Decision Problems

1 Optimisation vs Decision Problems

- A major variation of optimization problems: decision problems
- Answer is not a value but YES/NO
- Every optimisation -problem has a decision counterpart
- An optimisation problem has a fast algorithm iff the corresponding decision problem has a fast algorithm

2 Decision problems and encodings

The standard way to define a decision problem is to describe a generic instance and a yes-no question about each instance

Reachability:

- Instance: A finite directed graph G and vertices s and t
- Question: Is there a path in G from s to t?

To input problems to a computer, each instance must be encoded as a string of symbols over some alphabet. We need an encoding scheme

To ensure that encoding the problem does not change its essential nature, an encoding scheme must be concise

- Represent numbers efficiently
- Not add unnecessary information

3 Languages

An alphabet, Σ is a (finite) set of symbols
A string over Σ is a finite sequence of symbols from Σ
A language over Σ is any set of strings over Σ

For a problem Π and an encoding scheme e with alphabet Σ, the set of all strings corresponding to instances with answer yes is denoted $\mathscr{L}(\Pi, e)$ and is called the language associated with Π and e

For decision problems, we just want to decide whether the (encoding of a) given instance belongs to the alphabet $\mathscr{L}(\Pi, e)$

4 Complexity of Problems

- The problems encountered so far in this course have all proved to be tractable (since we have found fast algorithms for all of them)
- There are many problems however, which cannot be quickly solved in practice, i.e., which are intractable
- There are many difficulties:
  - What do we mean by tractable and intractable
  - Can we define these notions formally
  - Can we prove that a problem is one but not the other
- One technique that will prove very useful: showing that one decision problem can be transformed(reduced) into another

5 Complexity measures

Every decidable problems has a set of algorithms that solved it. The properties of this set of algorithms:

- The difficulty of constructing the algorithm
- The length of the shortest possible algorithm
  - Static complexity measure
  - Useful for classifying the complexity of strings, called Kolmogorov complexity
- The efficiency of the most possible algorithm?
  - Dynamic complexity measure
  - A numerical function that measures the maximum resources used by an algorithm to compute the answer to a given instance

6 Dynamic complexity measures

- The most critical resources are often time/space
- By considering Turing Machines as our model of computation

Definition: Time complexity

The time complexity of a Turing Machine T is the function $Time_T$ such that $Time_T(x)$ is the number of steps taken by the computation $T(x)$

Definition: Space complexity

The space complexity of a Turing Machine T is the function $Space_T$ such that $Space_T(x)$ is the number of distinct tape cells visited during the computation $T(x)$

7 Equivalence

Theorem 1 *An n-vertex graph G has an independent set of size k iff G has a vertex cover of size n-k*

Corollary - The decision problems independent set and vertex cover are equivalent

8 Clique

- Instance - A graph G and an integer k
- Question: Does G have a clique of size at least k - i.e. a set of at least k vertices that are all adjacent to one another

9 Complement

Definition: Complement

The complement of a graph G has the same vertex set, and there is an edge between two vertices u and v in the complement iff there is no edge from u to v in G

Theorem 2 *A graph G has an independent set of size k iff its complement $\overline{G}$ has a clique of size k*

Corollary - The decision problems independent set and clique are equivalent

10 Set cover

- Instance - A set U of n elements, a collection of subsets $S_1, S_2, ..., S_t$ whose union equals I, and an integer k
- Question: Does there exist a collection of at most k of these sets whose union is equal to all of U

`VERTEX COVER` can be solved using an algorithm for `SET COVER`, but not the other way round