

Implementation of algorithms and data structures

9. seminar

Jirka Fink

<https://ktiml.mff.cuni.cz/~fink/>

Department of Theoretical Computer Science and Mathematical Logic
Faculty of Mathematics and Physics
Charles University in Prague

Summer semestr 2021/22

Last change 4. května 2022

Licence: Creative Commons BY-NC-SA 4.0

Definitions

- (V, E) is an undirected graph on n vertices and m edges
- $M \subseteq E$ is a matching if every vertex is incident with at most one edge of M
- A vertex $v \in V$ is *M-covered* if some edge of M is incident with v ; otherwise v is *M-exposed*
- Matching M is perfect if all vertices are *M-covered*
- Matching is maximal if there is no matching having more edges

Definitions

- (V, E) is an undirected graph on n vertices and m edges
- $M \subseteq E$ is a matching if every vertex is incident with at most one edge of M
- A vertex $v \in V$ is *M-covered* if some edge of M is incident with v ; otherwise v is *M-exposed*
- Matching M is perfect if all vertices are *M-covered*
- Matching is maximal if there is no matching having more edges

Related problems

- Find a maximum matching
- Find a perfect matching
- Find minimum-weight perfect matching for a given weights of edges
- Find maximum-weight matching

Algorithms depends on whether a graph is bipartite or it contains an odd cycle

Definition

Let $M \subseteq E$ be a matching of a graph G

- A path P is *M-alternating* if its edges are alternately in and not in M .
- An *M-alternating* path is *M-augmenting* if both end-vertices are *M-exposed*.

Definition

Let $M \subseteq E$ be a matching of a graph G

- A path P is *M-alternating* if its edges are alternately in and not in M .
- An *M-alternating* path is *M-augmenting* if both end-vertices are *M-exposed*.

Augmenting path theorem of matchings

A matching M in a graph $G = (V, E)$ is maximum if and only if there is no *M-augmenting* path.

Definition

Let $M \subseteq E$ be a matching of a graph G

- A path P is *M-alternating* if its edges are alternately in and not in M .
- An *M-alternating* path is *M-augmenting* if both end-vertices are *M-exposed*.

Augmenting path theorem of matchings

A matching M in a graph $G = (V, E)$ is maximum if and only if there is no *M-augmenting* path.

Notes

- Algorithms for matching are based on finding augmenting paths
- However, testing non-existence of a alternating path is impractical

Definition

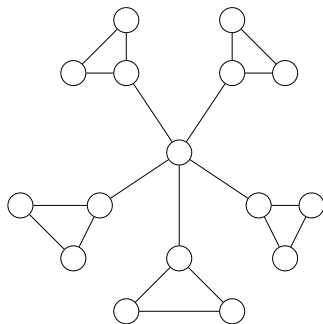
- Let $\text{defic}(G) = |V| - 2|M|$ be the number of exposed vertices by a maximum size matching in G
- Let $\text{oc}(G)$ be the number of odd components of a graph G

Definition

- Let $\text{defic}(G) = |V| - 2|M|$ be the number of exposed vertices by a maximum size matching in G
- Let $\text{oc}(G)$ be the number of odd components of a graph G

Observations

- $\text{defic}(G) \geq \text{oc}(G)$
- For every $A \subseteq V$ it holds that $\text{defic}(G) \geq \text{oc}(G \setminus A) - |A|$.



Definition

- Let $\text{defic}(G) = |V| - 2|M|$ be the number of exposed vertices by a maximum size matching in G
- Let $\text{oc}(G)$ be the number of odd components of a graph G

Observations

- $\text{defic}(G) \geq \text{oc}(G)$
- For every $A \subseteq V$ it holds that $\text{defic}(G) \geq \text{oc}(G \setminus A) - |A|$.

Tutte's matching theorem

A graph G has a perfect matching if and only if $\text{oc}(G \setminus A) \leq |A|$ for every $A \subseteq V$.

Definition

- Let $\text{defic}(G) = |V| - 2|M|$ be the number of exposed vertices by a maximum size matching in G
- Let $\text{oc}(G)$ be the number of odd components of a graph G

Observations

- $\text{defic}(G) \geq \text{oc}(G)$
- For every $A \subseteq V$ it holds that $\text{defic}(G) \geq \text{oc}(G \setminus A) - |A|$.

Tutte's matching theorem

A graph G has a perfect matching if and only if $\text{oc}(G \setminus A) \leq |A|$ for every $A \subseteq V$.

Theorem: Tutte-Berge Formula

$$\text{defic}(G) = \max \{ \text{oc}(G \setminus A) - |A|; A \subseteq V \}$$

Initialization of M -alternating tree T on vertices $A \dot{\cup} B$

$T = A = \emptyset$ and $B = \{r\}$ where r is an M -exposed root.

Building an alternating tree

Initialization of M -alternating tree T on vertices $A \dot{\cup} B$

$T = A = \emptyset$ and $B = \{r\}$ where r is an M -exposed root.

Use $uv \in E$ to extend T

Input: An edge $uv \in E$ such that $u \in B$ and $v \notin A \cup B$ and v is M -covered.

Action: Let $vz \in M$ and extend T by edges $\{uv, vz\}$ and A by v and B by z .

Building an alternating tree

Initialization of M -alternating tree T on vertices $A \dot{\cup} B$

$T = A = \emptyset$ and $B = \{r\}$ where r is an M -exposed root.

Use $uv \in E$ to extend T

Input: An edge $uv \in E$ such that $u \in B$ and $v \notin A \cup B$ and v is M -covered.

Action: Let $vz \in M$ and extend T by edges $\{uv, vz\}$ and A by v and B by z .

Properties

- r is the only M -exposed vertex of T .
- Vertices in A have only one child in T which is connected by an edge of M
- For every v of T , the path in T from v to r is M -alternating.
- $|B| = |A| + 1$

Building an alternating tree

Initialization of M -alternating tree T on vertices $A \dot{\cup} B$

$T = A = \emptyset$ and $B = \{r\}$ where r is an M -exposed root.

Use $uv \in E$ to extend T

Input: An edge $uv \in E$ such that $u \in B$ and $v \notin A \cup B$ and v is M -covered.

Action: Let $vz \in M$ and extend T by edges $\{uv, vz\}$ and A by v and B by z .

Properties

- r is the only M -exposed vertex of T .
- Vertices in A have only one child in T which is connected by an edge of M
- For every v of T , the path in T from v to r is M -alternating.
- $|B| = |A| + 1$

Use $uv \in E$ to augment M

Input: An edge $uv \in E$ such that $u \in B$ and $v \notin A \cup B$ and v is M -exposed.

Action: Let P be the path obtained by attaching uv to the path from r to u in T .
Replace M by $M \triangle E(P)$.

Definition

M -alternating tree T is M -frustrated if every edge of G having one end vertex in B has the other end vertex in A .

Definition

M -alternating tree T is M -frustrated if every edge of G having one end vertex in B has the other end vertex in A .

Observation

If a bipartite graph G has a matching M and an frustrated M -alternating tree, then G has no perfect matching.

Algorithm

```

1 Init:  $M := \emptyset$ 
2 while  $G$  contains an  $M$ -exposed vertex  $r$  do
3    $A := \emptyset$  and  $B = \{r\}$  # Build an  $M$ -alternating tree from  $r$ .
4   while there exists  $uv \in E$  with  $u \in B$  and  $v \notin A \cup B$  do
5     if  $v$  is  $M$ -covered then
6       | Use  $uv$  to extend  $T$ 
7     else
8       | Use  $uv$  to augment  $M$ 
9       | break # Terminate the inner loop.
10  if  $r$  is still  $M$ -exposed then
11    | return There is no perfect matching #  $T$  is a frustrated tree.
12 return Perfect matching  $M$ 

```

Algorithm

```
1 Init:  $M := \emptyset$ 
2 while  $G$  contains an  $M$ -exposed vertex  $r$  do
3    $A := \emptyset$  and  $B = \{r\}$  # Build an  $M$ -alternating tree from  $r$ .
4   while there exists  $uv \in E$  with  $u \in B$  and  $v \notin A \cup B$  do
5     if  $v$  is  $M$ -covered then
6       | Use  $uv$  to extend  $T$ 
7     else
8       | Use  $uv$  to augment  $M$ 
9       | break # Terminate the inner loop.
10  if  $r$  is still  $M$ -exposed then
11    | return There is no perfect matching #  $T$  is a frustrated tree.
12 return Perfect matching  $M$ 
```

Theorem

The algorithm decides whether a given bipartite graph G has a perfect matching and find one if exists. The algorithm calls $O(n)$ augmenting operations and $O(n^2)$ extending operations.

Definition

Let C be an odd circuit in G . The graph $G \times C$ has vertices $(V(G) \setminus V(C)) \cup \{c'\}$ where c' is a new vertex and edges

- $E(G)$ with both end-vertices in $V(G) \setminus V(C)$ and
- and uc' for every edge uv with $u \notin V(C)$ and $v \in V(C)$.

Edges $E(C)$ are removed.

Definition

Let C be an odd circuit in G . The graph $G \times C$ has vertices $(V(G) \setminus V(C)) \cup \{c'\}$ where c' is a new vertex and edges

- $E(G)$ with both end-vertices in $V(G) \setminus V(C)$ and
- and uc' for every edge uv with $u \notin V(C)$ and $v \in V(C)$.

Edges $E(C)$ are removed.

Proposition

- $\text{defic}(G) \leq \text{defic}(G \times C)$
- There exists a graph G and an odd cycle C such that $\text{defic}(G) < \text{defic}(G \times C)$

Definition

Let C be an odd circuit in G . The graph $G \times C$ has vertices $(V(G) \setminus V(C)) \cup \{c'\}$ where c' is a new vertex and edges

- $E(G)$ with both end-vertices in $V(G) \setminus V(C)$ and
- and uc' for every edge uv with $u \notin V(C)$ and $v \in V(C)$.

Edges $E(C)$ are removed.

Proposition

- $\text{defic}(G) \leq \text{defic}(G \times C)$
- There exists a graph G and an odd cycle C such that $\text{defic}(G) < \text{defic}(G \times C)$

Remarks

- To find a maximum matching in G , it is not sufficient to find a maximum matching in $G \times C$ and extended by edges of C .

Definition

Let C be an odd circuit in G . The graph $G \times C$ has vertices $(V(G) \setminus V(C)) \cup \{c'\}$ where c' is a new vertex and edges

- $E(G)$ with both end-vertices in $V(G) \setminus V(C)$ and
- and uc' for every edge uv with $u \notin V(C)$ and $v \in V(C)$.

Edges $E(C)$ are removed.

Proposition

- $\text{defic}(G) \leq \text{defic}(G \times C)$
- There exists a graph G and an odd cycle C such that $\text{defic}(G) < \text{defic}(G \times C)$

Remarks

- To find a maximum matching in G , it is not sufficient to find a maximum matching in $G \times C$ and extended by edges of C .
- We will contract only odd cycles on our alternating tree

Definition

Let C be an odd circuit in G . The graph $G \times C$ has vertices $(V(G) \setminus V(C)) \cup \{c'\}$ where c' is a new vertex and edges

- $E(G)$ with both end-vertices in $V(G) \setminus V(C)$ and
- and uc' for every edge uv with $u \notin V(C)$ and $v \in V(C)$.

Edges $E(C)$ are removed.

Proposition

- $\text{defic}(G) \leq \text{defic}(G \times C)$
- There exists a graph G and an odd cycle C such that $\text{defic}(G) < \text{defic}(G \times C)$

Remarks

- To find a maximum matching in G , it is not sufficient to find a maximum matching in $G \times C$ and extended by edges of C .
- We will contract only odd cycles on our alternating tree
- G' , M' a T' denotes graph, matching, and alternating tree obtained by a sequence of contractions

Use uv to shrink and update M' and T'

Input: A matching M' of a graph G' , an M' -alternating tree T' , edge $uv \in E'$ such that $u, v \in B'$

Action: Let C be the circuit formed by uv together with the path in T' from u to v .

Replace

- G' by $G' \times C$
- M' by $M' \setminus E(C)$
- T by the tree having edge-set $E(T) \setminus E(C)$
- $A' := A' \setminus V(C)$
- $B' := B' \setminus V(C) \cup \{c'\}$ where c' is a new pseudo-vertex

Use uv to shrink and update M' and T'

Input: A matching M' of a graph G' , an M' -alternating tree T' , edge $uv \in E'$ such that $u, v \in B'$

Action: Let C be the circuit formed by uv together with the path in T' from u to v .

Replace

- G' by $G' \times C$
- M' by $M' \setminus E(C)$
- T by the tree having edge-set $E(T) \setminus E(C)$
- $A' := A' \setminus V(C)$
- $B' := B' \setminus V(C) \cup \{c'\}$ where c' is a new pseudo-vertex

Pozorování

- T' je M' -alternating tree on vertices A' a B'

Use uv to shrink and update M' and T'

Input: A matching M' of a graph G' , an M' -alternating tree T' , edge $uv \in E'$ such that $u, v \in B'$

Action: Let C be the circuit formed by uv together with the path in T' from u to v .

Replace

- G' by $G' \times C$
- M' by $M' \setminus E(C)$
- T by the tree having edge-set $E(T) \setminus E(C)$
- $A' := A' \setminus V(C)$
- $B' := B' \setminus V(C) \cup \{c'\}$ where c' is a new pseudo-vertex

Pozorování

- T' je M' -alternating tree on vertices A' a B'
- A' contains only original vertices of G (no pseudo-vertex)

Use uv to shrink and update M' and T'

Input: A matching M' of a graph G' , an M' -alternating tree T' , edge $uv \in E'$ such that $u, v \in B'$

Action: Let C be the circuit formed by uv together with the path in T' from u to v .

Replace

- G' by $G' \times C$
- M' by $M' \setminus E(C)$
- T by the tree having edge-set $E(T) \setminus E(C)$
- $A' := A' \setminus V(C)$
- $B' := B' \setminus V(C) \cup \{c'\}$ where c' is a new pseudo-vertex

Pozorování

- T' je M' -alternating tree on vertices A' a B'
- A' contains only original vertices of G (no pseudo-vertex)
- Odd components in $G' \setminus A'$ corresponds to odd vertices of $G \setminus A'$

Use uv to shrink and update M' and T'

Input: A matching M' of a graph G' , an M' -alternating tree T' , edge $uv \in E'$ such that $u, v \in B'$

Action: Let C be the circuit formed by uv together with the path in T' from u to v .

Replace

- G' by $G' \times C$
- M' by $M' \setminus E(C)$
- T by the tree having edge-set $E(T) \setminus E(C)$
- $A' := A' \setminus V(C)$
- $B' := B' \setminus V(C) \cup \{c'\}$ where c' is a new pseudo-vertex

Pozorování

- T' je M' -alternating tree on vertices A' a B'
- A' contains only original vertices of G (no pseudo-vertex)
- Odd components in $G' \setminus A'$ corresponds to odd vertices of $G \setminus A'$
- If T' is a M' -frustrated tree in G' , then G has no perfect matching

Algorithm

```

1 Init:  $M := \emptyset$ 
2 while  $G$  contains an  $M$ -exposed vertex  $r$  do
3    $M' = M$ ,  $G' = G$  and  $T = (\{r\}, \emptyset)$ 
4   while there exists  $uv \in E(G')$  with  $u \in B$  and  $v \notin A$  do
5     if  $v \in B$  then
6       | Use  $uv$  to shrink and update  $M'$  and  $T$ 
7     else if  $v$  is  $M'$ -covered then
8       | Use  $uv$  to extend  $T$ 
9     else
10      | Use  $uv$  to augment  $M'$ 
11      | Extend  $M'$  to a matching  $M$  of  $G$ 
12      | break # Terminate the inner loop.
13   if  $r$  is still  $M$ -exposed then
14     | return There is no perfect matching
15 return Perfect matching  $M$ 

```

Simple algorithm for implementation

Build an alternating tree from every uncovered vertex one by one.

- If an augmenting path is found, augment matching.
- If a frustrated tree is found, do nothing.

Maximum matching in general graphs

Simple algorithm for implementation

Build an alternating tree from every uncovered vertex one by one.

- If an augmenting path is found, augment matching.
- If a frustrated tree is found, do nothing.

Algorithm which calculates A for Tutte-Berge formula

```
1 Init:  $M, \hat{A}, \hat{B} := \emptyset$ 
2 for  $u \in V$  do
3   if  $u$  is not  $M$ -covered then
4     Build an  $M$ -alternating tree rooted in  $u$  on vertices  $A$  and  $B$ 
5     if An augmenting path is found then
6       Augment matching  $M$ 
7     else
8        $\hat{A} := \hat{A} \cup A$ 
9        $\hat{B} := \hat{B} \cup B$ 
10       $G := G \setminus (A \cup B)$ 
11 return Maximal matching  $M$ 
```

Maximum matching in general graphs

Simple algorithm for implementation

Build an alternating tree from every uncovered vertex one by one.

- If an augmenting path is found, augment matching.
- If a frustrated tree is found, do nothing.

Algorithm which calculates A for Tutte-Berge formula

```
1 Init:  $M, \hat{A}, \hat{B} := \emptyset$ 
2 for  $u \in V$  do
3   if  $u$  is not  $M$ -covered then
4     Build an  $M$ -alternating tree rooted in  $u$  on vertices  $A$  and  $B$ 
5     if An augmenting path is found then
6       Augment matching  $M$ 
7     else
8        $\hat{A} := \hat{A} \cup A$ 
9        $\hat{B} := \hat{B} \cup B$ 
10       $G := G \setminus (A \cup B)$ 
11 return Maximal matching  $M$ 
```

Observe that $V(G) - 2|M| = \text{oc}(G \setminus \hat{A}) - |\hat{A}|$.

Complexity

$\mathcal{O}(n)$ alternating trees are build and building one tree requires

- $\mathcal{O}(m)$ edges uv are tested whether $u \in B$ and $v \notin A$
- $\mathcal{O}(n)$ edges extends the tree
- $\mathcal{O}(n)$ cycles are contracted
- The sum of length of contracted cycles is $\mathcal{O}(n)$
- The complexity of our algorithm will be $\mathcal{O}(n)m\alpha(n)$ but we need to handle shrinking efficiently

Complexity

$\mathcal{O}(n)$ alternating trees are build and building one tree requires

- $\mathcal{O}(m)$ edges uv are tested whether $u \in B$ and $v \notin A$
- $\mathcal{O}(n)$ edges extends the tree
- $\mathcal{O}(n)$ cycles are contracted
- The sum of length of contracted cycles is $\mathcal{O}(n)$
- The complexity of our algorithm will be $\mathcal{O}(n)m\alpha(n)$ but we need to handle shrinking efficiently

Speed up (Micali, Vazirani)

Basic idea:

- In every iteration, build alternating trees from all uncovered trees simultaneously
- The number of iterations is $\mathcal{O}(\sqrt{n})$ instead of $\mathcal{O}(n)$
- Total complexity is $\mathcal{O}(m\sqrt{n})$

Problem

For a graph G given by a list of edges sorted by their weights find a minimum spanning tree, i.e. tree covering all vertices.

Minimum spanning tree in a graph

Problem

For a graph G given by a list of edges sorted by their weights find a minimum spanning tree, i.e. tree covering all vertices.

Algorithm

Start with the empty set of edges T and process all edges uv in increasing weight. If u and v belong to a different components of (V, T) , then add uv into T .

Minimum spanning tree in a graph

Problem

For a graph G given by a list of edges sorted by their weights find a minimum spanning tree, i.e. tree covering all vertices.

Algorithm

Start with the empty set of edges T and process all edges uv in increasing weight. If u and v belong to a different components of (V, T) , then add uv into T .

Question

How to determine whether u and v belong to different components?

Minimum spanning tree in a graph

Problem

For a graph G given by a list of edges sorted by their weights find a minimum spanning tree, i.e. tree covering all vertices.

Algorithm

Start with the empty set of edges T and process all edges uv in increasing weight. If u and v belong to a different components of (V, T) , then add uv into T .

Question

How to determine whether u and v belong to different components?

Union-Find problem

- Init: Every element (vertex) u belong to the set $\{u\}$
- Union(u,v): Merge sets containing u a v
- Find(u): Find a set containing u

Forest

- A forest has one vertex for every element
- One of the forest corresponds to one set
- Every vertex u stores its parent $p[u]$ in the forest
- The parent of a root of a tree is NULL \Rightarrow initialize $p[u] = \text{NULL}$ for every vertices
- Every root stores the size of its tree

Forest

- A forest has one vertex for every element
- One of the forest corresponds to one set
- Every vertex u stores its parent $p[u]$ in the forest
- The parent of a root of a tree is NULL \Rightarrow initialize $p[u] = \text{NULL}$ for every vertices
- Every root stores the size of its tree

Union(u,v)

- Find roots u' , v' of trees containing u , v
- If u' contains more element than v' , then u' become the parent of v'

Forest

- A forest has one vertex for every element
- One of the forest corresponds to one set
- Every vertex u stores its parent $p[u]$ in the forest
- The parent of a root of a tree is NULL \Rightarrow initialize $p[u] = \text{NULL}$ for every vertices
- Every root stores the size of its tree

Union(u, v)

- Find roots u' , v' of trees containing u , v
- If u' contains more element than v' , then u' become the parent of v'

Find(u): Find the root of u

- Find the root u' of u
- For all vertices of the path from u do u' change the parent to be u' (except u')
- The amortized complexity is $O(\alpha(n))$

Tasks for next week

The second assignment

Finish Goldberg algorithm

The third assignment

- Understand algorithm and its correctness
- Design data representation
- Find invariants and tests
- Try to implement finding maximum matching in bipartite graphs

Tasks for next week

The second assignment

Finish Goldberg algorithm

The third assignment

- Understand algorithm and its correctness
- Design data representation
- Find invariants and tests
- Try to implement finding maximum matching in bipartite graphs

Literature

- Cunningham, Cook, Pulleyblank, Schrijver: Combinatorial optimization, John Wiley & Sons, 1997 (book chapter)
- Jan Vondrák: Polyhedral techniques in combinatorial optimization, 2010 (lecture notes)
<https://theory.stanford.edu/~jvondrak/CS369P/lec4.pdf>
- Michel X. Goemans: Combinatorial Optimization (lecture notes)
<http://math.mit.edu/~goemans/18433S15/matching-notes.pdf>
<http://math.mit.edu/~goemans/18433S15/matching-nonbip-notes.pdf>
- Uri Zwick: Lecture notes on: Maximum matching in bipartite and non-bipartite graphs
<https://www.cs.tau.ac.il/~zwick/grad-algo-0910/match.pdf>
- Visualization: <https://www-m9.ma.tum.de/graph-algorithms/matchings-blossom-algorithm/>