Brute Force Algorithms

Algorithmic Thinking

Luay Nakhleh

Department of Computer Science

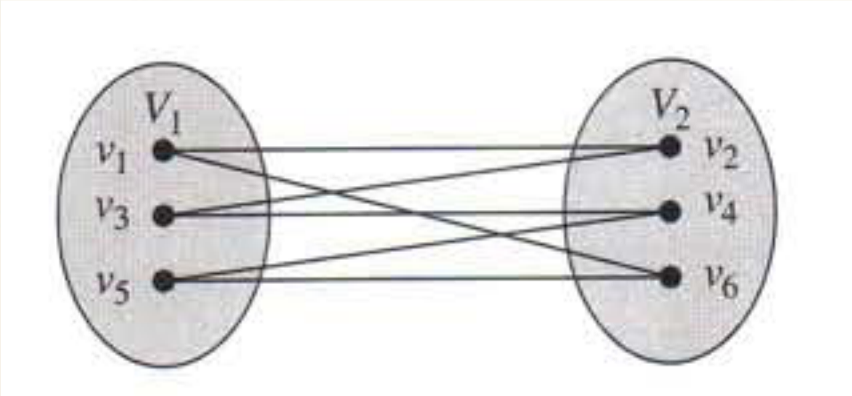
Rice University

Brute Force Algorithms

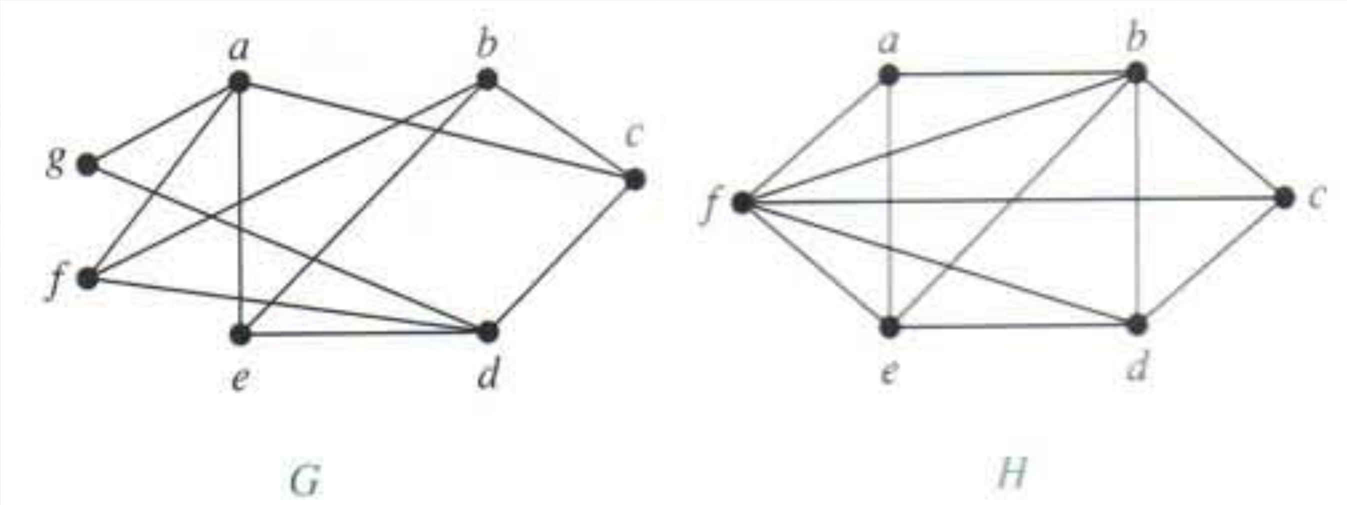
- A brute force algorithm is a solution that is based directly on the problem definition.
- It is often easy to establish the correctness of a brute force algorithm.
- This algorithmic strategy applies to almost all problems.
- Except for a small class of problems, this algorithmic strategy produces algorithms that are prohibitively slow.

Bipartite Graphs

- A graph G is called bipartite if its node set V can be partitioned into two disjoint and non-empty sets V_1 and V_2 such that every edge in the graph connects a node in V_1 and a node in V_2. When this condition holds, we call the pair (V_1,V_2) a bipartition of the node set V of G.



Are the following graphs bipartite?



Is a Given Graph Bipartite? A Brute Force Algorithm

Algorithm 1: IsBipartite.

Input: Undirected graph $g = (V, E)$.
Output: *True* if g is bipartite, and *False* otherwise.

foreach *Non-empty subset* $V_1 \subset V$ **do**
    $V_2 \leftarrow V \setminus V_1$;
    $bipartite \leftarrow True$;
    foreach *Edge* $\{u, v\} \in E$ **do**
        if $\{u, v\} \subseteq V_1$ *or* $\{u, v\} \subseteq V_2$ **then**
            $bipartite \leftarrow False$;
            Break;
    if $bipartite = True$ **then**
        return *True*;
return *False*;

Graph Connectivity: Paths

- Let k be a nonnegative integer and G a graph.
  - A <u>path of length k</u> from node v_0 to node v_k in G is a sequence of k edges $e_1,e_2,\ldots,e_k$ of G such that $e_1=\{v_0,v_1\}$, $e_2=\{v_1,v_2\}$, ..., $e_k=\{v_{k-1},v_k\}$, where $v_0,\ldots,v_k$ are all nodes in V, and $e_1,\ldots,e_k$ are all edges in E.
- We usually denote such a path by its node sequence $(v_0,v_1,\ldots,v_k)$.
- A path is <u>simple</u> if it does not contain the same node more than once.
- A <u>cycle</u> is a simple path that begins and ends at the same node.
- A path (not necessarily simple) that begins and ends at the same node is called a <u>circuit</u>.

Is There a Path Between i and j? A Brute Force Algorithm

Algorithm 2: IsConnected.

Input: Undirected graph $g = (V, E)$, $|V| \geq 2$, and two nodes $u, v \in V$, such that $u \neq v$.
Output: $True$ if there is a path between u and v in g, and $False$ otherwise.

```
Nodes ← V − {u, v};
for c ← 0 to |Nodes| do
    x_0 ← u;
    x_{c+1} ← v;
    foreach subset W ⊆ Nodes of size c do
        foreach permutation x_1, ..., x_c of the elements of W do
            Connected ← True;
            for i ← 0 to c do
                if {x_i, x_{i+1}} ∉ E then
                    Connected ← False;
                    Break;
            if Connected = True then
                return True;
return False;
```

The Shortest Path

- Given a graph $G=(V,E)$, and two connected nodes $i,j \in V$, a <u>shortest path</u> between i and j is a path P that connects the two nodes and every other path that connects i and j is either longer than P or of equal length.

- The shortest path between two nodes may not be unique.

- The <u>distance</u> between two nodes in the length of a shortest path between them.

The Shortest Path

- How would you modify IsConnected to produce a brute-force algorithm for finding
  - the distance between two given nodes i and j?
  - a shortest path between two given nodes i and j?
  - all shortest paths between two given nodes i and j?

The Clique Problem

- **Input**: Graph G=(V,E) and positive integer k.
- **Question**: Does G contain a clique of size ≥k, that is, a complete subgraph of size ≥k?
- Describe a brute-force algorithm for the problem.

The Traveling Salesman Problem

- **Input**: Graph G=(V,E).
- **Output**: The shortest Hamiltonian cycle of G, that is, the shortest cycle that visits all nodes of G exactly once.
- Describe a brute-force algorithm for the problem.

The Post Correspondence Problem

- **Input:** Two finite lists of words $x_1,x_2,...,x_n$ and $y_1,y_2,...,y_n$ over an alphabet that has at least two letters.
- **Output:** A sequence of indices i1,i2,...,iM, where M≥1, all index values are between 1 and n, and $x_{i1}x_{i2}...x_{iM}=y_{i1}y_{i2}...y_{iM}$.

The Post Correspondence Problem

x_1	x_2	x_3	x_4
ab	bbaaba	b	bb

y_1	y_2	y_3	y_4
a	a	bbbb	ab

The Post Correspondence Problem

x_1	x_2	x_3	x_4
ab	bbaaba	b	bb

y_1	y_2	y_3	y_4
a	a	bbbb	ab

Solution: Indices 1,3,2,4,4,3

(to verify: check that $x_1x_3x_2x_4x_4x_3=y_1y_3y_2y_4y_4y_3$)

The Post Correspondence Problem

x_1	x_2	x_3	x_4
ab	bbaaba	b	bb

y_1	y_2	y_3	y_4
a	a	bbbb	ab

Solution: Indices 1,3,2,4,4,3

(to verify: check that $x_1x_3x_2x_4x_4x_3=y_1y_3y_2y_4y_4y_3$)

x_1	x_2
a	ab

y_1	y_2
ba	b

The Post Correspondence Problem

x_1	x_2	x_3	x_4
ab	bbaaba	b	bb

y_1	y_2	y_3	y_4
a	a	bbbb	ab

Solution: Indices 1,3,2,4,4,3
(to verify: check that $x_1x_3x_2x_4x_4x_3=y_1y_3y_2y_4y_4y_3$)

x_1	x_2
a	ab

y_1	y_2
ba	b

No Solution!

The Post Correspondence Problem

- Devise a brute-force algorithm for PCP????

A Taxonomy of the Problems

- Group 1: [We ca do much better than brute force]
  - Bipartiteness, connectivity, shortest paths, and distance
- Group 2: [We can’t do much better than brute force]
  - Clique, traveling salesman
- Group 3: [Even brute force doesn’t work; there is no algorithm]
  - Post correspondence problem

- What's wrong with the algorithms we've seen?
- Stay tuned for efficiency analysis!