Design And Analysis Of Algorithms

Kunwar Shaanjeet Singh Grover

Contents

Chapter 1

Introduction

The course in one line: Better and Better Solutions to Problem.

But to understand this line, we need to define some terms in the line.

1.0.1 What do we mean by "Problems"?

Today, we'll try to appreciate what is so difficult about this question. Challenges: (We talk about sequential problems)

- Is it a Computational Problem?
- Is the problem well-posed?
- The problem of "Posing a problem without solving it"
- Infinite Ways of solving the same problem

Algorithms today dont only involve sequential problem. The challenges posed for sequential problems and the one line defination also are same for quantum, concurrent algo.

1.0.2 What do we mean by "Solution"

Challenges: (Sequenctial problem)

- It it a Computational Solution? A sequence of well defined steps which gives an answer if NOT a solution.
- Discrete vs Continous
- Is the Solution Realistic
- What problem is considered as solved by a solution

1.0.3 How to COMPARE various solutions?

To answer this we have more challenges: (Sequential algorithms)

- What is TIME?
- What is SPACE?
- Worst-case Analysis? Or-else
- Asymptotic Analysis. But Why?

Chapter 2

Computation Problems Computer Programs That Cannot Solve

2.0.1 Problems as membership queries in Languages

Decision Problems: A problem that takes an input and gives exactly one bit of output, where $1 \rightarrow$ true and $0 \rightarrow$ false.

We are not losing much in generality as we can split a problem which is not a decision problem into a number of decision problems. For example a problem having an output of 2bits can be split into 2 problems in which we get the first bit and second bit via two different decision problems.

Each decision problem is characterized by a subset of set of of all possible inputs (that is, subset of, say, $\{0, 1\}^*$)

Or **LANUAGES**

```
INPUT: String x;
OUTPUT: YES if x in language L, else NO
```

Example 1: Primality Testing
PRIMES = n — n is prime

Example 2: Graph Connectivity
GC = Graph G — G is connected

Example 3: Boolean Satisfiability
SAT = CNF Boolean Satisfiability f — f is not a fallacy; i.e f is satisfiable

A computational problem is a membership query in a language.

2.1 Countability and Computability

2.1.1 Countable Sets

An infinite set is countable if there is a bijection $f : \mathbb{N} \to S$.

Theorem 1. *The set of finite length bit string* $\{0,1\}^*$ *is countable*

Proof: Intutively we can strike up the 1-1 correspondence $f : \mathbb{N} \to \{0,1\}^*$ by listing the strings in short-lex order.

HW: Prove that f is 1-to-1 and onto.

Theorem 2. *The power set of bit strings* $P(\{0,1\}^*)$ *is uncountable.*

Proof: Any subset T in P can be viewed as a function $f : \{0,1\}^* \to \{0,1\}$. this is done by making the function take value 1 for elements in the set, and the value 0 for elements not in the set.

(Insert fig1)

Suppose that P were countable. Thus it would be possible to find a bijection from $\mathbb{N}$ to P.

Hence, we can list all binary languages as a sequence:

$$L_1, L_2, L_3, L_4, \ldots$$

supposedly containing all languages.

(Insert fig2)

Cantor's Diabolical Diagnol Argument: Take a language L' such that L_j is opposite of j^{th} element of L_j.

Cantor's diabolical diagonalization arguement shows that we can produce a language L' which is not in P. Therefore, our assumption that P was countable is false, and thus, P is uncountable.

□

2.1.2 Connection to Computing

We show that there are many more computational problems than computer program.

Claim 3. *Set of all programs is countable.*

Proof: Every computer program can be encoded in binary by some string. Consequently the cardinality of the set of programs is no greater than that of $\{0, 1\}^*$, which is countable.

Claim 4. *Set of all problems is uncountable.*

Proof: Every problem can be seen as a membership query in a language. Thus, the set of all problems is of the same cardinality as $P\{0, 1\}^*$, which is uncountable.

From 3 and , the set of problem is much bigger than the set of programs.