Progettazione di Algoritmi (9 CFU)
6/2/2020

Cognome e Nome:
Numero di Matricola:

Spazio riservato alla correzione

1	2	3	4	5	6	Totale
/18	/10	/23	20	/15	/14	/100

1.

a) Indicare quali delle seguenti affermazioni sono vere e quali sono false.
   1. $(\log n)n^{1/2} = O((\log n)^2)$ F
   2. $n^{1/4} = \Omega(2^{30})$ V
   3. $\log_4 n = \Theta(\log_2 n)$ V
   4. $(n/4)^n = \Theta(n^n)$ F
   5. $\frac{1}{4}\, n + n^{100} = O(n^{1/4n})$ V

b) Si dimostri che se $f(n) = \Omega(g(n))$ e $g(n) = \Omega(p(n))$ allora $f(n) = \Omega(p(n))$. **Occorre utilizzare solo la definizione di Ω e nessuna altra proprieta` (fornire le costanti c ed n_0).**

1) $f(n) = \Omega(g(n))$ e $g(n) = \Omega(p(n))$ allora $f(n) = \Omega(p(n))$

dalla definizione di Ω abbiamo che

2) $f(n) = \Omega(g(n)) \iff \exists\ c' > 0, n_0' \geq 0 \mid f(n) \geq c' g(n)\ \forall n \geq n_0'$

3) $g(n) = \Omega(p(n)) \iff \exists\ c'' > 0, n_0'' \geq 0 \mid g(n) \geq c'' p(n)\ \forall n \geq n_0''$

dalla prima abbiamo che

$$f(n) \geq c' g(n) \Rightarrow f(n) \geq c' c'' g\, p(n)$$

quindi $c = c'c''$ e $n_0 = \max\{n_0', n_0''\}$

a) Si analizzi il tempo di esecuzione nel caso pessimo del seguente segmento di codice fornendo una stima asintotica **quanto migliore e` possibile** per esso. **Si giustifichi in modo chiaro la risposta.**

```
FOR(i=1; i≤n; i=i+1){
    FOR(k=i; k<2i; k=k+1) {
        print(k);
    }
}
```

O(n)

O(n)

alla i-esima iterazione del for più esterno il for più interno itererà partendo da i fino a 2i quindi itera 2i-i volte quindi i volte

2 Si scriva lo pseudocodice di un algoritmo basato sul paradigma del Divide et Impera che trova la sottosequenza di somma minima di un array di numeri (l'array puo` contenere sia numeri positivi che numeri negativi). Per sottosequenza si intende un insieme di celle consecutive dell'array. L'algoritmo deve avere tempo di esecuzione uguale a O(nlog n).
N.B. : se non siete in grado di scrivere lo pseudocodice, descrivete in maniera chiara e schematica il comportamento dell'algoritmo. Alla versione dell'esercizio senza pseudocodice saranno detratti dei punti.

3.

a) Si definisca il concetto di ordine topologico di un grafo direzionato.

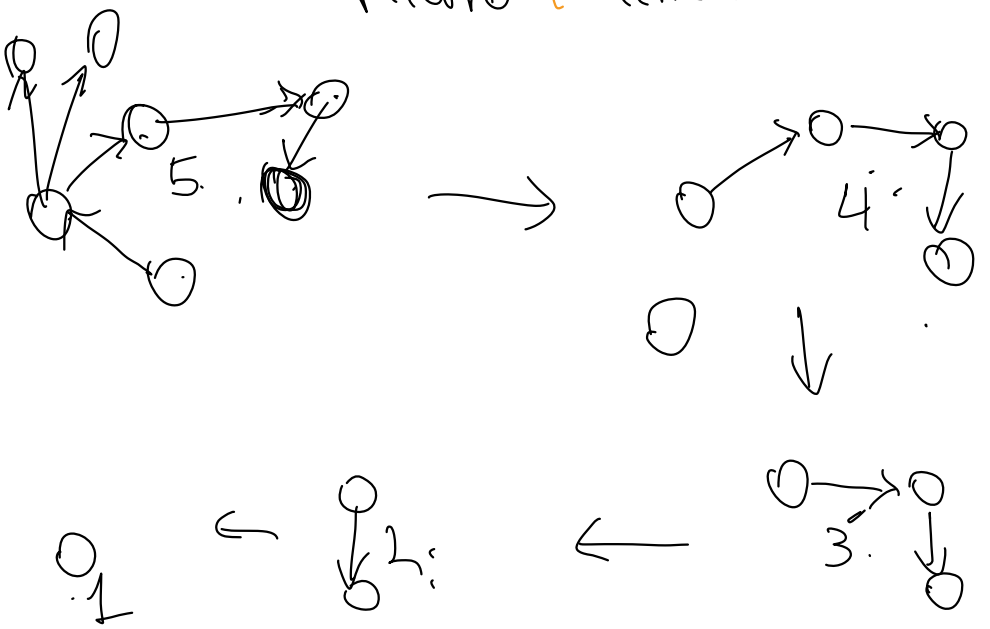
Un ordinamento topologico di un grafo direzionato è un'etichettatura dei vertici del grafo $v_1, v_2, \ldots, v_n$ tale che preso l'arco (v_i, v_j) $i < j$.

Per esempio, preso l'arco (u, v) di un grafo allora il vertice u precede v nell'ordinamento

b) Scrivere lo pseudocodice dell'algoritmo per ottenere l'ordinamento topologico di un DAG e si analizzi il tempo di esecuzione dell'algoritmo proposto. **Analizzare il tempo di esecuzione significa fornire un limite superiore asintotico quanto migliore e` possibile al tempo di esecuzione dell'algoritmo giustificando la risposta.**

```
TopologicalOrder(g)
    l = ∅
    se esiste un nodo v senza archi entranti:
        cancello v da {g} così da ottenere {g}-v
        l = TopologicalOrder(g-v)
        aggiungo v all'inizio di l
        ritorno l
    else
        ritorno l (che sarà vuota)
```

O(n)

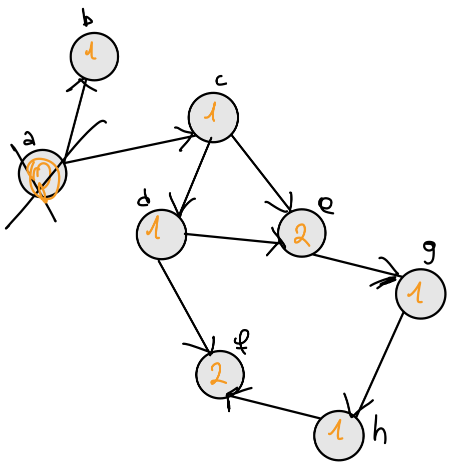


$$\sum_{i=1}^{n-1} i = \frac{n(n+1)}{2}$$

$$T(n) \leq T(n-1) + c'n \leq T(n-2) + c'(n-1) + c'n$$
$$\leq T(1) + c'2 + \ldots + c'(n-1) + c'n$$
$$\leq c'2 + \ldots + c'(n-1) + nc' = c + c'n(n+1)/2 - c' = O(n^2)$$

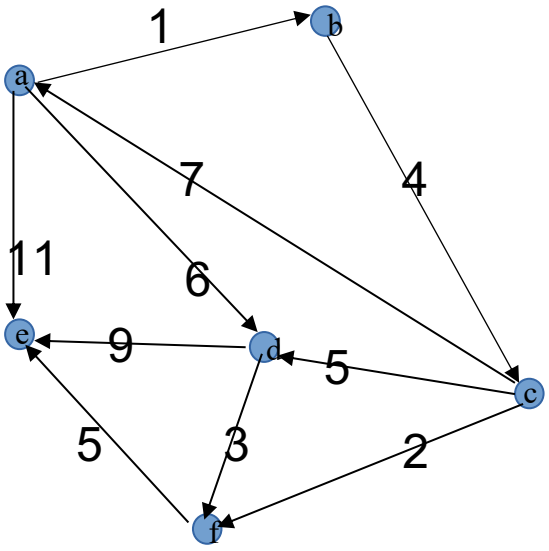
c) Modificare lo pseudocodice al punto precedente in modo che abbia tempo O(n+m), dove n ed m sono rispettivamente il numero di nodi e di archi del grafo e **dimostrare** che il tempo di esecuzione dell'algoritmo cosi` implementato e` appunto O(n+m).

```
TopologicaOrderNM(g) ←
    init count[]   //scandisco o nodi o arco        tempo O(m)
    init S         //vertici con count = 0          tempo O(n)
→ ordinetopologico(g)                                O(1)
{
    Trovo il nodo V senza archi entranti in S lo cancello
    cancello v da G                                  ciclo tutto l'array
    aggiorno count[] →                               deg{v} = m
    se count[] == 0 →
        S = S ∪ {}
    end if
    ordine topologico(G-v)   →   O(n)
    append(v)
}
end sub
```



count[a]=0
[b]=1 -0
[c]=1 -0
[d]=1
[e]=2
[f]=2
[g]=1 → S={a}
[h]=1
S={b,c}

d) Si mostri l'esecuzione dell'algoritmo di Dijkstra sul seguente grafo a partire dal nodo sorgente a. Si assuma che l'algoritmo scelga come radice il nodo **a. Occorre mostrare per ogni passo il contenuto della coda a priorita` e l'arco aggiunto all'albero dei percorsi minimi (arco formato dal nodo v aggiunto ad S in quel passo e dal nodo in S che precede v lungo il cammino minimo da s a v). Disegnare alla fine l'albero dei percorsi minimi generato.**



4. Algoritmi greedy

a) Si descriva in modo chiaro e schematico in che cosa consiste un'istanza del problema della minimizzazione dei ritardi (input) e qual e` l'obiettivo del problema (output). **Definire in modo preciso le quantita` che intervengono nella descrizione dell'output del problema.** Se dalla risposta a questo punto si evincera` che lo studente non sa in cosa consiste il problema della minimizzazione dei ritardi, i punti successivi dell'esercizio non saranno valutati.

b) Si fornisca un controesempio di 4 job che dimostra che la strategia shortest processing time first non sempre fornisce la soluzione ottima. **Giustificare in modo chiaro la risposta.**

c) Si scriva lo pseudocodice di un algoritmo greedy che trova la soluzione ottima per il problema della minimizzazione dei ritardi descrivendo il significato di tutte le variabili che compaiono nel codice. **Nel caso in cui non venga fornita questa descrizione, l'esercizio sara` valutato 0 punti.** Si analizzi inoltre il tempo di esecuzione dell'algoritmo nel caso pessimo. **Analizzare il tempo di esecuzione significa fornire un limite superiore asintotico quanto migliore e` possibile al tempo di esecuzione dell'algoritmo giustificando la risposta.**

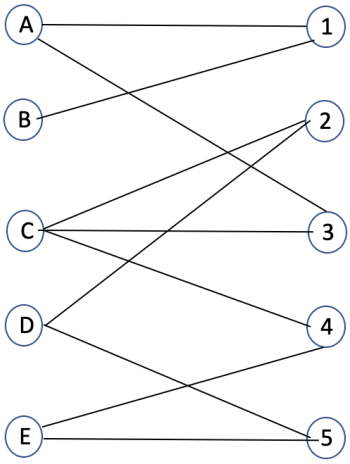
5. Programmazione dinamica

a) Si descriva in modo chiaro e schematico in che cosa consiste un'istanza del problema dell'interval scheduling pesato e qual e` l'obiettivo del problema. Se dalla risposta a questo punto si evincera` che lo studente non sa in cosa consiste il problema dell'interval scheduling pesato, i punti successivi dell'esercizio non saranno valutati.

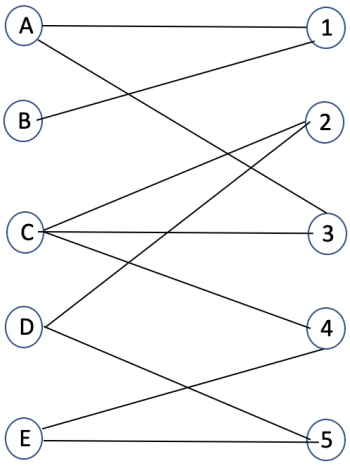
b) Fornire una formula per il calcolo del valore della soluzione ottima per il problema dell'interval scheduling pesato in termini di valori delle soluzioni ottime per sottoproblemi di taglia piu` piccola. **Spiegare in modo chiaro e schematico come si arriva alla formula da voi fornita definendo in modo chiaro tutte le quantita` e parametri che compaiono nella formula. In assenza di queste spiegazioni l'esercizio sara` valutato 0 punti.**

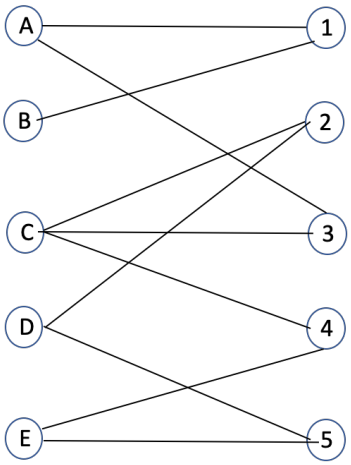
6.

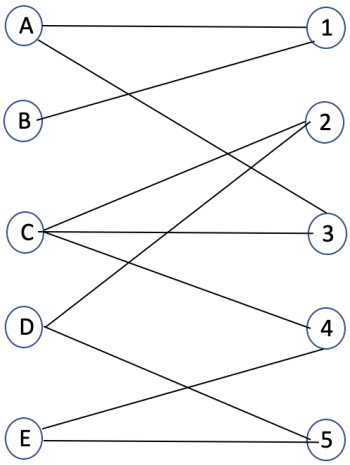
a) Mostrare i passi dell'algoritmo che computa il massimo matching di un grafo bipartito sul seguente grafo. **Il primo passo deve "utilizzare" l'arco (A,1). Se il primo passo non effettua la scelta indicata l'esercizio verra` valutato 0 punti.** Occorre mostrare l'esecuzione di tutto l'algoritmo e fornire il massimo matching individuato dall'algoritmo.

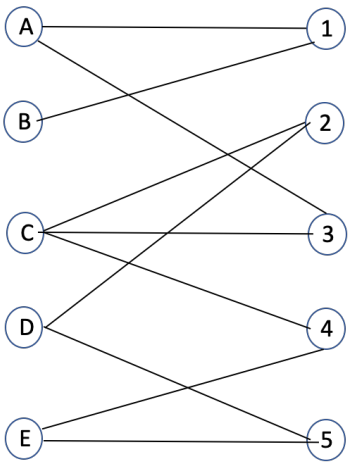


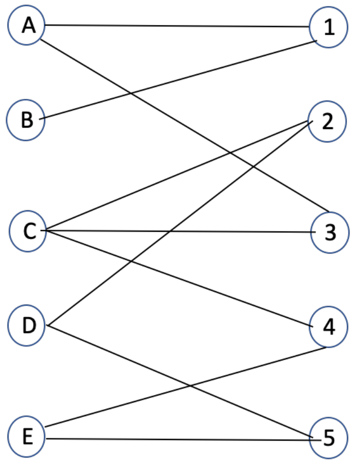
Per vostra comodita`, nelle pagine successive sono riportate diverse immagini del grafo. Il numero di coppie di immagini non e` indicativo del numero di passi effettuati dall'algoritmo.

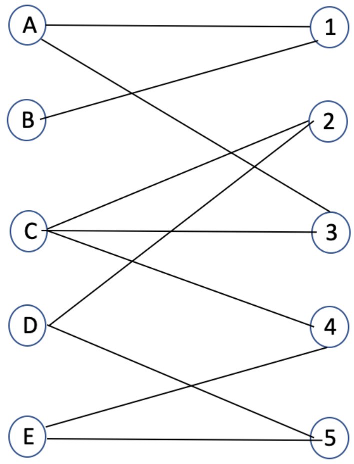


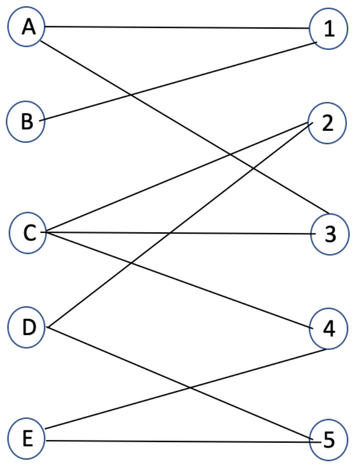


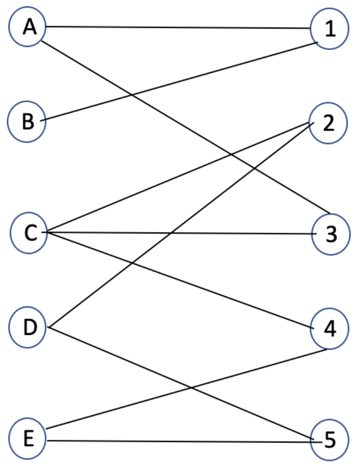


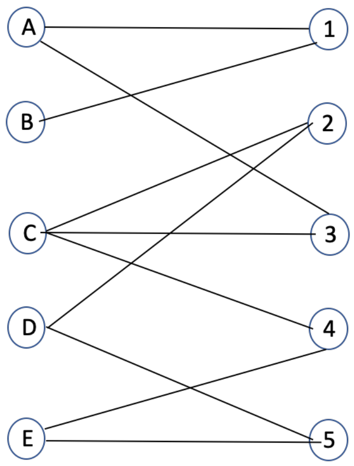


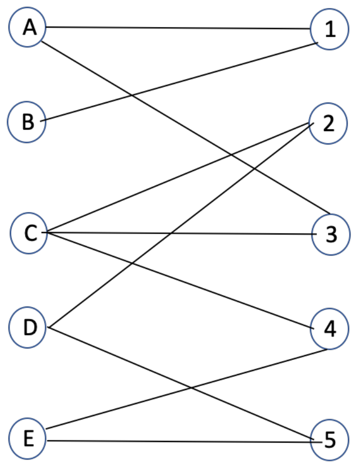

A
B
C
D
E
1
2
3
4
5

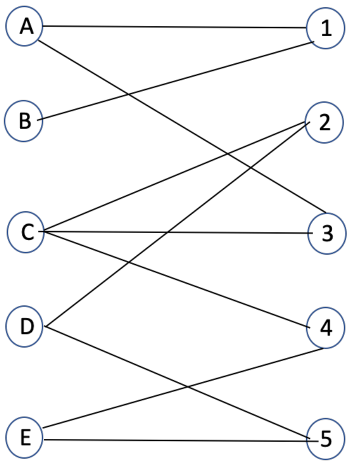

A
B
C
D
E
1
2
3
4
5

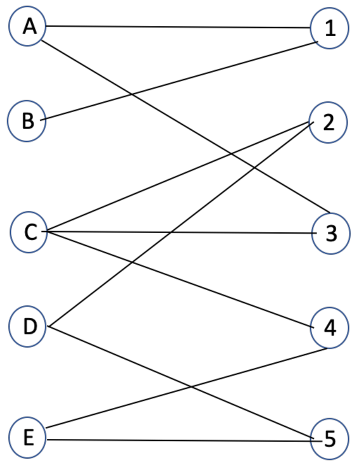

A
B
C
D
E
1
2
3
4
5

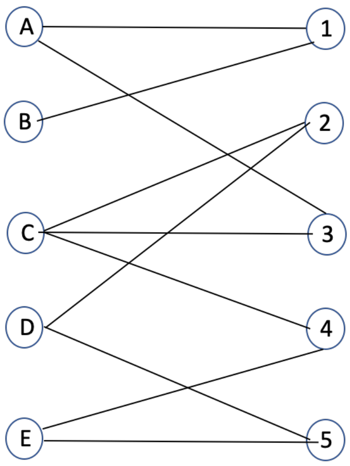

A
B
C
D
E
1
2
3
4
5

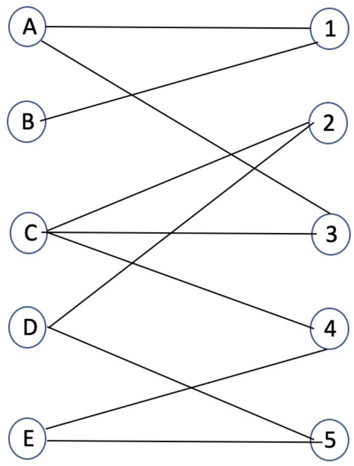

A
B
C
D
E
1
2
3
4
5

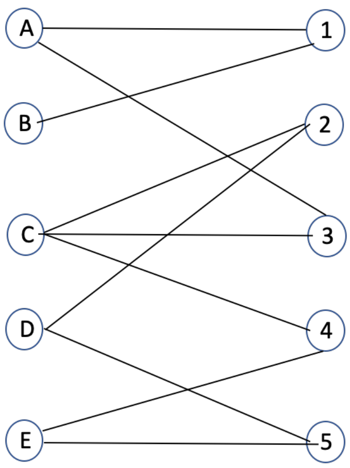

A
B
C
D
E
1
2
3
4
5

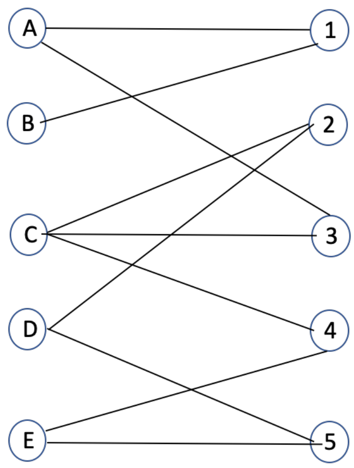

A
B
C
D
E
1
2
3
4
5


A
B
C
D
E
1
2
3
4
5


A
B
C
D
E
1
2
3
4
5


A
B
C
D
E
1
2
3
4
5


A
B
C
D
E
1
2
3
4
5


A
B
C
D
E
1
2
3
4
5

b) Si forniscano le definizioni di taglio s-t di una rete di flusso e di capacita` di un taglio s-t

c) Data una rete di flusso G ed una funzione di flusso f con valore di flusso massimo, spiegare come trovare in tempo O(n+m) un taglio di capacita` minima`.