# Binary Exponentiation (Fast Power Method)

## 1. Introduction

Binary Exponentiation is an efficient algorithm to calculate a^b (a raised to the power b) in O(log b) time instead of O(b). It is widely used in competitive programming, modular arithmetic problems, cryptography (RSA, Diffie-Hellman), and large number computations.

## 2. Recursive Approach

static final long MOD = 1\_000\_000\_007L;  
  
long power(long a, long b) {  
 if (b == 0) return 1; // base case: a^0 = 1  
 long half = power(a, b / 2) % MOD;  
 long result = (half \* half) % MOD;  
 if (b % 2 == 1) {  
 result = (result \* a) % MOD;  
 }  
 return result;  
}

👉 3:

.equals() and .compareTo() dono **String comparison** ke liye use hote hain, lekin inka **purpose aur return value alag** hota hai.

## ****1️⃣**** .equals()

* **Purpose:** Check karta hai ki **do strings bilkul same hain ya nahi** (character-by-character match).
* **Return type:** boolean (true ya false).
* **Case-sensitive:** "ABC".equals("abc") → false.

**Example:**

String a = "hello";

String b = "hello";

String c = "world";

System.out.println(a.equals(b)); // true

System.out.println(a.equals(c)); // false

👉 Bas **same ya different** batata hai.

## ****2️⃣**** .compareTo()

* **Purpose:** **Lexicographical comparison** (dictionary order) karta hai.
* **Return type:** int
  + 0 → Strings equal
  + > 0 → First string bada hai (dictionary order me)
  + < 0 → First string chhota hai
* **Case-sensitive** bhi hai.

**Example:**

String a = "hello";

String b = "hello";

String c = "world";

System.out.println(a.compareTo(b)); // 0 (equal)

System.out.println(a.compareTo(c)); // -15 (hello < world)

System.out.println(c.compareTo(a)); // 15 (world > hello

**1️⃣ Using String.valueOf() (Most common & safe)**

int num = 777;

String str = String.valueOf(num);

System.out.println(str); // "777"

Works for **any type**: int, double, boolean, char, even objects (calls toString() internally).

**1️⃣ Using Integer.parseInt()**

String s = "777";

int num = Integer.parseInt(s);

System.out.println(num); // 777

## ****2️⃣ Rule-based Memory Trick****

💡 Tumhe bas ye yaad rakhna hai:

| **String (Immutable)** | **StringBuilder (Mutable)** |
| --- | --- |
| Searching & checking (contains, startsWith, endsWith, matches) | Modifying (append, insert, delete, reverse, setCharAt) |
| Comparison (equals, compareTo) | Changing contents directly without making new object |

KNAPSACK:

TOP-DOWN APPROACH:

SUBSET SUM:

 boolean[][] dp=new boolean[n+1][sum/2+1];

         for(int i=0;i<=n;i++){

            dp[i][0]=true;

         }

         for(int i=1;i<=sum/2;i++){

            dp[0][i]=false;

         }

         for(int i=1;i<=n;i++){

            for(int j=1;j<=sum/2;j++){

                if(nums[i-1]<=j){

                    dp[i][j]=dp[i-1][j-nums[i-1]] || dp[i-1][j];

                }

                else{

                    dp[i][j]=dp[i-1][j];

                }

            }

         }

         return dp[n][sum/2];

in cpp:

count(st.begin(),st.end(),target) returns the frequency of the target;

in java:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

=>when an array in initialise globally bydefault all values will be 0; when locally then all the values bydefault will be garbage and if arr[]={}or {8} then rest all the values will become 0;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

=>array does not support bound checking means if the size is 5 and u are printing arr[6],arr[7] then it will be garbage value;

| **Feature** | **Java** | **C++** |
| --- | --- | --- |
| **Array bounds checking** | ✅ Always done at **runtime**. If an index is out of range, it throws **ArrayIndexOutOfBoundsException**. | ❌ Not done by default. Accessing invalid index leads to **undefined behavior** (garbage values, crashes, security issues). |