Introduction to NP

Analysis of Algorithms
CSCI 5620

Decision Problem

- Decision problem: One for which the output can is a simple yes or no answer for each input.
- Example:
  - The input is a positive integer C and n positive integers s1, … sn.
  - Is there a subset of the integers which add up to C?

Decision Problem

- Decision Problem
  - Does the graph have a Spanning Tree with total weight K?
- Optimization Problem
  - What is the MST weight?
- Decision Problem
  - For a given graph, is there a path from A to B of length k?
- Optimization Problem
  - What is length of a shortest path from A to B?
- Optimization Problems are at least as hard as Decision Problems.
  - So if the optimization problem is easy, then the decision problem is easy.

NP

- NP “informally” is the class of decision problems whose answer “yes/no” can be verified in polynomial time given a “guess” or “certificate”.
- Example Does G contain a Hamiltonian cycle?
  - What is a guess?
  - Can it be verified in polynomial time?
  - How many total guesses?
  - How many does it take to prove G contains a cycle?
- Example: Is n a composite (not prime)?
  - Does d divide n? (can be checked in polynomial time)
- Example: Can graph G be colored with k colors?
  - Chromatic Number, smallest number of colors in which G can be colored (optimization problem)

Strings

- Catenation
  - If s_1=abac and s_2=dade, then $s_1 \cdot s_2$ or s_1s_2=abacdade.
- Empty string, λ=“
  - $s_1 \lambda = \lambda s_1$ =abac

Alphabet / Words

- For a set A, the set A* consists of all finite sequences of elements of A.
  - Ex A={0,1}
  - $\lambda \in A^*$, $0 \in A^*$, $1 \in A^*$, $01 \in A^*$, $10 \in A^*$
  - 000000000… $\notin A^*$
- A is called an alphabet, elements of A* are called words.
- Ex. A={a, b, c,…, z}, A* contains all words, such as “discrete”, but also “ababccad”.

Example

- Let A={ab, bc, ba}
  - Is ababab Є A*?
  - Is abc Є A*?
  - Is abba Є A*?
  - Is abbcbaba Є A*?
  - Is bcabbab Є A*?
  - Is abbbcba Є A* ?

Regular Expressions

- A ***<u>regular expression over A,</u>*** is a string constructed from elements of A and symbols (,), v, *, λ.
  - First λ is a regular expression.
  - If x Є A, then symbol x is a regular expression.
  - If α and β are regular expression then (α v β) is regular.
  - If α is a regular expression than (α)* is a regular expression.

Regular Expressions

- Ex. Let A={0,1}
- 0* is a regular expression.
- (01)* is a regular expression
- (0v1)* is a regular expression
- 1(0v1)* is a regular expression
- (011)* is a regular expression
- 0 v 1* is a regular expression
- (0 is not a regular expression

Regular Expressions

- Give the regular expression for the set of strings over {a,b} in which all the a's precede the b's which in turn precedes the c's.

Regular Set

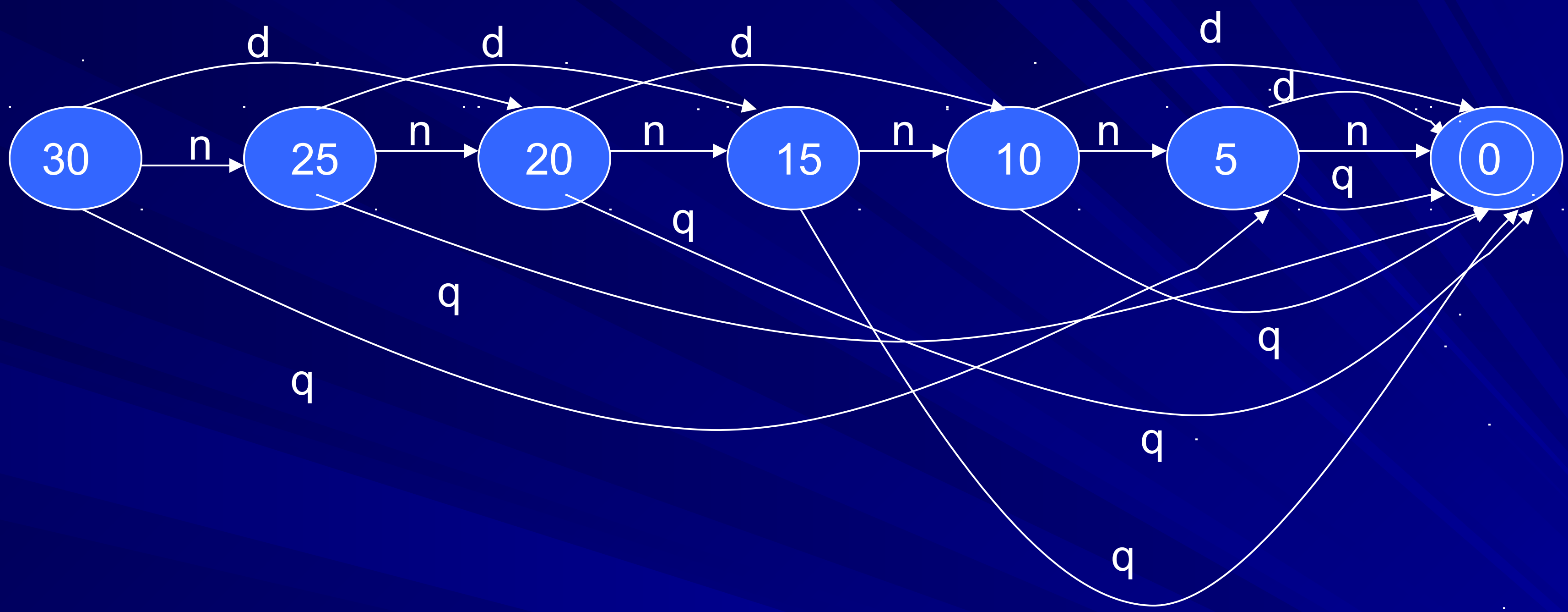
- For each regular expression, there is a corresponding subset of A*, called a ***<u>regular set.</u>***
- Example Let A={a, b}
  - For a*, the regular set Y contains
    - λ, a, aa, aaa, aaaa, aaaaa
    - Y={s, s = aaa…a (repeated n times for n$\geq$0)}
    - aaaaaaa… $\notin$ Y

Language

- Let Σ be an alphabet.
- A **language** over Σ is a subset of Σ^*.
- Computer Language
  - Contains words (if, then , else, etc)
- English Language
  - Σ ={a,b,c,d,….z}
  - Subset of Σ^*

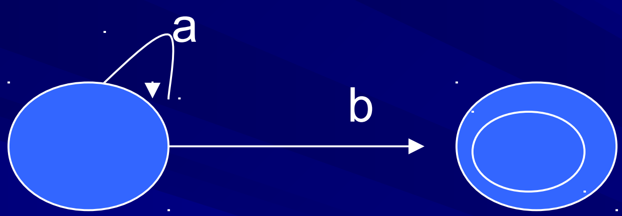
Deterministic Finite Automata (DFA)

- Finite State Machine
- Vending machine
  - Simple newspaper vending machine
  - 30 cents
  - No change given
  - Takes Nickels, dimes, and quarters
  - No memory used

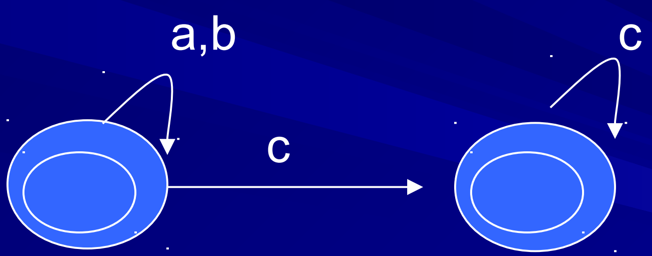

30
25
20
15
10
5
0
n
n
n
n
n
n
d
d
d
d
d
q
q
q
q
q
q
q

Regular expressions

- A={a,b,c}
- a*b



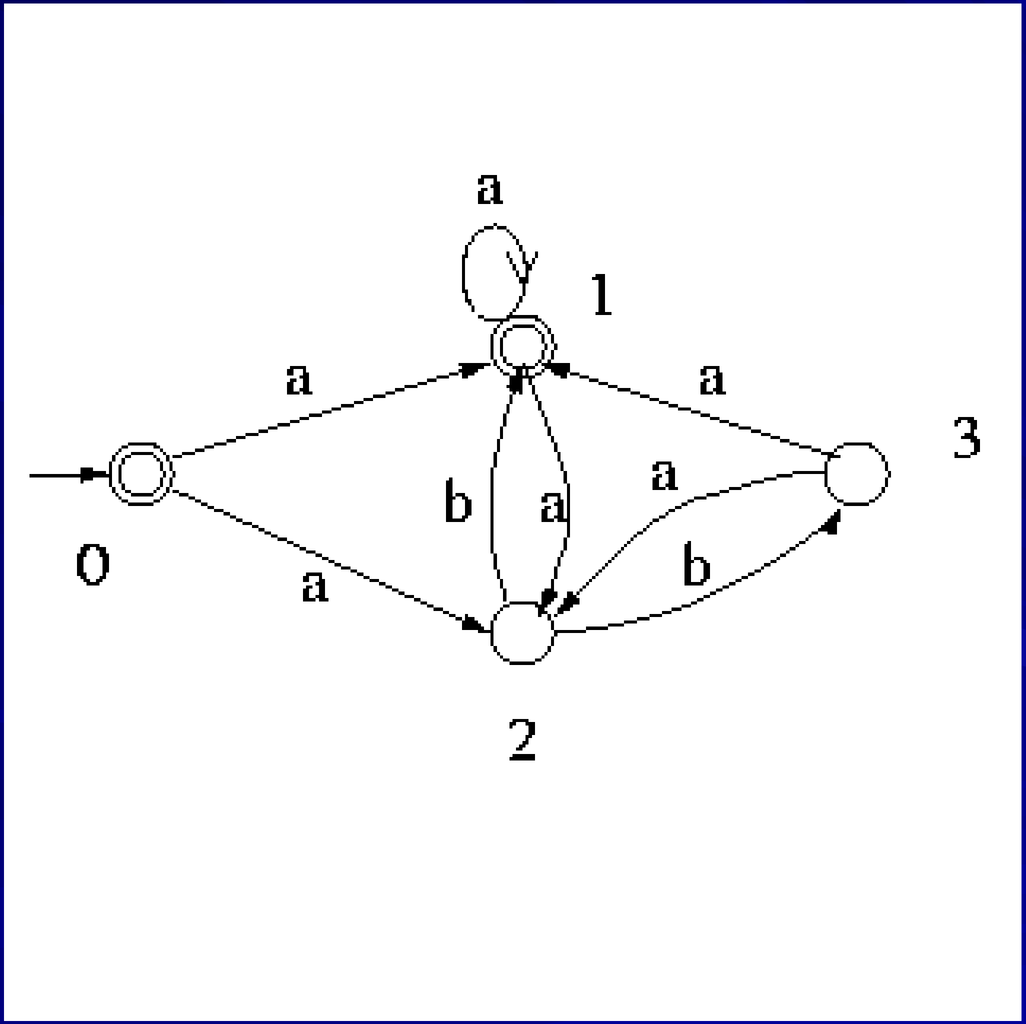
- (aub)*c*



Regular Language

- Equivalent to A regular set
- Accepted by a DFA (deterministic finite automation), NFA (nondeterministic finite automation).

NFA example



DFA example