The Importance of Algorithms

Guilan Faculty of Engineering
Hesam Haddad

Today we are going to be talking about the Importance of Algorithms in Programming and providing their usage, optimization after definitions.

The main propose of this is to help you have a clear view of algorithms or some basic knowledge.

What's An Algorithm?

Algorithms are sets of rules to be followed in calculation or other problem solving by computer.

Runtime Analysis

After The correctness of an algorithm to solve all the test cases, the most important thing is runtime.

Computer Scientists typically talk about the runtime relative to the size of input. If the input consists of N integers, an algorithm witch has a runtime proportion to N^2 represent as $O(N^2)$ means that this algorithm terminate and complete after doing N^2 operations.

An Example Comparison

As a computer science student you all know at least one sorting algorithm, we are going to compare runtime of Bubble Sort, The most famous sorting algorithm with $O(N^2)$ runtime and Merge Sort , Another well-known sorting algorithm with $O(NlogN)$ when the input size is a bit large. Also we can estimate that a normal computer complete 10^8 operation per second.

$$N = 10^5$$

Bubble Sorts Runtime:

$$\frac{N^2}{10^8} = \frac{10^{10}}{10^8} = 100\ seconds$$

Merge Sort Runtime:

$$\frac{NlogN}{10^8} = \frac{17.10^5}{10^8} = 0.17\ seconds$$

As you know 100 seconds so differs from within a second for a user waiting. And we can understand that just finding a correct way to solve problem is not enough we need to do in in efficient way. N^2 is not so bad. There are some algorithms with $N!$ or 2^N runtime just imagine how terrible will they work when the size of input get bigger.

The Algorithm Developing Ways

They are not only one algorithm. They're ways to develop new algorithms no one will be able to explain them all even in a whole day. So I decided to give you one example in Divide and Conquer.

Name	Example or Description
Brute Force	Very General – Check All Possible Candidates
Divide and Conquer	Modular Power – Merge Sort
Dynamic Programming	Knapsack Problem – Longest Increasing Sequence
Max - Flow	Airline Scheduling
Greedy	Prim (Minimum Spanning Tree) – Activity Selection Problem
DFS	Connected Neighbors In Graph – Detecting Cycle In Graph
BFS	Shortest Path In 2dgrid
Binary Search	Finding Element In Sorted Array - Longest Increasing Sequence

Modular Power Problem

We need to design a fast algorithm getting 3 parameters a, n and k and return $a^n \ mod \ k$. Where the mod means remind divisor and you can use it as % sign in c like languages.

I will explain the both efficient and inefficient solutions.

The basic math rules used below as you all know by your high school knowledge.

$$N.M \ mod \ k = N \ mod \ k \ .M \ mod \ K$$

$$a^N.a^M = a^{N+M}$$

Inefficient solution:

We do the calculation easily just by one loop and then return the output. The runtime is $O(N)$.

```
int modpow(int a,int n, int k){
        int i,out=1%k;
        for(i=0;i<n;i++)
                out=out*a%k;
        return out;
}
```

Efficient solution:

We use the Divide and Conquer way and recursively call the function by half n. and if n was odd we return $rec.rec.a$ mod k because we drop the floating part of the division. By squaring it we will get $a^{n-1} \ mod \ k$ so we multiply it to another $a \ mod \ k$ to reach $a^n \ mod \ k$. The runtime is $O(logN)$.

When the n is even we haven’t lost anything and just return the square of sub problem in mod k.

We also have a stop point at n=0 I write the code in these part such a way to work even with a=n=k=1

```
int modpow(int a,int n, int k){
        if(n==0)return 1%k;
        int tmp=modpow(a,n/2,k);
        if(n%2) return (tmp*tmp*a)%k;
        else return (tmp*tmp)%k;
}
```

If we replace Int with $Unsigned\ Long\ Long\ Int$ we can use solution for numbers up to 10^{20} .

As a final conclusion we will compare runtime of these two solutions of Modular Power Problem.

Efficient Solution Runtime:

$$\frac{logN}{10^8} = \frac{53}{10^8} = within\ a\ second$$

Inefficient Solution Runtime:

$$\frac{N}{10^8} = \frac{10^{20}}{10^8} = 10^{12}\ seconds = 31710\ years$$

Even when we use a super computer that perform 1000 times faster than normal computers it will terminates after 31 years. So we understand the real importance of algorithms, even more than such expensive super computer in this example.

Recommended Books and links for learning Algorithms

Introduction to Algorithms _ Thomas H.Cormen

Art of Programming Contest

http://community.topcoder.com/tc?module=Static&d1=tutorials&d2=alg_index

http://www.geeksforgeeks.org/fundamentals-of-algorithms/