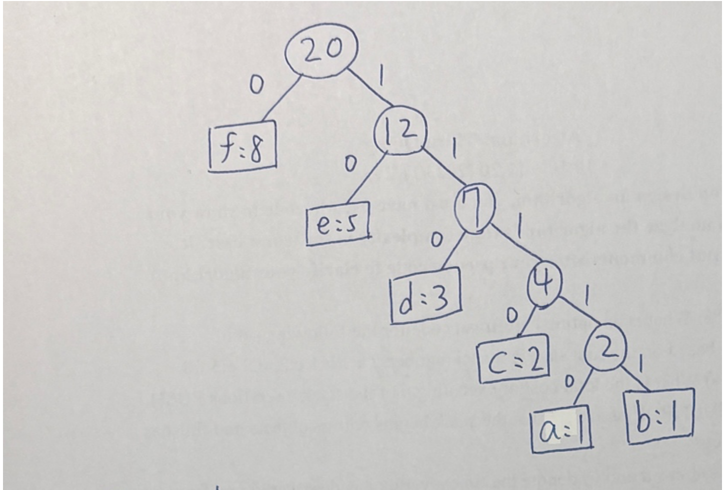
1.



Huffman code

$\Rightarrow$ a: 11110

b: 11111

c: 1110

d: 110

e: 10

f: 0

評分標準：

寫出正確且合理的 Huffman code，10 分

Huffman code 錯誤，但有畫出正確且合理的樹，5 分

2.

2\. push : $\phi(S_{m+1}) - \phi(S_m) + 1 = 2$

multipop(k) : $\phi(S_{m+1}) - \phi(S_m) + k = 0$

amortized cost $\leq 2n$

total cost $=$ amortized cost $- \phi(S_n) + \phi(S_0)$

$\leq 2n - S_n + S_0$

3.

First,

1. u is an ancestor of $v \Leftrightarrow u.d < v.d < v.f < u.f$.
2. u is a descendant of $v \Leftrightarrow v.d < u.d < u.f < v.f$.

Therefore,

a. (u, v) is a tree edge or forward edge $\Leftrightarrow$ u is an ancestor of v.(5%)

b. (u, v) is a back edge $\Leftrightarrow$ u is a descendant of v.(5%)

4.

4.
```
bool vis[V];
vector<int> G[V];
bool DFS(int u, int f){
    if(vis[u]) return 1;
    vis[u] = 1;
    bool cyc = 0;
    for(int v : G[u]){
        if(v == f) continue;
        cyc ||= DFS(v, u);
    }
    return cyc;
}
bool Cycle(int V){
    bool cyc = 0;
    for(int v = 1; v ≤ V; v++){
        if(!vis[v]) cyc ||= DFS(v, 0);
    }
    return cyc;
}
```

5.

You can either use Kruskal’s or Prim’s algorithm to solve this problem.

Kruskal's algorithm:

```
MST-KRUSKAL(G, w)
A = Ø
for each vertex v ∈ G.V
    MAKE-SET(v)
create a single list of the edges in G.E
sort the list of edges into monotonically increasing order by weight w
for each edge (u, v) taken from the sorted list in order
    if FIND-SET(u) ≠ FIND-SET(v)
        A = A ∪ {(u, v)}
        UNION(u, v)
return A
```

Time complexity of Kruskal's algorithm:

Use an edge array for sorting and use Union-Find operation.

The overall time complexity is $O(E \log V)$.

Prim's algorithm:

```
MST-PRIM(G, w, r)
for each vertex u ∈ G.V
    u.key = ∞
    u.π = NIL
r.key = 0
Q = Ø
for each vertex u ∈ G.V
    INSERT(Q, u)
while Q ≠ Ø
    u = EXTRACT-MIN(Q)          // add u to the tree
    for each vertex v in G.Adj[u]   // update keys of u's non-tree neighbors
        if v ∈ Q and w(u, v) < v.key
            v.π = u
            v.key = w(u, v)
            DECREASE-KEY(Q, v, w(u, v))
```

Time complexity of Prim's algorithm:

Use a binary heap as the priority queue.

The overall time complexity of Prim's algorithm is $O(E \log V)$.

Grading policy:

(7%) Design a correct algorithm.

(3%) Analyze the time complexity.