Hopcroft-Karp Algorithm

1905047-Rakib Abdullah
1905048-Md. Al-Amin Sany
1905052-Bijoy Ahmed Saiem

Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology

February 25, 2023

Table of Contents

Introduction

Hopcroft-Karp Algorithm

The Hopcroft-Karp algorithm is a graph algorithm that finds the maximum cardinality matching in a bipartite graph.

Introduction

Hopcroft-Karp Algorithm

The Hopcroft-Karp algorithm is a graph algorithm that finds the maximum cardinality matching in a bipartite graph.

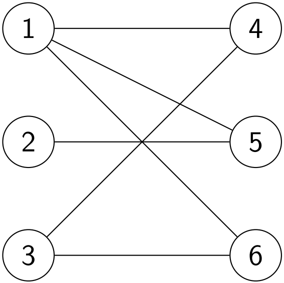


Figure: Bipartite graph

Introduction

Hopcroft-Karp Algorithm

The Hopcroft-Karp algorithm is a graph algorithm that finds the maximum cardinality matching in a bipartite graph.

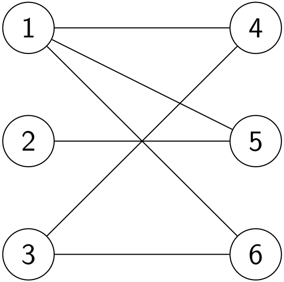


Figure: Bipartite graph

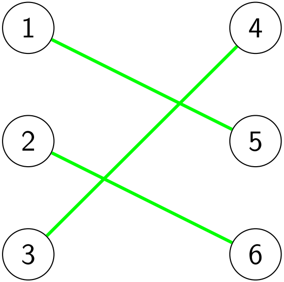


Figure: Maximum cardinality matching

Table of Contents

Definitions

Bipartite Graph

A graph is bipartite if its vertices can be divided into two disjoint sets such that every edge connects a vertex in one set to a vertex in the other set.

Definitions

Bipartite Graph

A graph is bipartite if its vertices can be divided into two disjoint sets such that every edge connects a vertex in one set to a vertex in the other set.

Maximum Cardinality

A matching in a graph is a set of edges such that no two edges share a common vertex.Maximum cardinality means maximum possible matching.

Definitions

Bipartite Graph

A graph is bipartite if its vertices can be divided into two disjoint sets such that every edge connects a vertex in one set to a vertex in the other set.

Maximum Cardinality

A matching in a graph is a set of edges such that no two edges share a common vertex.Maximum cardinality means maximum possible matching.

Free vertex

A free vertex is a vertex with no matching edge connected to it.

Definitions

Bipartite Graph

A graph is bipartite if its vertices can be divided into two disjoint sets such that every edge connects a vertex in one set to a vertex in the other set.

Maximum Cardinality

A matching in a graph is a set of edges such that no two edges share a common vertex.Maximum cardinality means maximum possible matching.

Free vertex

A free vertex is a vertex with no matching edge connected to it.

Augmenting Path

- Starts and ends both at free vertex
- Edges in the path alternate between being in the matching and not in the matching

Table of Contents

Algorithm

Hopcroft-Karp(G)

$M = \phi$

Algorithm

Hopcroft-Karp(G)

```
M = φ
repeat
    P = {p_1, p_2, ..., p_k}  //maximal set of vertex-disjoint shortest augmenting paths
```

Algorithm

Hopcroft-Karp(G)

$M = \phi$
repeat
    $P = \{p_1, p_2, \ldots, p_k\}$ //maximal set of vertex-disjoint shortest augmenting paths
    $M := M \oplus \{p_1 \cup p_2 \cup \cdots \cup p_k\}$

Algorithm

Hopcroft-Karp(G)

$M = \phi$
repeat
    $P = \{p_1, p_2, \ldots, p_k\}$ //maximal set of vertex-disjoint shortest augmenting paths
    $M := M \oplus \{p_1 \cup p_2 \cup \cdots \cup p_k\}$
until $P = \phi$

Algorithm

Hopcroft-Karp(G)

$M = \phi$

repeat

  $P = \{p_1, p_2, \ldots, p_k\}$ //maximal set of vertex-disjoint shortest augmenting paths

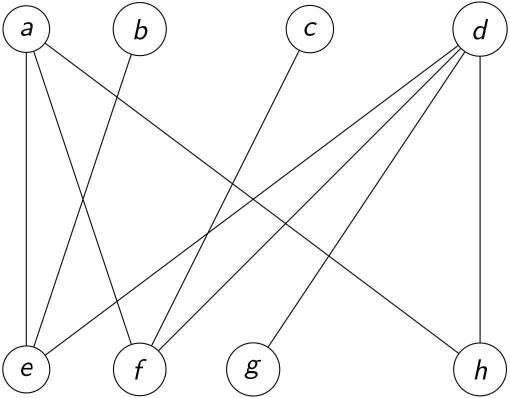
  $M := M \oplus \{p_1 \cup p_2 \cup \cdots \cup p_k\}$

until $P = \phi$

Output the matching M as the maximum cardinality matching.

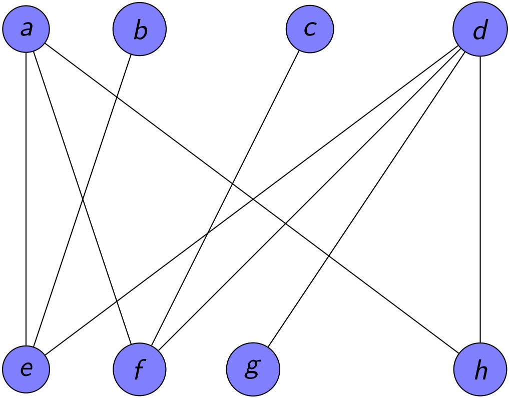
Example

A bipartite graph is given below where we will find its maximum cardinality mathching:



First iteration:BFS

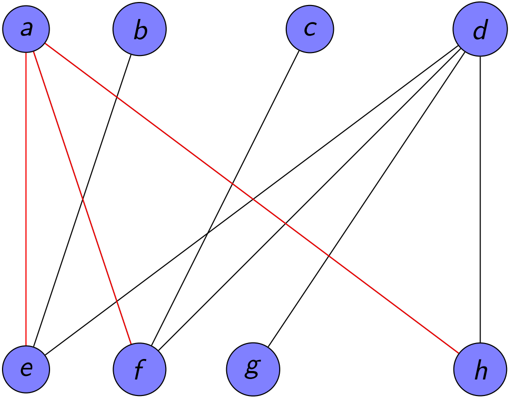
$$\text{BFS sources} = \{a, b, c, d\}$$
$$F = \{\}$$



First iteration:BFS

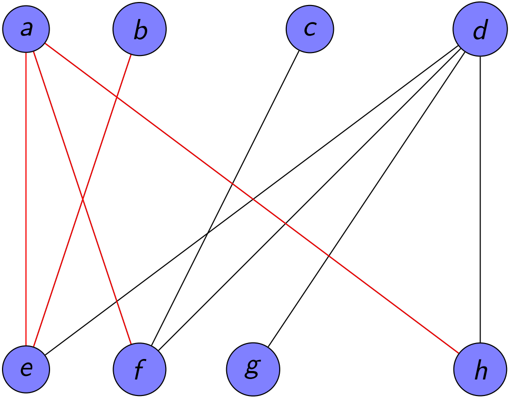
BFS sources $= \{a, b, c, d\}$

$F = \{e, f, h\}$



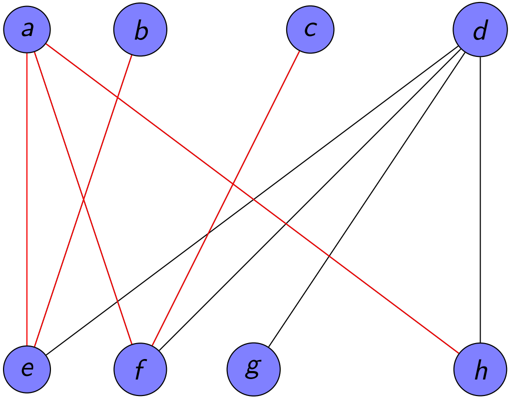
First iteration:BFS

BFS sources $= \{a, b, c, d\}$
$F = \{e, f, h\}$



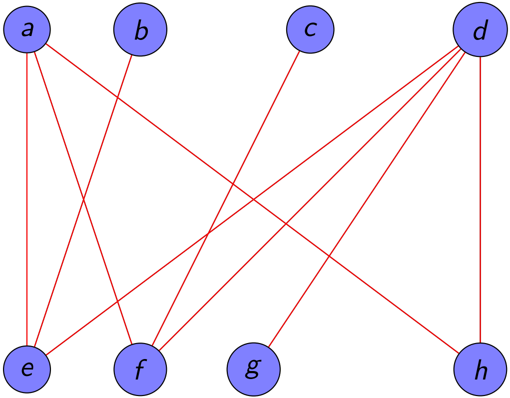
First iteration:BFS

BFS sources $= \{a, b, c, d\}$
$F = \{e, f, h\}$



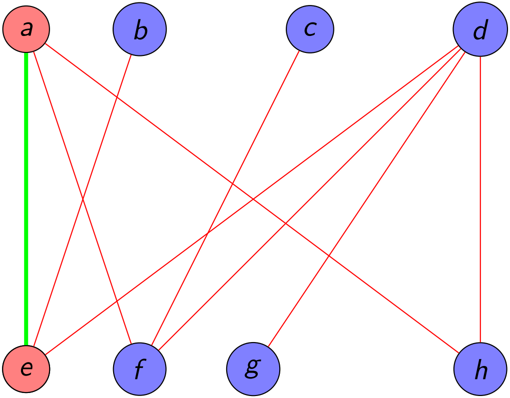
First iteration:BFS

BFS sources $= \{a, b, c, d\}$
$F = \{e, f, h, g\}$



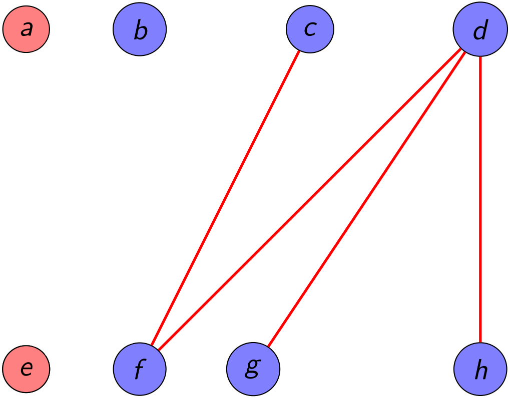
First iteration:DFS

$$F = \{e, f, h, g\}$$
$$P = \{(e - a)\}$$



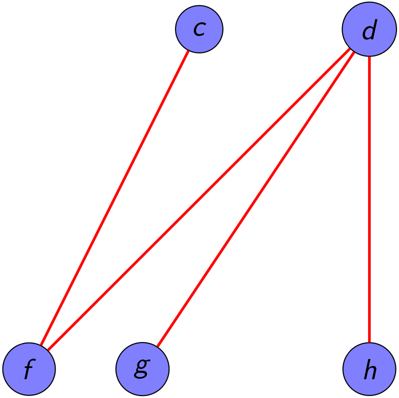
First iteration:DFS

$F = \{e, f, h, g\}$
$P = \{(e - a)\}$



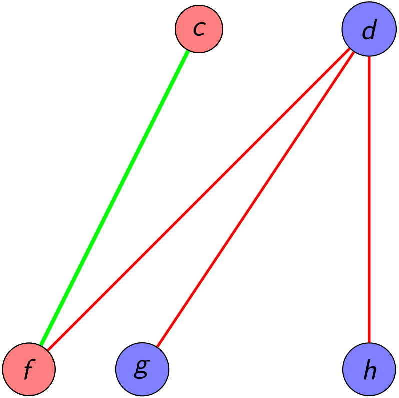
First iteration:DFS

$$F = \{f, h, g\}$$
$$P = \{(e - a)\}$$



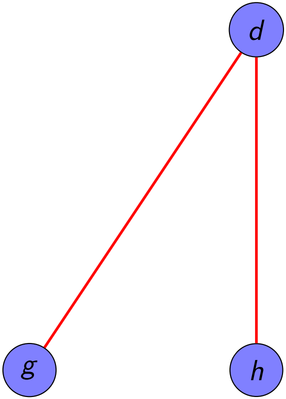
First iteration:DFS

$$F = \{f, h, g\}$$
$$P = \{(e - a), (f - c)\}$$



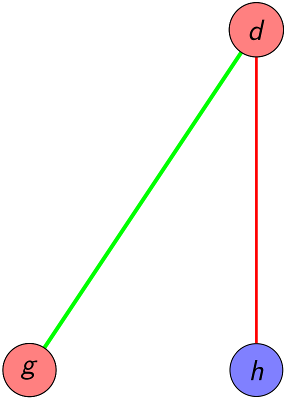
First iteration:DFS

$$F = \{h, g\}$$
$$P = \{(e - a), (f - c)\}$$



First iteration:DFS

$F = \{h, g\}$
$P = \{(e - a), (f - c), (g - d)\}$



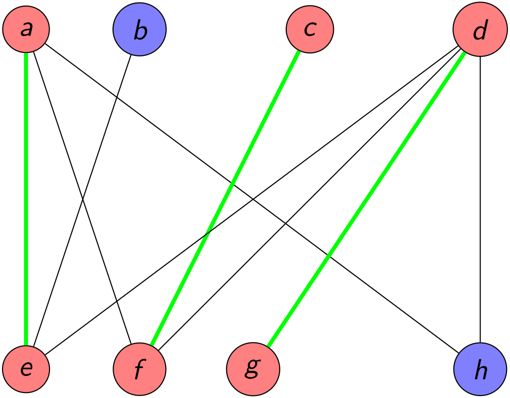
First iteration:DFS

$F = \{h\}$
$P = \{(e-a),(f-c),(g-d)\}$
Before $M = \phi$
After $M = M \oplus \{(e-a)\cup(f-c)\cup(g-d)\}$
$M = \{(e-a),(f-c),(g-d)\}$

After first iteration

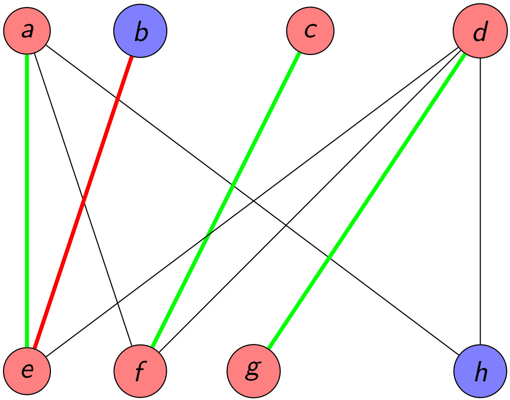
After running the DFS,

$$M = \{(e - a), (f - c), (g - d)\}$$



Second iteration:BFS

BFS sources $= \{b\}$
$F = \{\}$
BFS tree $= b- > e$

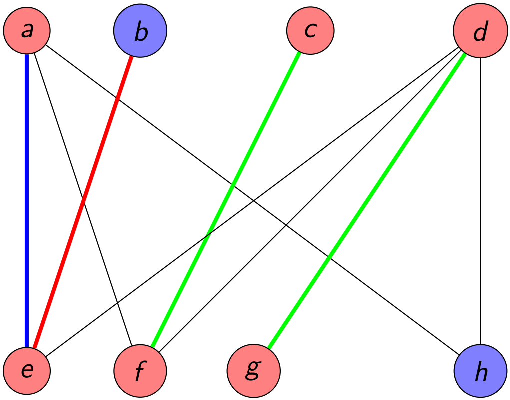


Second iteration:BFS

BFS sources $= \{b\}$

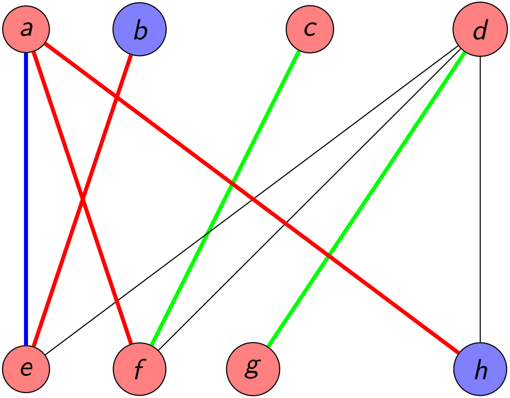
$F = \{\}$

BFS tree $= b- > e- > a$



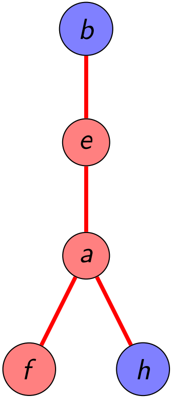
Second iteration:BFS

BFS sources $= \{b\}$
$F = \{h\}$
BFS tree $= b- > e- > a- > f, h$



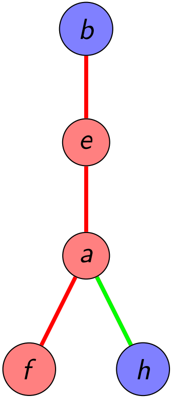
Second iteration:DFS

DFS sources $= \{h\}$
$P = \{\}$



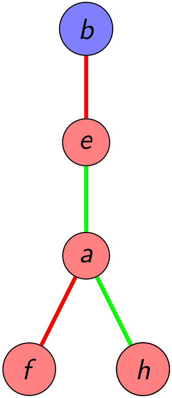
Second iteration:DFS

DFS sources $= \{h\}$
$P = \{(h - a-\}$



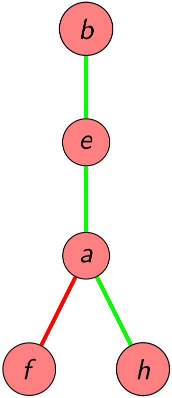
Second iteration:DFS

DFS sources $= \{h\}$
$P = \{(h - a - e-\}$



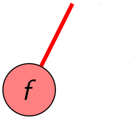
Second iteration:DFS

DFS sources $= \{h\}$
$P = \{(h - a - e - b)\}$



Second iteration:DFS

DFS sources $= \{\}$
$P = \{(h - a - e - b)\}$



Second iteration:DFS

$P = \{(h - a - e - b)\}$
Before $M = \{(a - e), (f - c), (g - d)\}$
After $M = M \oplus \{(h - a), (a - e), (e - b)\}$
$M = \{(h - a), (e - b), (f - c), (g - d)\}$

Algorithm termination

As no more free vertex is available in upper set,the algorithm terminates and finally we get the following graph with maximum cardinaliy 4 where $M = \{(h-a),(e-b),(f-c),(g-d)\}$

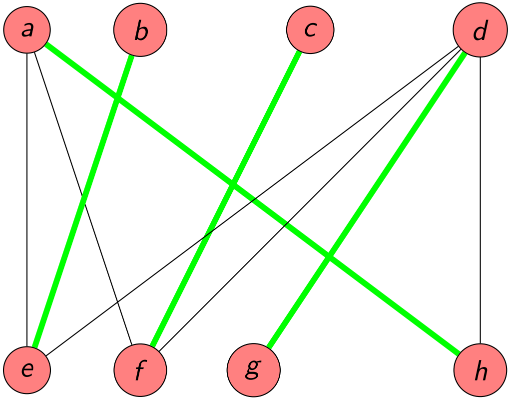


Table of Contents

Complexity Analysis

Hopcroft-Karp(G)

```
M = φ //O(1)
repeat
    P = {p_1, p_2, ..., p_k}
    M := M ⊕ {p_1 ∪ p_2 ∪ ··· ∪ p_k}
until P = φ
Return M //O(1)
```

Complexity Analysis

Per Loop Iteration

Breadth First Search: Uses all edges at most once so time complexity is $O(|E|)$

Depth First Search: Since vertices and edges are deleted once used,all edges are also used at most once,so time complexity is $O(|E|)$

S**o over all each iteration is linear with the number of edges. i.e** $O(|E|)$

Complexity Analysis

- Breadth first search terminates when it reaches the free vertex.Therefor,there is no shorter path to a free vertex.Therefore,in subsequent iterations a shorter path cannot be found.
- Once it terminates it collects all free vertex on that level so all possible paths of that length are found.Therefore,in subsequent iterations the paths found must be longer.
- Since the paths alternate between matched and unmatched edges,and free vertices cannot be connected to a matched edge,then in subsequent iterations the paths must be at least two edges longer.

Complexity Analysis

- After $\sqrt{|V|}$ iterations the minimum path length would therefore be $2\sqrt{|V|}$
- Since in $P = \{p_1, p_2, \ldots, p_k\}$ the paths are vertex disjoint and there are only $|V|$ vertices in the graph,then there can only be $\frac{|V|}{2\sqrt{|V|}} = \frac{1}{2}\sqrt{|V|}$
- Therefore after $\sqrt{|V|}$ iterations only $\frac{1}{2}\sqrt{|V|}$ more are needed so the loop will terminate after $\frac{3}{2}\sqrt{|V|}$ repeats.

Table of Contents

Applications

- Dating app-The algorithm can be used in online dating apps to match people based on their interests and preferences.

Applications

- Dating app-The algorithm can be used in online dating apps to match people based on their interests and preferences.
- Image segmentation - finding matches between objects in an image and a pre-defined set of object templates.

Applications

- Dating app-The algorithm can be used in online dating apps to match people based on their interests and preferences.
- Image segmentation - finding matches between objects in an image and a pre-defined set of object templates.
- Job scheduling - matching workers with tasks based on their skills and availability.

Applications

- Dating app-The algorithm can be used in online dating apps to match people based on their interests and preferences.
- Image segmentation - finding matches between objects in an image and a pre-defined set of object templates.
- Job scheduling - matching workers with tasks based on their skills and availability.
- Online advertising - matching ads with potential viewers based on demographic and behavioral data.

Thank You!

Gracias por todo

- Thank you for your attention.
- We appreciate your support.