Interview Coding Problems – C++ and Python Solutions

1. Breadth First Search on a Linked List (with next and child pointers)

Python:

```
from collections import deque

def bfs_list(head):
if not head:
return []
q = deque([head])
visited = set()
result = []
while q:
node = q.popleft()
if node in visited:
continue
visited.add(node)
result.append(node.val)
if node.next:
q.append(node.next)
if hasattr(node, 'child') and node.child:
q.append(node.child)
return result
```

C++:

```
#include
#include
using namespace std;

struct Node {
int val;
Node* next;
Node* child;
```

```
};

vector bfsList(Node* head) {
if (!head) return {};
queue q;
unordered_set visited;
vector res;
q.push(head);

while (!q.empty()) {
Node* node = q.front(); q.pop();
if (visited.count(node)) continue;
visited.insert(node);
res.push_back(node->val);

if (node->next) q.push(node->next);
if (node->child) q.push(node->child);
}
return res;
}
```

2. Restore IP Addresses From Digits

Python:

```
def restore_ip_addresses(s):
res = []
def backtrack(i, parts):
if len(parts) == 4:
if i == len(s):
res.append(".".join(parts))
return
for length in range(1, 4):
if i + length > len(s):
break
seg = s[i:i+length]
```

```
if (seg.startswith("0") and length > 1) or int(seg) > 255:
continue
backtrack(i + length, parts + [seg])
backtrack(0, [])
return res
```

C++:

```
#include
#include
using namespace std;

class Solution {
public:
vector restoreIpAddresses(string s) {
vector res;
vector parts;
backtrack(s, 0, parts, res);
return res;
}

void backtrack(string &s;, int i, vector &parts;, vector &res;) {
if (parts.size() == 4) {
if (i == s.size()) {
res.push_back(parts[0] + "." + parts[1] + "." + parts[2] + "." + parts[3]);
}
return;
}
for (int len = 1; len <= 3; len++) {
if (i + len > s.size()) break;
string seg = s.substr(i, len);
if (seg.size() > 1 && seg[0] == '0') continue;
if (stoi(seg) > 255) continue;
parts.push_back(seg);
```

```
backtrack(s, i + len, parts, res);
parts.pop_back();
}
}
};
```

3. Edit Distance Between Two Binary Strings

Python:

```
def edit_distance(a, b):
n, m = len(a), len(b)
dp = [[0]*(m+1) for _ in range(n+1)]
for i in range(n+1):
dp[i][0] = i
for j in range(m+1):
dp[0][j] = j
for i in range(1, n+1):
for j in range(1, m+1):
if a[i-1] == b[j-1]:
dp[i][j] = dp[i-1][j-1]
else:
dp[i][j] = min(dp[i-1][j] + 1,
dp[i][j-1] + 1,
dp[i-1][j-1] + 1)
return dp[n][m]
```

C++:

```
#include
#include
using namespace std;

int editDistance(const string &a;, const string &b;) {
int n = a.size(), m = b.size();
vector> dp(n+1, vector(m+1));
```

```
for (int i = 0; i <= n; i++) dp[i][0] = i;
for (int j = 0; j <= m; j++) dp[0][j] = j;

for (int i = 1; i <= n; i++) {
for (int j = 1; j <= m; j++) {
if (a[i-1] == b[j-1])
dp[i][j] = dp[i-1][j-1];
else {
dp[i][j] = min({dp[i-1][j] + 1,
dp[i][j-1] + 1,
dp[i-1][j-1] + 1});
}
}
}
return dp[n][m];
}
```