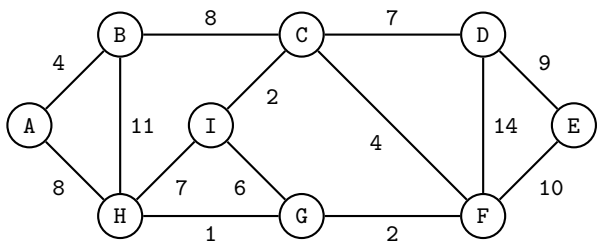
**Problema 1** – Insieme disgiunto di intervalli pesati

```
SET maxinterval(int a, int b, int w, int n)
    { ordina gli intervalli per estremi di fine crescenti }
    int[][] pred = computePredecessor(a, b, n)
    int[][] DP = new int[0...n]
    DP[0] = 0
    da i = 1 fino a n
        DP[i] = max(DP[i-1], w[i] + DP[pred[i]])
    // Costruisco l'insieme dei predecessori
    i = n
    SET S = Set()
    finché i > 0 fai % fintanto che ci sono intervalli disponibili
        se DP[i-1] > w(i) + DP[pred[i]] allora % commento
            i = i - 1 % non considerarlo
        allora
            S.insert(i) % inseriscilo nell'insieme
            i = pred[i] % scorri gli interalli
    ritorna S % ritorna l'insieme di intervalli ordinati
int[][] computePredecessor(int[][] a, int[][] a, int n)
    int[][] pred = new int[0...n]
    pred[0] = 0
    da i = 1 fino a n
        j = i - 1
        finché j > 0 e b[j] > a[i] fai
            j = j - 1
        pred[i] = j
    ritorna pred
SET independentSet(int[] a, int[] b)                                   (3)
    { ordina a e b in modo che b[1] ⩽ b[2] ⩽ ··· ≤ b[n] }
    // O(n) se già ordinati, O(n log n) altrimenti
    SET S = Set()
    S.insert (1) % Inserisco il primo intervallo
    int ultimo = 1 % Ultimo intervallo inserito
    da i = 2 fino a n
        se a[i] ⩾ b[ultimo] allora
            // Gli intervalli sono indipendenti
            S.insert (i) % Lo inserisco
            ultimo = i % Lo rendo l'ultimo inserito
    ritorna S
```




(4)

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum. (5)

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris. (6)

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa. (7) (8) (9)

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula. (10)

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

**Problema 2** – Cammini Minimi

```
SET kruskal(EDGE[] A, int n, int m)
    // EDGE[]: vettore di archi
(1) SET T = Set() % insieme (inizialmente vuoto) che conterrà gli archi dell'albero minimo
    MFSET M = Mfset(n) % insieme disgiunto grande
    // ordino per peso crescente
(2) { ordina A[1...m] in modo che A[1].peso ⩽ ··· ⩽ A[m].peso }
    int c = 0 % quanti archi ho aggiunto
    int i = 1 % quale arco sto guardando
(3) finché c < n - 1 e i ⩽ m fai % Termina quando l'albero è costruito
        // c < n - 1: ho raggiunto tutti gli archi necessari per fare un albero
        // i ⩽ m: ho esaurito tutti gli archi da guardare (per controllo)
        se M.find(A[i].u) ≠ M.find(A[i].v) allora % non fanno parte dello stesso albero
            M.merge(A[i].u, A[i].v) % unisco gli insiemi disgiunti
            T.insert(A[i]) % inserisco l'arco all'albero
            c = c + 1 % ho aggiunto un altro arco
        i = i + 1 % guardo il prossimo arco
    ritorna T % Ritorna l'albero di copertura minimo
prim(GRAPH G, NODE r, int[] p)
    // r: nodo dalla quale parto
    // p: vettore dei padri
    PRIORITYQUEUE Q = MinPriorityQueue ()
    PRIORITYITEM[] pos = new PRIORITYITEM[1...G.n]
    // inserisco i nodi nella coda, memorizzando la loro posizione
    per ciascun u ∈ G.V() - {r} fai
        pos[u] = Q.inserisci(u, +∞)
    // Inserisco il "nodo di partenza"
    pos[r] = Q.inserisci(r, 0)
    p[r] = 0 % convenzione per indicare che non ha padre
    finché non Q.isEmpty() fai % non ci sono più nodi
        NODE u = Q.deleteMin() % cancello e restituisco il nodo
        pos[u] = nil % non considero più quel nodo
        // per ciascun nodo adiacente a quello considerato
        per ciascun v ∈ G.adj(u) fai
            se pos[v] ≠ nil e w(u,v) < pos[v].priority allora
                // pos[v] ≠ nil: è già stato visitato
                // w(u,v) < pos[v].priority:
                Q.decrease(Pos[v], w(u,v)) % commento
                p[v] = u % commento
int[], int[] CamminiMinimi(GRAPH G, NODE s)
    PRIORITYQUEUE S = PriorityQueue() % O(n) · 1
    S.inserisci(s, 0)
    finché non S.isEmpty() fai % O(n)
        // O(n) vettore ordinato / O(log n) heap binario
        int u = S.deleteMin()
        b[u] = falso
        per ciascun v ∈ G.adj(u) fai
            se d[u] + G.w(u,v) < d[v] allora
                se non b[v] allora
                    // O(1) · n vettore ordinato / O(log n) · n heap binario
                    S.inserisci(v, d[u] + G.w(u,v))
                    b[v] = vero
                altrimenti
                    // O(1) · m vettore ordinato / O(log n) · m heap binario
                    S.decrease(v, d[u] + G.w(u,v))
                T[v] = u
                d[v] = d[u] + G.w(u,v)
    ritorna (T, d)
```