# Notes for competitive programming

James Sungarda

May 21, 2025

## 1 Useful algorithms

### 1.1 Binary exponentiation

The calculation $a^b$ can be calculated as $a^{b/2^2}$ for *even* b or $a^{b/2^2+1}$ for *odd* b.
We do this recursively and thus the time complexity is $O(\log b)$.

```
long long binpow(long long a, long long b) {
    long long result = 1;

    while (b > 0) {
        if (b % 2 == 1) result *= a;

        a *= a;
        b /= 2;
    }

    return result;
}
```

Note: The recursive version is a more intuitive way to understand the algorithm, but for efficiency sake we will be using the iterative version.
The idea of the iterative version works by the realizing that every number $b$ can be represented in binary.
*e.g.* $b = 13$ can be represented as $1101_2$ which is equal to $2^3 + 2^2 + 2^0$.
Thus we "multiply" result for every 1 in the binary representation of $b$.

### 1.2 Binary search

Binary search works by having two pointers *left* and *right* that point to the first and last elements of the array respectively.
The time complexity is $O(\log n)$.

```
int binary_search(int arr[], int n, int x) {
    int left = 0, right = n - 1;
    while (left <= right) {
        int mid = left + (right - left) / 2;
        if (arr[mid] == x) return mid;
        if (arr[mid] < x) left = mid + 1;
        else right = mid - 1;
    }
    return -1; // not found
}
```

# 2 Useful snippets

## 2.1 Fast input/output

Contains other ease of use functions.

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define fastio ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);

int32_t main() {
    fastio;

    // Your code here
}
```