①

```
TSP-tour(G, c) // G is graph, c is cost function
    T = MST-Prim(G, c, r) // r is root vertex, arbitrary
    H = preorder(T, r) // H is list of vertices
    return hamiltonian(H) // compute cycle
                          // returning to root

MST-Prim(G, c, r)
    for each u ∈ G.V // for each vertex
        u.key = ∞
        u.π = NIL
    A = ∅; r.key = 0
    Q = G.V // Priority Q initialized to all vertices
    while Q ≠ ∅
        u = Extract-Min(Q)
        for each v ∈ G.Adj[u] // for each vertex
                              // adjacent to u.
            if v ∈ Q and c(u,v) < v.key
                v.π = u
                v.key = w(u,v)
        A = A ∪ {u,v} // A is MST
```

If the cost function satisfies the triangle inequality, $c(u,w) \leq c(u,v) + c(v,w)$, the tour T returned by TSP-tour is less than W with $W = 2H^*$ where H^* is cost of optimal tour.
$c(T) \leq c(H^*)$ since T can be obtained by deleting an edge from H^*. A full walk, W, traverses every edge of T twice so $c(W) = 2c(T) \leq 2c(H^*)$. By the triangle inequality we can delete any vertex from W and the cost does not increase, which can generate a tour from a walk visiting a vertex more than once. Call this H, which must be $\leq W$ thus $c(H) \leq c(W) \leq 2c(H^*)$

(2) makeChange(d, v) // d is array containing possible
// denominations. v is input bill

```
table = [d.length][v]
for i = 1 to v
    table[0][i] = ∞
for i = 1 to d.length
    for j = 1 to v
        if (j - d[i] >= 0)
            amount = table[i][j - d[i-1]]
        else
            amount = ∞
        table[i][j] = min(table[i-1][j], amount)
for i = 1 to d.length
    for j = 1 to v
        if table[i][j] ≠ ∞
            return true
```

(3) a) $W(i,j) = \max(W(i+1,j), W(i,j+1)) + W(i,j)$

b)

```
w(i, j)
    if i==n and j==n {
        return max(weight[i-1][j], weight[i][j-1])
                + weight[i][j]
    }
    right = table[i+1][j]
    down = table[i][j+1]
    if right and down ≥ 0
        if right > down
            w(i+1, j)      return
        else
            w(i, j+1)      return
    if right ≥ 0
        table[i+1][j] = weight[i][j] + weight[i+1][j]
        w(i+1, j)
    else
        table[i][j+1] = weight[i][j] + weight[i][j+1]
        w(i, j+1)
```