1 RAM Model

1.1 Memory

Infinite sequence of cells, contains w bits. Every cell has an address starting at 1

1.2 CPU

32 registers of width w bits.

1.2.1 Operations

Set value to register (constant or from other register). Take two integers from other registers and store the result of; $a+b$, $a-b$, $a \cdot b$, a/b. Take two registers and compare them; $a < b$, $a = b$, $a > b$. Read and write from memory.

1.3 Definitions

An algorithm is a set of atomic operations. It's cost is is the number of atomic operations. A word is a sequence of w bits

2 Worst-case

Worst-case cost of an algorithm is the longest possible running time of input size n

3 Dictionary search

let n be register 1, and v be register 2
register $left \to 1$, $right \to 1$
while $left \leq right$
    register $mid \to (left + right)/2$
    if the memory cell at address $mid = v$ then
        return yes
    else if memory cell at address $mid > v$ then
        $right = mid - 1$
    else
        $left = mid + 1$
return no

Worst-case time: $f_2(n) = 2 + 6\log_2 n$

4 Big-O

We say that $f(n)$ grows asymptotically no faster than $g(n)$ if there is a constant $c_1 > 0$ such that $f(n) \leq c_1 \cdot g(n)$ and holds for all n at least a constant c_2. This is denoted by $f(n) = O(g(n))$.

4.1 Example

$1000\log_2 n = O(n)$,$n \neq O(10000\log_2 n)$
$\log_{b_1} n = O(\log_{b_2} n)$ for any constants $b_1 > 1$ and $b_2 > 1$. Therefore $f(n) = 2 + 6\log_2 n$ can be represented; $f(n) = O(\log n)$

5 Big-Ω

If $g(n) = O(f(n))$, then $f(n) = \Omega(g(n))$ to indicate that $f(n)$ grows asymptotically no slower than $g(n)$. We say that $f(n)$ grows asymptotically no slower than $g(n)$ if $c_1 > 0$ such $f(n) \geq c_1 \cdot g(n)$ for $n > c_2$; denoted by $f(n) = \Omega(g(n))$

6 Big-Θ

If $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$, then $f(n) = \Theta(g(n))$ to indicate that $f(n)$ grows asymptotically as fast as $g(n)$

7 Sort

7.1 Merge Sort

Divide the array into two parts, sort the individual arrays then combine the arrays together. $f(n) = O(n \log n)$.
This is the fastest sorting time possible (apart from $O(n \log\log n)$

7.2 Counting Sort

A set S of n integers and every integer is in the range [1, U]. (all integers are distinct)
Step 1: Let A be the array storing S. Create array B of length U. Set B to zero.
Step 2: For $i \in [1, n]$; Set x to A[i], Set B[x] = 1
Step 3: Clear A, For $x \in [1, U]$; If B[x] = 0 continue, otherwise append x to A

7.2.1 Analysis

Step 1 and 3 take O(U) time, while Step 2 O(n) time. Therefore running time is O(n + U) = O(U).

8 Random

RANDOM(x, y) returns an integer between x and y chosen uniformly at random