Compte Rendu Corrigé - TP Algorithmique Avancée

EXERCICE 1 - Suite de Fibonacci

1. Fonction récursive en C++ :

```
int fibonacci_recursive(int n) {
    if (n <= 1)
        return n;
    return fibonacci_recursive(n - 1) + fibonacci_recursive(n - 2);
}
```

2. Fonction itérative avec tableaux :

```
int fibonacci_iteratif(int n) {
    if (n <= 1)
        return n;
    int fib[n+1];
    fib[0] = 0;
    fib[1] = 1;
    for (int i = 2; i <= n; i++)
        fib[i] = fib[i-1] + fib[i-2];
    return fib[n];
}
```

3. Comparaison des temps avec clock():

```
#include <iostream>
#include <ctime>
using namespace std;

int fibonacci_recursive(int n);
int fibonacci_iteratif(int n);

int main() {
```

```
    int n = 45;

    clock_t start1 = clock();
    int rec = fibonacci_recursive(n);
    clock_t end1 = clock();
    double time1 = double(end1 - start1) / CLOCKS_PER_SEC;

    clock_t start2 = clock();
    int it = fibonacci_iteratif(n);
    clock_t end2 = clock();
    double time2 = double(end2 - start2) / CLOCKS_PER_SEC;

    cout << "Fibonacci(" << n << ") Recursive = " << rec << ", Time: " << time1 << " sec" << endl;
    cout << "Fibonacci(" << n << ") Iterative = " << it << ", Time: " << time2 << " sec" << endl;

    return 0;
}
```

4. Version optimisée O(log n) par multiplication matricielle :

```
void multiply(long long F[2][2], long long M[2][2]) {
    long long x = F[0][0]*M[0][0] + F[0][1]*M[1][0];
    long long y = F[0][0]*M[0][1] + F[0][1]*M[1][1];
    long long z = F[1][0]*M[0][0] + F[1][1]*M[1][0];
    long long w = F[1][0]*M[0][1] + F[1][1]*M[1][1];
    F[0][0] = x; F[0][1] = y; F[1][0] = z; F[1][1] = w;
}

void power(long long F[2][2], int n) {
    if(n == 0 || n == 1)
        return;
    long long M[2][2] = {{1,1},{1,0}};
    power(F, n/2);
    multiply(F, F);
```

```
    if(n % 2 != 0)
        multiply(F, M);
}

long long fibonacci_matrix(int n) {
    long long F[2][2] = {{1,1},{1,0}};
    if(n == 0) return 0;
    power(F, n-1);
    return F[0][0];
}
```