Progettazione di Algoritmi (9 CFU)
13/6/2018

Cognome e Nome:
Numero di Matricola:

Spazio riservato alla correzione

1	2	3	4	5	6	Totale
/20	/20	/15	/15	/15	/15	/100

1. Analisi degli algoritmi e notazione asintotica

a) Indicare quali delle seguenti affermazioni sono vere e quali sono false.
   1. $n^3 \log n + 100n^2 = O(n^4)$ √
   2. $n^{1/3} = \Omega(n^{1/4})$ √
   3. $n^{1/3} = O(\log n)$ F
   4. $n^3 - 1000n^2 + 8 = \Omega(n^3)$ √
   5. $\log(\log n) = O((\log n)(\sqrt{n}))$ √

b) Si dimostri che se $1 < f(n) = O(h(n))$ e $1 < g(n) = O(p(n))$ allora $f(n)g(n) = O(h(n)p(n))$. Occorre utilizzare solo la definizione di O e nessuna altra proprieta`.

c) 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=3*i){
  FOR(j=0; j<m; j=j+1) {
     print(j);
  }
}
```

$\log_3 n$

$O(m)$

$O(n \log n)$

2. Divide et Impera

a) Si descriva in modo chiaro e schematico un algoritmo ricorsivo che prende in input un array di caratteri e computa il massimo numero di occorrenze consecutive del carattere ‘d’ . Ad esempio, se l’array contiene la sequenza <d b c c d d d c d d> allora l’algoritmo restituisce 3. Per ogni procedura, occorre specificare l’input e l’output della procedura.

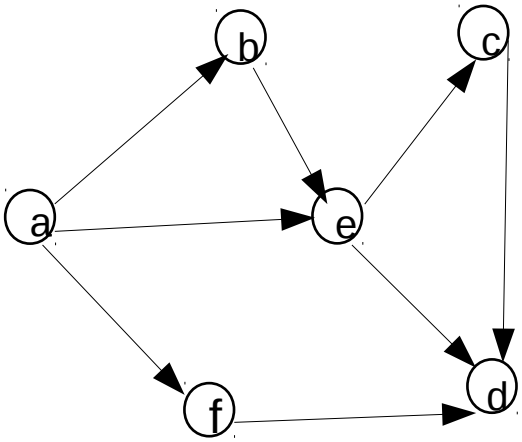
b) Si fornisca la relazione di ricorrenza che esprime un limite superiore al tempo di esecuzione dell'algoritmo al punto a) e si fornisca una stima quanto migliore e` possibile del tempo di esecuzione dell'algoritmo a partire dalla suddetta relazione di ricorrenza.

3. Grafi

a) Si scriva lo pseudocodice dell'algoritmo che effettua la visita DFS di un grafo in modo **ricorsivo** e costruisce l'**albero DFS**. 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**.

```
foreach Node n in g poni explored[n] = false        → O(n)
init T = DFS Tree vuoto
DFS(Graph g, Node s (sorgente), explored[])
    poni explored[s] = true
    foreach Node n adiacente ad s                   } deg(v)
        se explored[n] == false
            aggiungi l'arco (s, n) all'albero T
            DFS(g, n, explored)
        end if
    end foreach
```

b) Si mostri l'esecuzione dell'algoritmo per l'ordinamento topologico disegnando il grafo passato in input ad ogni chiamata ricorsiva dell'algoritmo e producendo un ordinamento topologico del grafo qualora questo esista.



4. Algoritmi greedy

a) Si scriva lo pseudocodice dell'algoritmo di Prim che fa uso della coda a priorita`. Si spieghi che cosa rappresenta la chiave associata nella coda a priorita` ad un generico vertice u .

b) Si analizzi il tempo di esecuzione dell'algoritmo nel caso in cui la coda a priorita` sia implementata mediante un heap binario. Analizzare il tempo di esecuzione significa fornire un limite superiore asintotico quanto migliore e` possibile al tempo di esecuzione dell'algoritmo **giustificando la risposta**.

c) Dire in che punto dell'algoritmo di Prim viene effettuata la scelta greedy e per quale motivo tale scelta garantisce che gli archi selezionati dall'algoritmo fanno parte del minimo albero ricoprente.

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.**

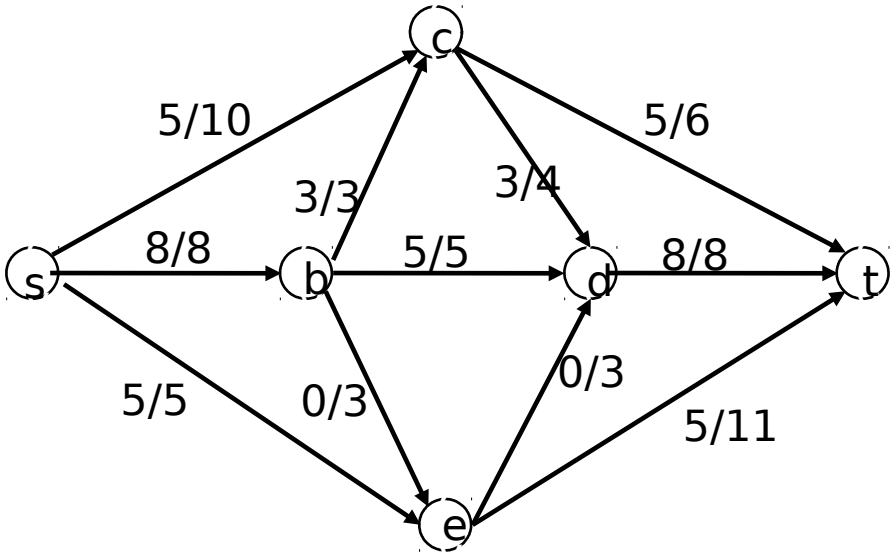
6. **Massimo flusso**

a) Si consideri la seguente rete di flusso e la funzione di flusso i cui valori sono indicati a sinistra delle capacita` degli archi. Si disegni la rete residua rispetto alla funzione flusso indicata e si dica se questa funzione ha valore massimo. Nel caso in cui la funzione non abbia valore massimo, si fornisca **la funzione flusso con valore massimo e il taglio di capacita` minima.** A tal fine si eseguano una o piu` iterazioni dell'algoritmo di Ford-Fulkerson **a partire dalla funzione di flusso data.**

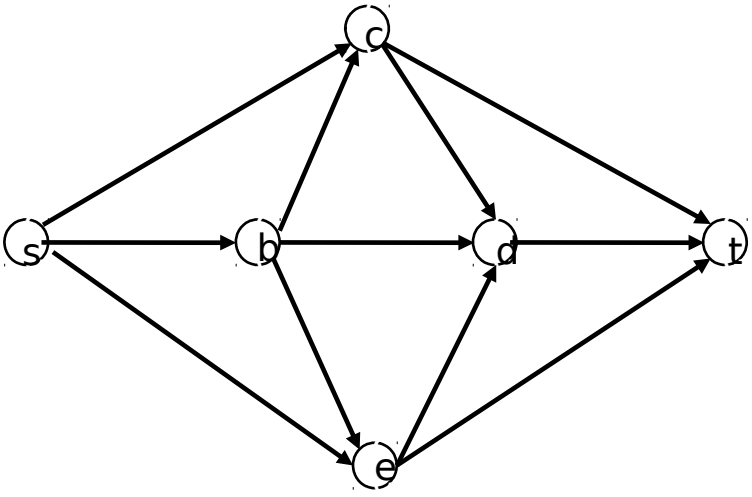
Per ogni iterazione dell'algoritmo, occorre

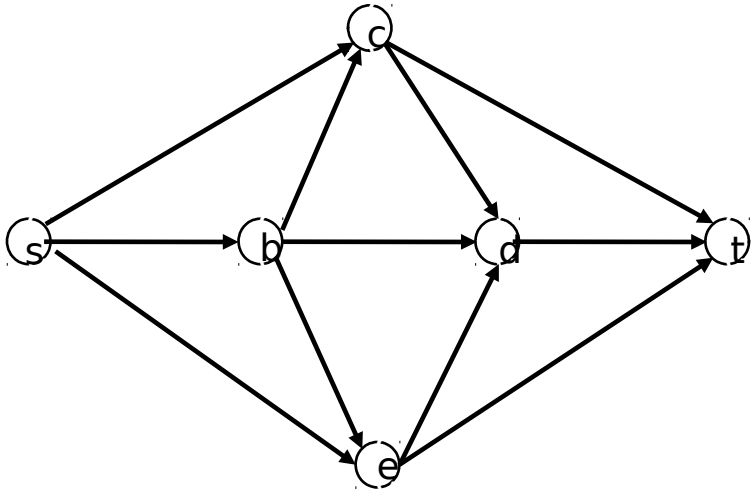
- **disegnare la rete residua all'inizio di quell'iterazione**
- **indicare il cammino aumentante da voi scelto**
- **mostrare il valore associato ad ogni arco del grafo al termine di quella iterazione**

N.B.: le risposte che non sono ottenute a partire dalla funzione di flusso data non saranno valutate.

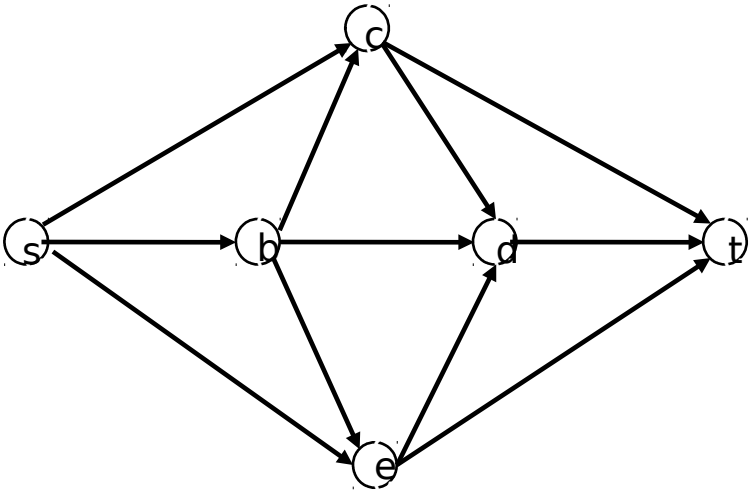


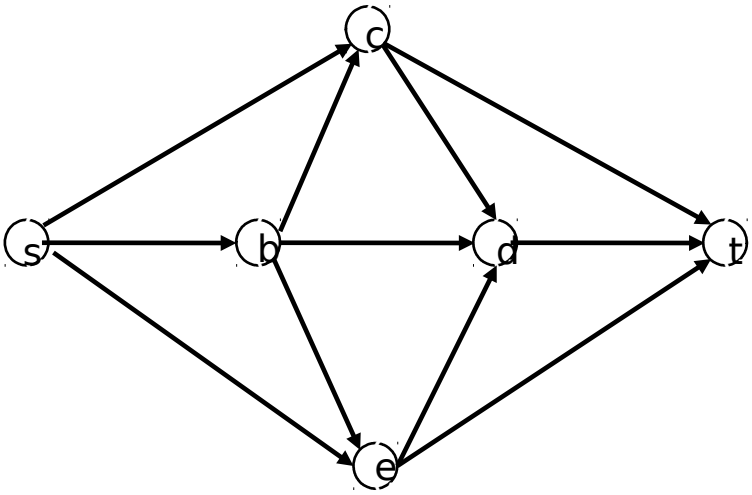
Per vostra comodita`, di seguito sono riportate diverse copie della rete di flusso, suddivise a coppie. **A partire dalla funzione di flusso data, usate l'immagine di sinistra di ciascuna coppia per disegnare la rete residua e l'immagine di destra per riportare i valori della funzione flusso assegnati a ciascun arco.** Ovviamente potrebbe essere necessario aggiungere e/o cancellare (con una x) degli archi nelle immagini di sinistra. Il numero di coppie non e` indicativo del numero di iterazioni effettuate dall'algoritmo di Ford-Fulkerson. Procedete dall'alto verso il basso utilizzando solo le coppie di grafi che vi servono per illustrare l'intera esecuzione dell'algoritmo. **N.B.: Se non saranno rispettate queste indicazioni per lo svolgimento dell'esercizio, l'esercizio non sara` valutato.**

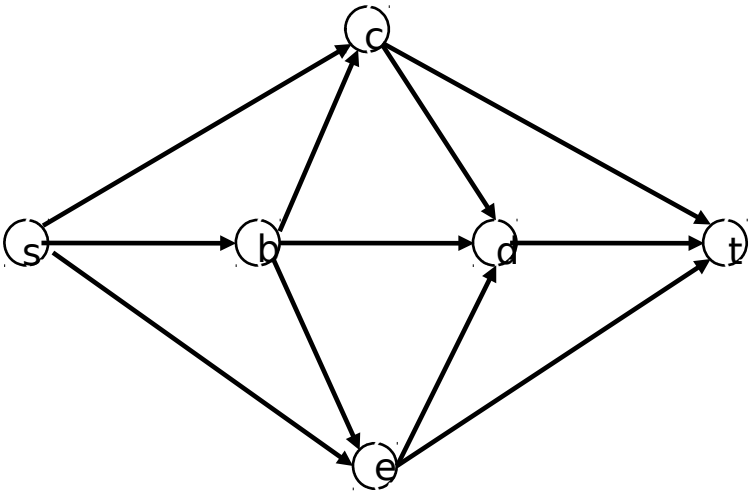


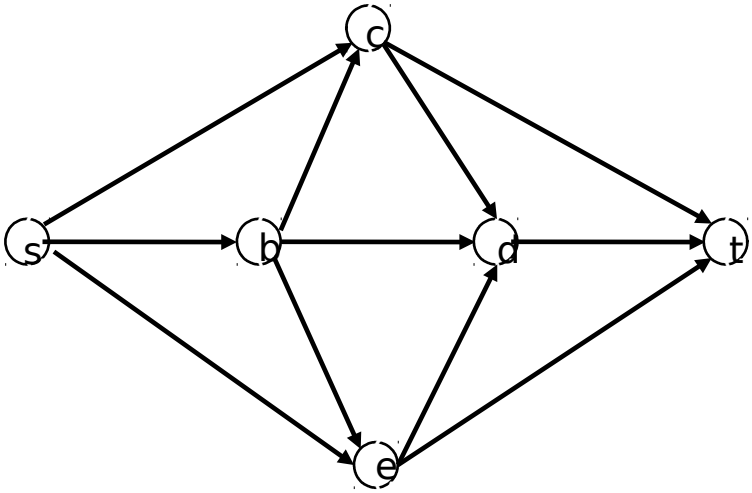


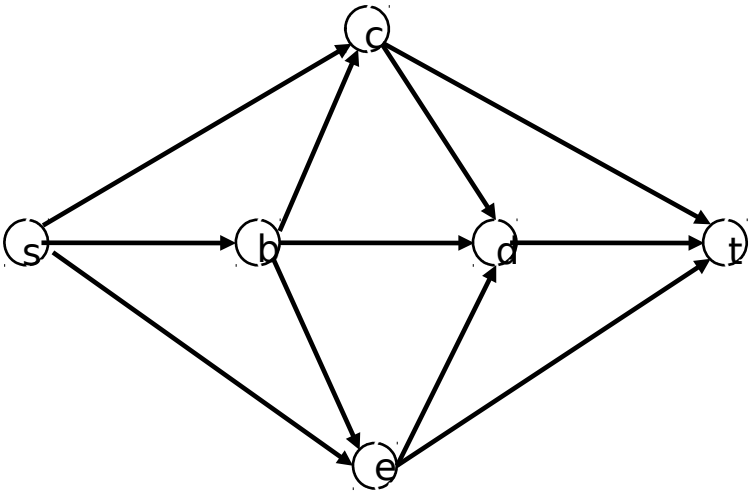
Progettazione di Algoritmi (9 CFU)
13/6/2018

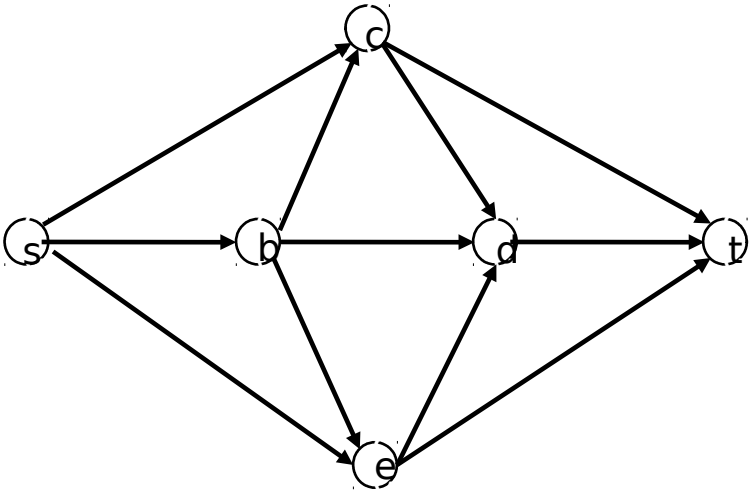












b) Descrivere in modo chiaro il comportamento dell'algoritmo che computa il massimo numero di percorsi disgiunti tra due nodi di un grafo direzionato. Occorre anche descrivere in che modo l'agoritmo alla fine genera i percorsi disigunti.