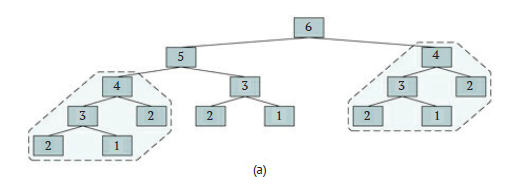
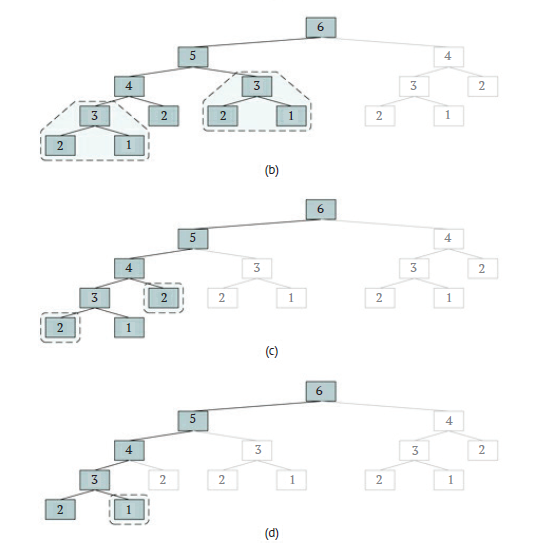
**23. Мемоізація, як спосіб прискорення рекурсії. Приклад.**

Мемоізація – це підхід, що дозволяє уникнути повторних обчислень при вирішенні перекриваючихся підзадач у тих рекурсивних алгоритмах, які декілька раз виконують одні й ті ж рекурсивні виклики. Суть підходу полягає у збереженні результатів рекурсивних викликів у пошукових таблицях. Тобто до рекурсивного виклику програма перевіряє, чи був такий же виклик раніше і чи збережений його результат у таблиці. Якщо так, тоді алгоритм замість повторного рекурсивного виклику просто повертає його раніш обчислений і збережений у таблиці результат.

Наприклад, функція Фібоначчі F(n) = F(n – 1) + F(n – 2):



Тут ми маємо два однакових піддерева fibonacci(4), тобто дві перекриваючих підзадачі, які виконують одне і те ж обчислення і повертають один і той же результат. По завершенню першого виклику fibonacci(4) алгоритм може зберегти його результат у пошуковій таблиці і скористатись їм згодом, щоб не виконувати зайвих обчислень. Так же само і зберігаються результати від піддерев fibonacci(3) і fibonacci(2), тобто у результаті час виконання алгоритму зводиться до лінійного часу, коли початковий алгоритм мав показникову складність:



#include <iostream>

#include <ctime>

using namespace std;

const int m = 100;

int mem[m];

int fib(int n)

{

if (n <= 1)

return n;

return fib(n - 1) + fib(n - 2);

}

void init()

{

for (int i = 0; i < m; i++) {

mem[i] = -1;

}

}

int fib\_mem(int n)

{

if (mem[n] == -1)

{

if (n <= 1)

mem[n] = n;

else

mem[n] = fib\_mem(n - 1) + fib\_mem(n - 2);

}

return mem[n];

}

int main()

{

int n, t1;

long int fibonacci, fibonacci\_mem;

cin >> n;

init();

t1 = clock();

fibonacci = fib(n);

cout << fibonacci << endl << clock() - t1 << endl;

t1 = clock();

fibonacci\_mem = fib\_mem(n);

cout << fibonacci\_mem << endl << clock() - t1 << endl;

return 0;

}