Design Verification

Lecture 02 - Review on Notation, Graphs, and Algorithms

1. Set = a collection of elements
   - $|Set\ cardinality|$ = # elements
   - power set $P(A)$ = set of all subsets in A. $|P(A)| = 2^{|A|}$
   - Ordered Set
2. Relations: a R b = (a, b) $\in A \times B$; $a \in A, b \in B, R \in A \times B$
   - reflexive relation: (a, a)
   - symmetric relation: $(a, b) \in R \Rightarrow (b, a) \in R$
   - transitive relation: $(a, b) \in R, (b, c) \in R \Rightarrow (a, c) \in R$
   - partial order: reflexive, anti-symmetric, and transitive
   - equivalence relation: reflexive, symmetric, and transitive

   Let A = {a, b, c}, B = {1, 2}

   $A \times B =$

   $A \times A =$

   Example 1:

   - $(a, b) \in R$ if a did better than b $\Rightarrow$
   - $(a, b) \in R$ if a did not do worse than b $\Rightarrow$
   - $(a, b) \in R$ if a lives in same town as b $\Rightarrow$
   - $(a, b) \in R$ if a is ancestor of b $\Rightarrow$
   - $(a, b) \in R$ if a and b have the same major $\Rightarrow$
   - $(a, b) \in R$ if a divides b $\Rightarrow$
3. Partitions: division of a set into non-empty *disjoint* subsets

   $A = \{a, b, c, d, e\}$

   $\Pi = \{\{a, b\}, \{c, d, e\}\}$

4. Graphs $G = (V, E)$: V = set of vertices, E = binary relation on V. E can be ordered (directed) or unordered (undirected)

   *A directed graph induces partial order on V.

5. Computer representation of Graphs

   Example 2

   - Adjacency List

   - Adjacency Matrix

   - Incidence Matrix

   Bipartite graph: vertices can be partitioned into 2 sets such that each edge spans across two partitions

6. Some Properties/Definitions of Graphs

   - A path in G is a sequence of vertices $< V_0, V_1, ..., V_n >$ such that $< V_i, V_{i+1} > \in E$.
   - Simple path: all vertices in path distinct (i.e., no repeated vertex)

- Cycle: end vertices the same
- V (can be a set of vertices) is "reachable" from U if there exists some path(s) from U to V
- G is a "connected graph" if there exists a path from any vertex to any other vertex (all pairs of vertices joined by a path)
  Example 3

- Cut set: minimum set of edges that makes G disconnected
- Complete graph: every pair of vertices connected by an edge
- Clique: a complete subgraph of a graph G. *Clique number* $w(G)$ = cardinality of largest clique
- Planar graph: no edges intersect
- Isomorphism: relabeling of vertices in G_2 makes G_1
- Tree: connected graph with no cycles. There is an *unique* path between every pair of vertices, and $|E| = |V| - 1$

7. Problems & Algorithms

- decision problem: problem with binary-valued solution (true or false)
- optimization problem: solution measured as a cost function (performance, area, etc.). This can be reduced to a sequence of decision problems
- Algorithm: a computational procedure with a a set of inputs, outputs, a finite number of defined steps and terminates in finite number of steps
- Exact algorithm: always provides an exact solution (deterministic algorithms)
- Approximation algorithm: do not always provide exact solution, also called "heuristic algorithm"

8. Algorithm Complexity

- n = number of inputs
- $O(n) = c\ f(n)$ = complexity; c = some constant
- $O(n), O(n\ lg\ n), O(n^2), O(2^n)$

9. Tractability

   - P: There exists a polynomial time solution ($O(n^k)$)
   - NP: (non-deterministic polynomial): polynomial solutions can be derived on non-deterministic machines. Often intractable.
   - $P \subseteq NP$

10. Example Algorithms

    - Branch-and-bound:
      Step 1: branching: partition solution space
      Step 2: bounding: calculate lower bound on cost of solution in any solution in any partition

    - ILP (Integer Linear Programming)
      $A\mathbf{x} = \mathbf{b}$. Formulate problem into this form and solve for $\mathbf{x}$: $\mathbf{x} = A^{-1}\mathbf{b}$
      A^{-1} can be solved by various methods, including branching and bound
    - Dynamic Programming (an optimization solution based on bottom-up)
    - Greedy Algorithm (top-down)
      - sequence of local decisions
      - optimality not guaranteed
      - Example: packaging problem, start from biggest items down to smaller items $\rightarrow$ not optimal, but can be shown to be within bounds

11. Graph Problems

- Single-pair shortest path: given a pair of vertices in a connected graph G, find a path with shortest length

```
Initialize() {
   dist(v) = 0; dist(all other nodes) = infinity;
   mark(v) = 1; mark(all other nodes) = 0; }
scan(v);
```

```
scan(v) {
   mark(v) = 1;
   for every (v, w) ∈ E such that dist(v) + len(v,w) < dist(w) {
      dist(w) = dist(v) + len(v, w);
      scan(w); }
}
```

this is a depth-first search

Example 4

- Vertex-Cover Problem: Given an undirected graph G(V,E), a vertex cover is a subset of vertices such that each edge in E has at least one end point in the subset:

Example 5

An irredundant cover can be computed in linear time (by greedy algorithm)
Minimal cover requires branch-and-bound exact solution.

```
VC_1(V, E) {
   C = empty set;
   while (E ≠ empty) {
      pick a vertex v ∈ V; // best v
      delete v from G
      C = C ∪ {v};
   }
   return C;
}
```

* VC_1() selects vertex with largest degree (most number of edges incident)
* worst case: greater than 2 times minimal cover

```
VC_2(V, E) {
   C = empty set;
   while (E ≠ empty) {
      pick (v, w) ∈ E; // best edge
      C = C ∪ {v} ∪ {w};
      delete v & w from G
   }
   return C;
}
```

* VC_2() picks edge incident on the largest degree vertices
* worst case: at most 2 times minimal cover
* Note: both VC_1 and VC_2 can result in redundant covers.