Đệ quy có nhớ (Recursion with memoization)

- Xét hàm đệ quy tính hằng số tổ hợp C(k,n) sau đây

Đệ quy có nhớ (Recursion with memoization)

- Xét hàm đệ quy tính hằng số tổ hợp C(k,n) sau đây: C(k, n) = { 1, nếu k = 0 hoặc k = n; C(k-1,n-1) + C(k, n-1), nếu ngược lại

```
public class Main {
    public static int C(int k, int n){
        if(k == 0 || k == n) return 1;
        return C(k-1,n-1) + C(k,n-1);
    }
    public static void main(String[] args) {
        double t0 = System.currentTimeMillis();
        System.out.println(C(16,32));
        double t = System.currentTimeMillis() - t0;
        System.out.println("Execution time = " + t*0.001);
    }
}
```

Đệ quy có nhớ (Recursion with memoization)

- Xét hàm đệ quy tính hằng số tổ hợp C(k,n) sau đây: C(k, n) = { 1, nếu k = 0 hoặc k = n; C(k-1,n-1) + C(k, n-1), nếu ngược lại

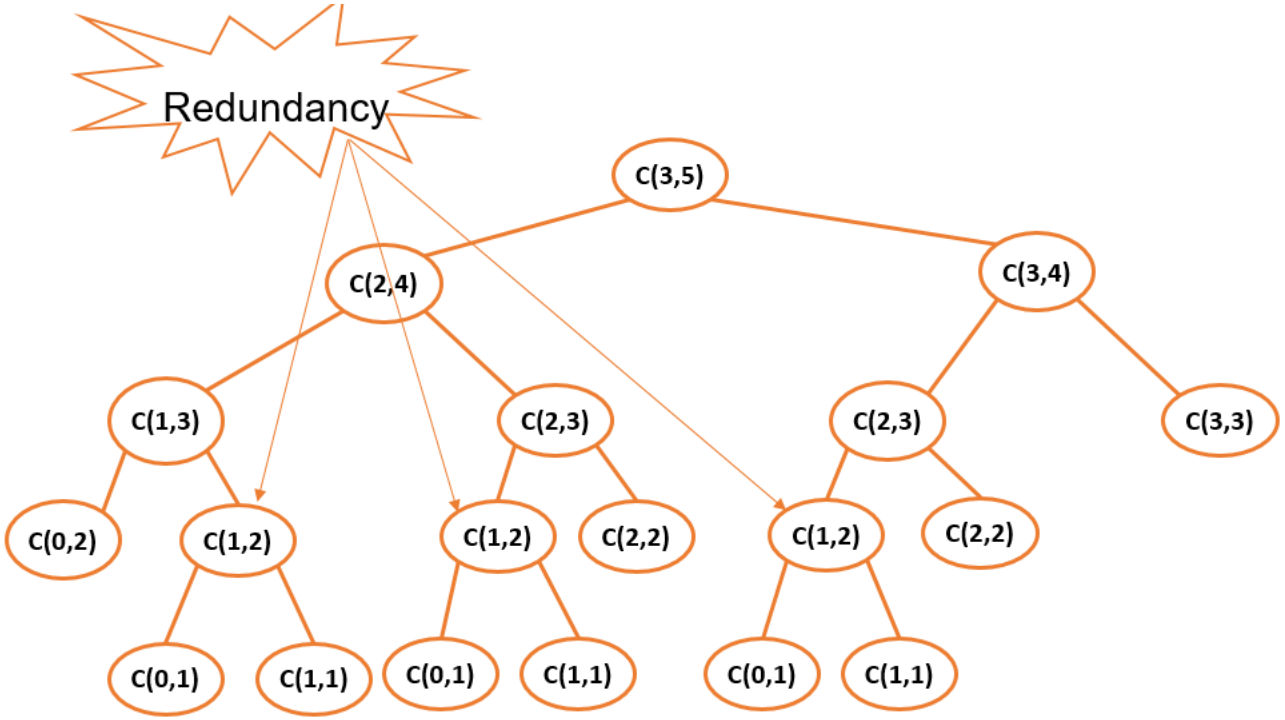
```
public class Main {
    public static int C(int k, int n){
        if(k == 0 || k == n) return 1;
        return C(k-1,n-1) + C(k,n-1);
    }
    public static void main(String[] args) {
        double t0 = System.currentTimeMillis();
        System.out.println(C(16,32));
        double t = System.currentTimeMillis() - t0;
        System.out.println("Execution time = " + t*0.001);
    }
}
```

```
601080390
Execution time = 1.456
```

Đệ quy có nhớ (Recursion with memoization)

- Xét hàm đệ quy tính hằng số tổ hợp C(k,n) sau đây: C(k, n) = { 1, nếu k = 0 hoặc k = n; C(k-1,n-1) + C(k, n-1), nếu ngược lại

```
public class Main {
    public static int C(int k, int n){
        if(k == 0 || k == n) return 1;
        return C(k-1,n-1) + C(k,n-1);
    }
    public static void main(String[] args) {
        double t0 = System.currentTimeMillis();
        System.out.println(C(16,32));
        double t = System.currentTimeMillis() - t0;
        System.out.println("Execution time = " + t*0.001);
    }
}
```



Đệ quy có nhớ (Recursion with memoization)

- Xét hàm đệ quy tính hằng số tổ hợp C(k,n) sau đây: C(k, n) = $\begin{cases} 1, \text{ nếu } k = 0 \text{ hoặc } k = n \\ C(k-1,n-1) + C(k, n-1), \text{ nếu ngược lại} \end{cases}$
- Đệ quy có nhớ
  - Dùng bộ nhớ để lưu kết quả các hàm (chương trình con) với các tham số khác nhau, mỗi hàm với 1 bộ tham số xác định được ánh xạ vào 1 ô nhớ xác định
  - Khi hàm được gọi lần đầu tiên thì cho hàm thực hiện, kết quả được lưu vào ô nhớ
  - Về sau, nếu lời gọi hàm với cùng bộ tham số trên xuất hiện, thì chỉ việc truy cập vào ô nhớ, lấy giá trị ra và sử dụng, khôn g cần tính toán lại

Đệ quy có nhớ (Recursion with memoization)

- Xét hàm đệ quy tính hằng số tổ hợp C(k,n) sau đây: C(k, n) = { 1, nếu k = 0 hoặc k = n; C(k-1,n-1) + C(k, n-1), nếu ngược lại
- Đệ quy có nhớ
  - Dùng bộ nhớ để lưu kết quả các hàm (chương trình con) với các tham số khác nhau, mỗi hàm với 1 bộ tham số xác định được ánh xạ vào 1 ô nhớ xác định
  - Khi hàm được gọi lần đầu tiên thì cho hàm thực hiện, kết quả được lưu vào ô nhớ
  - Về sau, nếu lời gọi hàm với cùng bộ tham số trên xuất hiện, thì chỉ việc truy cập vào ô nhớ, lấy giá trị ra và sử dụng, khôn g cần tính toán lại

```
import java.util.Arrays;
public class Main {
    static int[][] M = new int[100][100];
    public static int C(int k, int n){
        if(k == 0 || k == n)    M[k][n] = 1;
        else
            if(M[k][n] <= 0)  M[k][n] = C(k,n-1) + C(k-1,n-1);
        return M[k][n];
    }
    public static void main(String[] args) {
        for(int i = 0; i < 100; i++)  Arrays.fill(M[i],0);
        double t0 = System.currentTimeMillis();
        System.out.println(C(16,32));
        double t = System.currentTimeMillis() - t0;
        System.out.println("Execution time = " + t*0.001);
    }
}
```

Đệ quy có nhớ (Recursion with memoization)

- Xét hàm đệ quy tính hằng số tổ hợp C(k,n) sau đây: C(k, n) = { 1, nếu k = 0 hoặc k = n; C(k-1,n-1) + C(k, n-1), nếu ngược lại }
- Đệ quy có nhớ
  - Dùng bộ nhớ để lưu kết quả các hàm (chương trình con) với các tham số khác nhau, mỗi hàm với 1 bộ tham số xác định được ánh xạ vào 1 ô nhớ xác định
  - Khi hàm được gọi lần đầu tiên thì cho hàm thực hiện, kết quả được lưu vào ô nhớ
  - Về sau, nếu lời gọi hàm với cùng bộ tham số trên xuất hiện, thì chỉ việc truy cập vào ô nhớ, lấy giá trị ra và sử dụng, khôn g cần tính toán lại

```
import java.util.Arrays;
public class Main {
    static int[][] M = new int[100][100];
    public static int C(int k, int n){
        if(k == 0 || k == n)    M[k][n] = 1;
        else
            if(M[k][n] <= 0)  M[k][n] = C(k,n-1) + C(k-1,n-1);
        return M[k][n];
    }
    public static void main(String[] args) {
        for(int i = 0; i < 100; i++)  Arrays.fill(M[i],0);
        double t0 = System.currentTimeMillis();
        System.out.println(C(16,32));
        double t = System.currentTimeMillis() - t0;
        System.out.println("Execution time = " + t*0.001);
    }
}
```

```
601080390
Execution time = 0.0
```