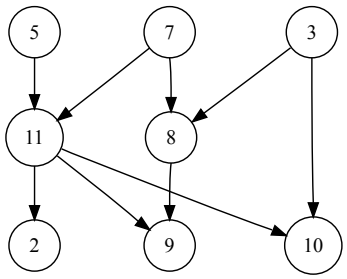
Analisi e progettazione di algoritmi

(III anno Laurea Triennale - a.a. 2022/23)

Prova scritta 1 febbraio 2024

Esercizio 1 Si consideri il seguente grafo:



1. Si ottenga un ordine topologico con il primo algoritmo visto a lezione, specificando, ogni iterazione, l'insieme di nodi sorgente (`S`), il nodo estratto (`u`), e l'ordine corrente (`Ord`). **Ogni volta che si hanno più scelte possibili, si segua l'ordine numerico.**

2. Si ottenga un ordine topologico con il secondo algoritmo visto a lezione (DFS ccon timestamp), spiegando cosa succede a ogni iterazione. **Ogni volta che si hanno più scelte possibili, si segua l'ordine numerico** (quindi, la prima visita inizierà dal nodo 2).

3. Si dia un altro ordine topologico diverso da quelli ottenuti precedentemente, e un ordine totale dei nodi che non sia topologico, spiegando perchè.

Soluzione

1. Il primo algoritmo estrae di volta in volta dal grafo un nodo sorgente. Si ha quindi:

insieme nodi sorgente	nodo estratto	ordine
3,5,7	3	3
5,7	5	3,5
7	7	3,5,7
8,11	8	3,5,7,8
11	11	3,5,7,8,11
2,9,10	2	3,5,7,8,11,2
9,10	9	3,5,7,8,11,2,9
10	10	3,5,7,8,11,2,9,10

2. Il secondo algoritmo consiste nell'effettuare una visita in profondità con timestamp e poi prendere i nodi in ordine inverso di fine visita. Si ha quindi:

nodo	inizio visita	fine visita
2	1	2
3	3	10
8	4	7
9	5	6
10	8	9
5	11	14
11	12	13
7	15	16

L'ordine topologico risultante è quindi: 7 5 11 3 10 8 9 2

3. Un altro ordine topologico è per esempio: 7 5 3 11 10 8 9 2
   Un ordine totale dei nodi che non sia topologico è, per esempio, qualunque ordine che inizi con 11, in quanto, dato che esiste l'arco $(5, 1)$, 11 deve essere preceduto da 5.

Esercizio 2 Si consideri il seguente algoritmo ricorsivo.

```
fun(A,i,j)// A array [1..n], 1<=i,j<=n
if ( i > j ) return 0
else if ( i = j ) return A[i]
else
  m  = ( i + j ) / 2
  return fun(A, i, m) + fun(A, m+1, j)
```

L'algoritmo viene inizialmente invocato con `fun(A, 1, n)`.

1. Scrivere e risolvere la relazione di ricorrenza che descrive il costo di `fun` in funzione di `n`.
2. Cosa calcola `fun(A, 1, n)`? Giustificare la risposta, e dire se l'algoritmo è ottimo.

Soluzione

1. La relazione di ricorrenza è la seguente (conviene esprimere n come 2^k):

   $T(2^0) = 1$
   $T(2^k) = 1 + T(2^{k-1})$, per $k > 0$.

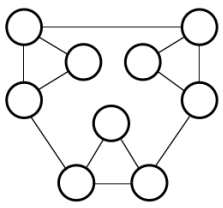
   Utilizzando la tecnica per sostituzioni successive si ha:

   $$T(2^k) = 1 + 2 \cdot T(2^{k-1}) = 1 + 2 + 2 \cdot T(2^{k-2}) = \ldots = 2^0 + 2^1 + \ldots + 2^k = \frac{2^{k+1}-1}{2-1}$$

   quindi $T(n) = O(n)$.

2. La chiamata `fun(A, 1, n)` calcola la somma degli elementi dell'array, come è immediato vedere per induzione aritmetica forte. Infatti, nel caso di zero o un elemento l'algoritmo è corretto. Nel caso di almeno due elementi, l'algoritmo viene richiamato sulle due metà dell'array (di dimensione strettamente minore), restituisce per ipotesi induttiva la somma degli elementi delle due metà e poi le somma. L'algoritmo è ottimo in quanto per calcolare la somma degli elementi di un array ovviamente occorre almeno esaminare tutti gli elementi.

Esercizio 3 Supponi di applicare l'algoritmo *MCMinCut* al grafo in figura.



Calcola quante volte devi ripetere *MCMinCut* per ottenere il taglio minimo con probabilità almeno del 99%.