Algoritmi e Laboratorio
a.a. 2004/2005

Prova scritta del 14 aprile 2005

COGNOME ………………………. NOME ………..………………

Matr. n. …………………..

1. **(PUNTI 2 + 1)** Completare i seguenti due algoritmi con gli <u>invarianti di ciclo</u> e (dove richiesto) le <u>asserzioni finali</u> che permettono di dimostrarne la correttezza.

<u>ALGORITMO 1</u>
{A.I.: *n >= 1 e A[0..n-1] contiene numeri interi*}
current ← *A*[0]
i ← 1

{I.C. : …………………………………………………………………………………}
while *i* < *n* **do**
  if *current* < *A[i]* **then** *current* ← *A[i]*
  i ← *i* + 1
return *current*
{A.F.: *current* = ……………………………………………………………………}

<u>ALGORITMO 2</u>
{A.I.: *n >= 0*}
c ← 1
r ← 1

{I.C.: ……………………………………………………………………………………}
while *c* <= *n* **do**
  r ← *r* **c*
  c ← *c* + 1

return *r*
{A.F.: *r* = ………………………………………………………………………………}

2. **(PUNTI 2)**

Fornire la definizione di “delimitazione superiore alla complessita’ di un problema”.

3. **(PUNTI 2 + 2)**
a) Spiegare (sono sufficienti un paio di frasi in linguaggio naturale) l’idea alla base dell’algoritmo “integer sort”.
b) Simulare l’esecuzione dell’algoritmo sul seguente array X (contenente numeri interi di valore compreso nell’intervallo 1..5).

X

3	2	5	1

4. **(PUNTI 5)**
Scrivere un algoritmo “Divide et Impera” che risolva il seguente problema:
Dati in input un intero *n* che sia una potenza di 2 e due array A[1..n] e B[1..n] di interi restituisca:
TRUE se A[1]=B[n], A[2]=B[n-1],…,A[n-1]=B[2], A[n]=B[1];
FALSE, altrimenti

5. **(PUNTI 3)**
Scrivere l'algoritmo di Dijkstra seguendo lo schema di algoritmo greedy:

Dijkstra (G, w, s)
S ← Φ
{valuta le *appetibilità* iniziali dei vertici }

while ……………………… **do** {ci sono elementi da scegliere}
{scegli il vertice più *appetibile*}

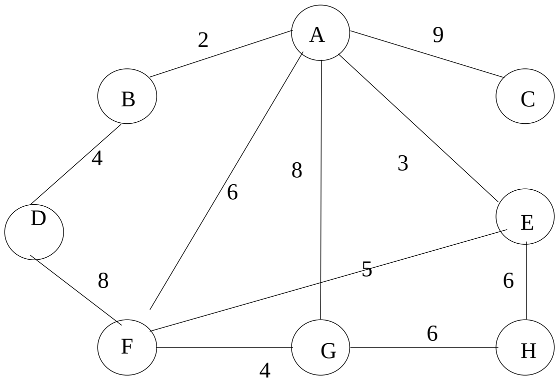
{aggiorna le *appetibilità* dei vertici}

6. **(PUNTI 1 + 2)**
a) Definire che cos'è una "componente connessa" in un grafo non orientato.

b) Scrivere un algoritmo iterativo che, dato un grafo non orientato, restituisca l’insieme degli insiemi dei vertici delle sue componenti connesse (si assuma di avere a disposizione il tipo di dato insieme, con le relative operazioni).

7. (PUNTI 5 + 4)

a) Applicando l'algoritmo di Prim a partire dal vertice H si determini un minimum spannino tree per il seguente grafo. Nel seguito ci riferiremo al minimum spanning tree restituito dall'algoritmo con il termine "*soluzione*".



Per svolgere correttamente l'esercizio occorre:

- compilare le seguenti tabelle (la riga 0 è già compilata), e
- disegnare (un disegno per ogni riga delle tabelle) l'albero mantenuto dall'algoritmo (contenente SIA gli archi che fanno parte della *soluzione* CHE gli archi candidati) al termine di ogni iterazione (cioe' DOPO che sono state aggiornate le appetibilita' dei vertici in coda).

La prima tabella indica, per ogni iterazione del ciclo esterno, per ogni vertice v che non appartiene alla *soluzione*, il peso k[v] di un arco di peso minimo che collega tale vertice alla soluzione **(∞ se un tale arco non e' ancora stato trovato)**. Il valore di k non deve essere più riportato quando il vertice e' ormai parte della *soluzione* **(si metta il simbolo -)**.

La seconda tabella indica, per ogni iterazione del ciclo esterno, l'insieme dei vertici v che sono entrati a far parte della *soluzione* (per i quali k[v] è il peso dell'arco incluso).

La riga 0 corrisponde al temine dell'inizializzazione (prima di entrare nel ciclo).
Quando nella coda con priorità ci sono vertici con lo stesso valore minimo di d si scelga quello che viene prima secondo l'ordine alfabetico.

k	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**
0	∞	∞	∞	∞	∞	∞	∞	**0**
1								
2								
3								
4								
5								
6								
7								
8								

	Insieme dei vertici t inclusi nella soluzione
0	**{ }**
1	
2	
3	
4	
5	
6	
7	
8	

b) Si determini, applicando l'algoritmo di Kruskal, un minimum spannino tree per il grafo riportato in precedenza. Disegnare, passo per passo, la costruzione dell'albero e indicare il peso dell'albero ottenuto.

8. (PUNTI 4) Sia G un grafo non orientato, non pesato, e connesso.
Scrivere un algoritmo ricorsivo che, dato un vertice x di G, restituisca l'altezza dell'albero di copertura ottenuto effettuando una visita in profondita' di G a partire dal vertice dato.