Esercizi

Tutte le strade portano a Roma

Un vertice v in un grafo orientato G si dice di tipo "Roma" se ogni altro vertice w in G può raggiungere v con un cammino orientato che parte da w e arriva a v.

1. Descrivere un algoritmo che dati un grafo G e un vertice v, determina se v è un vertice di tipo "Roma" in G.
2. Descrivere un algoritmo che, dato un grafo G, determina se G contiene un vertice di tipo "Roma".

In entrambi i casi è possibile trovare un algoritmo con complessità $O(m + n)$, ma anche altre complessità verranno considerate.

Esercizi

Double gap

In un vettore V di n interi non necessariamente ordinato, si dice double-gap un indice i, $1 \leq i < n$, tale che $V[i+1] - V[i] \geq 2$.

1. Dato un vettore V di $n \geq 2$ interi tale che $V[n] - V[1] \geq n$, provare che V ha almeno un double-gap. Suggerimento: per induzione.
2. Progettare un algoritmo che, dato un vettore V di $n \geq 2$ interi tale che $V[n] - V[1] \geq n$, restituisca la posizione (primo indice) del double-gap in $O(\log n)$ tempo.

Samarcanda

Nel gioco di Samarcanda, ogni giocatore è figlio di una nobile famiglia della Serenissima, il cui compito è di partire da Venezia con una certa dotazione di denari, arrivare nelle ricche città orientali, acquistare le merci preziose al prezzo più conveniente e tornare alla propria città per rivenderle.
Dato un vettore P di n interi in cui $P[i]$ è il prezzo di una certa merce al giorno i, trovare la coppia di giornate (x, y) con $x < y$ per cui risulta massimo il valore $P[y] - P[x]$. Calcolare la complessità e dimostrare la correttezza. È possibile risolvere il problema in $O(n)$.

Esercizi

Stesso livello

Scrivere un algoritmo che preso in input un albero binario T i cui nodi sono associati ad un valore intero $T.key$, restituisca il numero di nodi dell'albero il cui valore è pari al livello del nodo.
Vi ricordo che il livello del nodo è pari al numero di archi che devono essere attraversati per raggiungere il nodo dalla radice. Per cui la radice ha livello 0, i suoi figli hanno livello 1, etc.

Esercizi

Distanza tra insiemi di nodi

Dato un grafo G e due sottoinsiemi V_1 e V_2 dei suoi vertici si definisce distanza tra V_1 e V_2 la distanza minima in numero di archi da un nodo in V_1 ad un nodo in V_2. Nel caso V_1 e V_2 non siano disgiunti allora il valore è 0.

Descrivere un algoritmo mindist(Graph G, Set V_1, Set V_2) che restituisce la distanza minima. Discutere complessità e correttezza, assumendo che l'implementazione degli insiemi sia tale che il costo di verificare l'appartenenza di un elemento all'insieme abbia costo $O(1)$.

Nota: è facile descrivere un algoritmo $O(nm)$; esistono tuttavia algoritmi di complessità $O(n^2)$ (con matrice di adiacenza) e $O(m+n)$. La valutazione dipenderà dall'efficienza dell'algoritmo trovato.

Esercizi

Il gioco delle coppie

Scrivere un algoritmo che, dato un vettore A di n interi distinti (n pari), ritorna **true** se è possibile partizionare A in coppie di elementi che hanno tutte lo stesso somma (intesa come la somma degli elementi della coppia), **false** altrimenti. Ad esempio:

$7, 4, 5, 2, 3, 6$

può essere partizionato in $7 + 2 = 4 + 5 = 3 + 6$.

Soluzioni

Tutte le strade portano a Roma

```
isRoma(GRAPH G, NODE v)
integer status ← new integer[1 ... G.N]
foreach v ∈ G.V() − {r} do
    status[v] ← UNVISITED
status[r] ← REACH-ROMA
foreach v ∈ G.V() − {r} do
    if status[v] = UNVISITED then
        visitRoma(G, v, status)
    if status[v] = VISITED then
        return false
return true
```

```
visitRoma(GRAPH G, NODE v, integer[] status)
status[v] ← VISITED
foreach w ∈ G.adj(v) do
    if status[w] = UNVISITED then
        visitRoma(G, w, status)
    if status[w] = REACH-ROMA then
        status[v] ← REACH-ROMA
```

Soluzioni

Double-gap

```
doublegap(integer[] V, integer i, integer j)
if j = i + 1 then
    return i
integer m ← ⌊(i + j)/2⌋
if V[m] − V[i] ≥ m − i + 1 then
    return doublegap(V, i, m)
else
    return doublegap(V, m, j)
```

Soluzioni

Stesso livello

```
sameLevel(TREE T, integer ℓ)
if T = nil then
  └ return 0
integer count ← sameLevel(T.right(), ℓ + 1) + sameLevel(T.left(), ℓ + 1)
if T.key = ℓ then
  └ count ← count + 1
return count
```

Soluzioni - Stesso livello

```
mindist(GRAPH G, SET V1, SET V2)
QUEUE Q ← Queue()
integer[] dist ← new integer[1 ... V.n]
foreach u ∈ G.V() do
    if V1.contains(u) then
        Q.enqueue(u)
        dist[u] ← 0
        if V2.contains(u) then return 0
    else
        dist[u] ← ∞
while not Q.isEmpty() do
    NODE u ← Q.dequeue()
    foreach v ∈ G.adj(u) do                      % Esamina l'arco (u, v)
        if dist[v] = ∞ then            % Il nodo u non è già stato scoperto
            dist[v] ← dist[u] + 1
            if V2.contains(v) then return dist[v]
            Q.enqueue(v)
return +∞
```

Soluzioni

Il gioco delle coppie

```
checkPairs(integer[] A, integer n)
sort(A, n)
integer s ← A[1] + A[n]
for i ← 2 to n/2 − 1 do
    if A[i] + A[n − i + 1] ≠ s then
        return false
return true
```