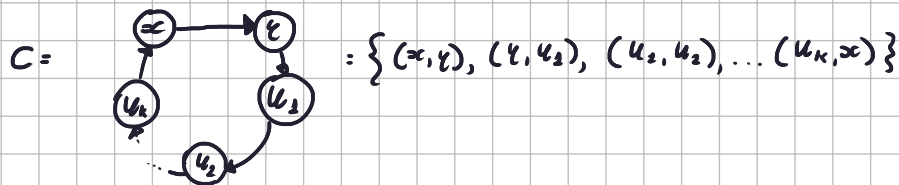


Esercizio 1. Dare lo pseudocodice di un algoritmo $O(n+m)$ che accetta in input un grafo diretto G e un arco (x,y) e restituisce "YES" in output se esiste un vertice z e un albero di visita T di un DFS partendo da z per il quale l'arco (x,y) è un arco all'indietro. Noto bene che z potrebbe essere x oppure y . Dare un argomento giustificando la correttezza dell'algoritmo.

Se (x,y) fa parte di un ciclo, allora può essere un arco all'indietro. Sia C questo ciclo



Allora una DFS che inizia da u_1 ha (x,y) come arco all'indietro.

Es1 (G : grafo, (x,y) : arco) {

$Vis = DFS(G, y)$ // con un DFS controllo se esiste un percorso da y ad x
 if ($Vis[x] == 1$) { return 'Yes' }
 return 'No'

}

Esercizio 2. Sia $A[1, \dots, n]$ un array di n elementi. Un *super majority element* di A è un elemento che occorre in almeno $2n/3$ posizioni di A (quindi se $n = 6$, un super majority element deve occorrere al minimo $4 = 2/3 \times 6$ volte in A ; se $n = 8$, deve occorrere al minimo $6 \geq 2/3 \times 8 = 5.333$ volte). Assumiamo che gli elementi di A non possono essere ordinati ma si può calcolare se due elementi sono uguali (per esempio, gli elementi di A potrebbero essere stringhe). Dare il pseudocodice per un algoritmo $O(n \log n)$ che prenda in input un array A e restituisce "YES" se esiste un super majority element in A .

SME (A : array) {

$n = A.length()$

 if ($n == 1$) { return $A[0]$ }

 if ($n == 2 \wedge A[0] == A[1]$) { return $A[0]$ }

$a = SME(A[0:n/3])$

$b = SME(A[n/3:2n/3])$

$c = SME(A[2n/3:n])$

$ca = 0$

$cb = 0$

$cc = 0$

 for ($i = 0, 1, \dots, n$) {

$ca += (A[i] == a)$

$cb += (A[i] == b)$

$cc += (A[i] == c)$

 }

 if ($ca \geq 2 \cdot n/3$) { return a }

 if ($cb \geq 2 \cdot n/3$) { return b }

 if ($cc \geq 2 \cdot n/3$) { return c }

 return NULL

}

Costo : $T(n) = 3T(\frac{n}{3}) + O(n) \Rightarrow O(n \cdot \log(n))$

Esercizio 3. Dare lo pseudocodice di un algoritmo che prende in input un grafo diretto aciclico con vertici colorato rosso, blu, o verde e due vertici x e y e restituisce "YES" in output se esiste un cammino da x a y con al minimo un vertice di tutti i tre colori (quindi il cammino dovrebbe avere al minimo un vertice rosso, al minimo uno blu, e al minimo uno verde). Se un tal cammino non esiste, l'algoritmo restituisce "NO" in output.

Prop: Sia λ una combinazione di colori, siano u, v, w . Se esiste un cammino λ da u a v e $(v, w) \in E(G)$, allora esiste un cammino λ da u a w .

● = 100 ● = 010 ● = 001 $C: V(G) \rightarrow \{100, 010, 001\}$

Colori: $(G: \text{grafo}, C: \text{funzione colore}, x: \text{nodo}, y: \text{nodo}) \{$

```

n = |V(G)|
T[n]: array lungo n
T[x] = C[x]
Q: coda
Q.push(x)
do {
    u = Q.pop()
    for each w in u.adj {
        T[w] = T[u] | C[w] // bit wise OR
        Q.push(w)
    }
}
if (T[y] == 8) { return "YES" }
return "NO"
}

```

Esercizio 4. Dare il pseudocodice di un algoritmo che accetta in input un grafo G non diretto con pesi sugli archi e un arco (x, y) e restituisce "YES" se esiste un albero di copertura di peso minimo che non contiene l'arco (x, y) .

Es 4 $(G: \text{grafo}, (x, y): \text{arco}) \{$

```

a = w(Kruskal(G))
b = w(Kruskal(G \ {(x, y)}))
if (a == b) { return "YES" }
return "NO"
}

```

tutti gli MST hanno identico peso!