Add_Edge** che accetta in ingresso due interi i e j. Se l'edge è già presente restituisce **false** altrimenti aggiunge l'edge e restituisce **true**.

(6 - Obb) Implementare una funzione **Remove_Edge** che accetta in ingresso due interi i e j. Se l'edge non è presente restituisce **false** altrimenti elimina l'edge e restituisce **true**.

TIP: Modificare in modo opportuno **edges** ogni qual volta si agisce sul numero di edge.

(c) Operatori:

(1 - Obb) Implementare l'operatore **operator=** che accetta in ingresso un **Network**.

(2 - Obb) L'unione di due network è quel network che possiede gli edge (presi una volta sola) di entrambi i network. Implementare l'operatore **operator+** che accetta in ingresso un **Network** e restituisce un **Network**.

(3) Implementare l'operatore **operator+=**.

(4) Implementare l'operatore **operator==** che restituisce **true** se i network sono identici, altrimenti **false**.

(5) Implementare l'operatore **operator!=**.

(6) Dati due network A e B, la loro differenza $C = A-B$ è quel network che possiede tutti gli edge presenti nel network A ma non nel network B. Implementare l'operatore **operator-** che accetta in ingresso un **Network** e restituisce un **Network**.

(7) Implementare l'operatore l'operatore **operator-=**.

(8) Dati due network A e B, la loro intersezione $C = A - B$ è quel network che possiede solo gli edge presenti sia network A sia nel network B. Implementare l'operatore **operator∧** che accetta in ingresso un **Network** e restituisce un **Network**.

(9) Implementare l'operatore l'operatore **operator∧ =**.

(10) Implementare l'operatore l'operatore **operator!=**.

2 II Parte: classe Random_Network

(d) Classe Random_Network: La classe **Random_Network** è una classe derivata dalla classe **Network** e permette di creare network aleatori con un dato numero di nodi e di edge.

(1 - Obb) Implementare un costruttore che accetti in ingresso il numero di nodi ed il numero di edge e generi un network aleatorio.

(2 - Obb) È necessario implementare il costruttore di copia?

(3 - Obb) Implementare il distruttore di default.

TIP: Per generare un numero random è sufficiente includere la libreria <**cstdlib**> ed usare il codice di esempio

```
int nodes = 10;
int random = lrand48() % nodes;
```

per generare numeri compresi tra [**0, nodes-1**].

3 III Parte: Main

(e) Main:

(1 - Obb) Creare un file main che crei un **Network** 10×10 e aggiungere ed eliminare qualche nodo.

(2) Creare un **Random_Network** di dimensinone 10×10.

Mathematica

Si consideri il sistema di equazioni differenziali

$$\begin{aligned} \frac{d^2\theta_1(t)}{dt^2} &= -m_1^2\theta_1(t) - k\,\theta_1(t) + k\,\theta_2(t) \\ \frac{d^2\theta_2(t)}{dt^2} &= -m_2^2\theta_2(t) + k\,\theta_1(t) - k\,\theta_2(t) \end{aligned} \tag{1}$$

- Leggendo i due secondi membri, si scriva la matrice A dei coefficienti del vettore colonna $\Theta = (\theta_1, \theta_2)$ e si calcolino i corrispondenti autovalori $\lambda_{1,2}$.

- Si ponga, da qui in poi, $m_2 = m_1$.
  Si calcoli la soluzione delle equazioni $A \cdot \Theta = \lambda_{1,2}\Theta$, che permette di esprimere θ_2 in funzione di θ_1 (modi normali).
  Si può semplificare la soluzione con la sostituzione $\frac{1}{\sqrt{d^2}} \to \frac{1}{d}$.

- Si risolva il sistema di equazioni differenziali, imponendo come condizione al contorno $\dot\theta_1(0) = \dot\theta_2(0) = 0, \quad \theta_1(0) = \bar\theta_1, \quad \theta_2(0) = \bar\theta_2$.
  Si calcolino e semplifichino le soluzioni somma e differenza $\theta_1 + \theta_2$ e $\theta_1 - \theta_2$.
  Si impongano le relazioni che determinano i modi normali sui valori al contorno $\bar\theta_1$ e $\bar\theta_2$.
  Si rappresentino i due modi normali per $t \in [0, 10]$ con $\bar\theta_{1,2} = 0.5$, $m_1 = 1$, $k = 1$.
  Si commenti la dipendenza da k delle due soluzioni.

- Si risolva il sistema di equazioni differenziali, imponendo, come condizioni al contorno: $\dot\theta_1(0) = \dot\theta_2(0) = 0, \quad \theta_1(0) = \bar\theta, \quad \theta_2(0) = 0$.
  Si implementi una funzione dei parametri k e $\bar\theta$, che permetta di disegnare $\theta_1(t)$, con $t \in [0, 1000]$.
  Si disegni la soluzione, con $\bar\theta = 0.5$ e con $k = 0.01,\ 0.1,\ 1$; si presentino accostati i tre grafici ottenuti.
  Si ricordi che è in generale necessario utilizzare il comando `Evaluate[fun]` *all'interno di* `Plot` *per ottenere il grafico di* `fun` .

- Si mostri che l'espressione $a^\mu b^\nu c^\lambda d^\rho \ \epsilon_{\mu\nu\lambda\rho}$,
  con $1 \le \mu, \nu, \lambda, \rho \le 4$ e dove $\epsilon_{\mu\nu\lambda\rho}$ è il tensore totalmente antisimmetrico in 4 dimensioni, coincide con il determinante della matrice 4x4 costruita con i 4 vettori a, b, c, d.
  Si consideri l'utilizzo del comando `Signature`.