CS F364
Design & Analysis of Algorithms

ALGORITHMS - COMPLEXITY

Non-Deterministic Computation
- Ideas, Examples

Non-Deterministic Computations - Definition

- In a deterministic TM:
  - Computation proceeds by moving from one state to another in a "deterministic way"
    - i.e. given a state, and an input, the next state can be "determined"
    - i.e. the state transitions can be defined by a function (on set of states and set of symbols)
- By contrast, in a non-deterministic TM:
  - Computation proceeds by moving from one state to one of many states in a "non-deterministic" way
    - i.e. given a state, and an input, the next state is *not known a-priori* (before the transition)
    - i.e. the state transitions must be defined by a relation (on set of states and set of symbols)

Non-Deterministic Computations - Definition [2]

- Non-deterministic computations are defined using abstract machines:
  - There are no inherently non-deterministic computers (if you don't allow Schrodinger's cats!)

- Conceptually, a non-deterministic computation proceeds by making "non-deterministic" choices in real-time i.e.
  - Each choice takes O(1) time irrespective of the number of options available
    - as long as the number of choices is finite

Non-Deterministic Algorithms - Example 1

- Problem:
  - Given an array A of values, and a key k, find whether a value with key k is contained in A.
  - Algorithm NDSearch(A, k)
    1. len = A.length;
    2. ind = *choose(0, len-1)*;
    3. if (A[ind]==k) return 1;
    4. else return 0;

Non-Deterministic Algorithms - Example 2

- **Problem SAT(isfiability)**:
  - Given a Boolean expression E with variables $x_0,\ldots, x_{n-1}$ find whether there exists an assignment of (Boolean) values that satisfies E.
  - NDSAT(E, X)

```
1. len = X.length;
2. for(j=0;j <len;j++)    {  X[j] = choose(0, 1);  }
3. if (evaluate (E, X) == 1)    return 1;
4. else return 0;
```