1 Question 1

The algorithm is similar to maximum flow problem in directed graph solved by Ford-Fulkerson algorithm, the extension can be made by transformed each undirected edge into a pair of anti-parallel directed edge with the same capacity one.

for each \{u, v\} in E:
    //assign initial flow to zero
    f[u, v] = 0
    f[v, u] = 0

while there is a acyclic path s->v in residual network G:
    //note that a undirected edge \{u, v\} corresponds to
    //two directed edge \{u,v\} and \{v, u\} in residual network
    //with capacity one in residual network
    delta = min\{cf(u,v) \mid \{u, v\} in path s->v\}
    for each \{u, v\} in s->v:
        f[u,v] = f[u, v] + delta //increase current path
        f[v,u] = -f[u,v]

If using BFS to find augmenting path, takes O(E) for each iteration, since the capacity is 1 for every edge, then the maximum flow is bounded by O(V*1) = O(V), thus the time complexity is O(VE)

2 Question 2

1. Tree is bipartite graph, (the nodes at depth of even number is one set, the nodes at depth of odd number is the other set).

2. Solving the maximum matching in bipartite graph by reducing it to Maximum flow problem similar to Question 1 with every edge of 1 capacity.
The time complexity is $O(VE)$

3 Question 3

Using 1 to mark the vertices in the independent set, and 0 to mark the vertices not in the set, the problem is of maximum independent set can be expressed as following linear programming problem:

$$\text{maximize } \sum x_i$$
subject to

1. $x_i + x_j \leq 1$ for every $\{i, j\}$ belongs to $E$ (Only one vertex of an edge is allowed to be in independent set)
2. $x_i = 0$ or $x_i = 1$ for all $i$

4 Question 4

1. Randomly generate an assignment.
2. If the assignment leads to CNF to be true, return result.
3. Randomly generate another assignment. and jump to 2 otherwise.

Since the CNF is satisfiable, the possibility is definitely larger than 0.

5 Question 5

1. Guess the $c-1$ points in the tape that divided it into $c$ part.
2. Insert $c-1$ # on the tape at those points.
3. Compare each $L_i$ in $L_1#L_2#\ldots#L_c$, and accept if they are all the same.