Computer Science for Engineers

Lecture 13

Algorithms – part 3

Prof. Dr. Dr.-Ing. Jivka Ovtcharova
Dipl. Wi.-Ing. Dan Gutu
12th of February 2010
• Definition: A set of plane points is called convex if it contains for every two points also the connecting line between these points.

convex Set A

non-convex Set B

• Definition: The convex cover of a set of \( n \) points is **the smallest convex polygon** containing all \( n \) points.
The Convex Cover (2)

- Motivation: Calculate a route for a robot
  - Situation:
    - A mobile robot must trace back the way from A to B
    - There is an obstacle on the way (in form of a polygon)
  
  - Problem: Determining the shortest way from A to B with a detour around the obstacle

  - Solution: Ride along the convex cover of
    - Start Point A
    - Points in Obstacle (H)
    - Destination Point B
• Motivation: CAD-Modeling
  - In tests of insertion or collision of parts, the curves and surfaces are approximated as polygons.
  - In intersection-tests, the convex covers of the point sets to be tested are first checked for intersection.
  - If the intersection of the convex sets is empty, the curves do not intersect as well.
• Given: A point set in a two-dimensional coordinate system. For simplicity, we will assume that no three points are in one line, and that no two points have the same x-coordinate.

• Goal: Find the minimal convex cover

• Principle of the algorithm:
  - Sort points based on their x-coordinate in ascending order
  - Divide point set in two partial sets L and R, with L containing the n/2 left and R the n/2 right points
  - Determine recursively the convex cover for every partial set (CS_L or CS_R).
  - Merge the convex covers of both partial sets and determine the convex cover for the entire set (the so-called Merge-Step). The recursion breaks when n < 3 or n = 3.
- Given: n points
- Sort the points according to the x axis in ascending order.
- Divide the point set recursively until one point set contains 3 points max.
- Since the convex cover of 3 points is a triangle, build convex covers for each of the smallest partial sets
- With 15 points, 3 division steps are necessary!
- Determine the convex cover for the partial sets. These are either lines or triangles.
- Then, merge the convex polygons recursively.
Example: Combination Level 3

- Merge the convex polygons recursively

1. Division

2. Division

3. Division

5.5.3. Convex Cover

5.5. Examples for Algorithms
• Merge the convex polygons recursively
Example: Combination Level 1

- Merge the convex polygons recursively
5.5. Examples for Algorithms
5.5.3. Convex Cover

Merge-Step: Define the Upper and Lower Tangent

upper Tangent

lower Tangent

CS_L

CS_R
• The upper and the lower extreme points divide the border of the CC in two border lines.

upper Extreme Point

lower Extreme Point
• **P2** is the successor of **P1** on the Border of CC exactly if the polar angle belonging to it is minimal on P2.
Merge-Step: Definition of the Upper Tangent

If $CS_L = \{P_1, \ldots, P_n\}$ and $CS_R = \{Q_1, \ldots, Q_m\}$.

1. View the upper extreme points $P_1$ and $Q_1$ and the successors $P_2$ and $Q_2$ clockwise, and let $P_1$ be higher than $Q_1$.

2. Define the minimum of the angle associated with $P_1P_2$, $P_1Q_1$ and $P_1Q_2$.

3. Cases:
   - $P_1Q_1$ is minimal: tangent found, ready
   - $P_1P_2$ minimal: replace $P_1$ by $P_2$ and $P_2$ by $P_3$ (walk clockwise on the left convex cover)
   - $P_1Q_2$ minimal: replace $Q_1$ by $Q_2$ and $Q_2$ by $Q_3$ (walk clockwise on the right convex cover)


The case of the lower tangent is symmetrical.
Finding the Upper Tangent (1)

Check $P_1$ and $Q_1$

Angle not minimal

5.5. Examples for Algorithms
5.5.3. Convex Cover
Finding the Upper Tangent (2)

Angle minimal

Check $P_1$ and $Q_1$

$P_1P_2$ minimal $\Rightarrow$ replace $P_1$ by $P_2$
Finding the Upper Tangent (3)

Angle minimal

Check $P_2$ and $Q_1$

$P_2Q_2$ minimal $\Rightarrow$ replace $Q_1$ by $Q_2$
Find the Upper Tangent (4)

Check $P_2$ and $Q_2$

$P_2Q_2$ minimal $\Rightarrow$ Tangent found

5.5.3. Convex Cover
Outline

Lecture Content

1. Preface
2. Basics
3. Object orientation
4. Data Structures
5. Algorithms
   5.1. Introduction
   5.2. Characteristics
   5.3. Complexity
   5.4. Design Methods
   5.5. Examples of Algorithms
      5.5.1 Sorting procedures
      5.5.2 Voronoi Diagrams
      5.5.3 Convex Cover
      5.5.4 Network flow
Network Flow (1)

- Motivation: Solution of logistical problems
- Examples:
  - Number of vehicles that can drive through a street net
  - Amount of electric power that can flow through a power lines net
  - Package handling in computer networks: When as many packages as possible (or even real-time) must be transmitted from the source computer to the target computer, the question about the maximal number of transmissible packages arises

![Diagram of network flow with Source and Target highlighted]
Problem description:

1. Set of computers (= nodes of a graph), one being the source of a data set to be transmitted, one the target.

2. Two different computers are connected through singular connections (= edges) and can transmit data through this connection.
   - Data is transmitted as packages of fixed length.
   - Every connection has a capacity (maximal amount of transmitted data packages per unit of time).

3. The networking is such that different paths are created by which the data can be transported from **source** to **target**.

Question: how many data packages can be transmitted at most per unit of time from source to target?
Network Flow (3)

• A network is a directed graph \( G = (V, E, c) \) with nodes \( q \) (source) and \( s \) (target), as well as a capacity function, \( c : E \rightarrow R^+ \)

\( V \) is the node set and \( E \) the edge set

• A flow for the network is a function \( f : E \rightarrow R \), consisting of:
  - Capacity limitation
    \[ \forall u, v \in V : f(u, v) \leq c(u, v) \quad e(u, v) \in E \]
  - Flow consistency
    \[ \forall u, v \in V : f(u, v) = -f(v, u) \quad e(u, v) \in E \]

• The value of \( f(u, v) \) can be positive or negative

• The value of a flow is described by \( w(f) = \sum_{v \in V} f(q, v) \quad e(q, v) \in E \)

The maximal flow is defined as maximal allowed flow in \( G \) regarding \( q \) and \( s \)
Network Flow (4)

- Assume \( f \) is an allowable flow for \( G = (V, E) \).

  - The **rest capacity** of an edge \((u, v)\) is defined as follows:
    
    \[
    \text{rest}(u,v) = c(u,v) - f(u,v)
    \]

  - The **rest graph** of \( G \) (reg. \( f \)) is defined by \( G_f = (V, E_f) \), \( e(u,v) \in E \), while it is true that:
    
    \[
    E_f = \{(u,v) \in V \times V : \text{rest}(u,v) > 0\} \quad e(u,v) \in E
    \]

  - Every ordered path from \( q \) to \( s \) in rest graphs is called an **increasing path**.

  - An **intersection** \((A, B)\) of a network is a separation of \( V \) in 2 disjunct partial sets \( A \) and \( B \), so that \( q \in A \) and \( s \in B \). The **capacity of the intersection** is
    
    \[
    c(A, B) = \sum_{u \in A, v \in B} c(u,v), \quad e(u,v) \in E
    \]
The following statements are equivalent:

- $f$ is maximal flow in $G$

- The rest graph of $f$ contains no increasing (augmenting) path

- $w(f^{'}) = c(A, B)$ for an intersection $(A, B)$ of $G$

if $f$ is an allowable flow for $G$. 
Network Flow: The Ford-Fulkerson - Algorithm

• Principle of the algorithm:

The main algorithm of Ford-Fulkerson is based on the Idea of the Max-Flow Min-Cut – theorem, e.g., the maximal flow is found when there are no more augmenting Paths.

• Pseudocode:

```plaintext
for(all (u,v) in E ) f(u,v) = 0; // Initialising

while( there is an increasing path p in Rest Graphs G_f) {
    r = min{rest(u,v) | (u,v) lies in p};
    for(all (u,v) on Path p ){
        f(u,v) = f(u,v) + r;
        // f(v,u) = f(v,u) - r;
    }
}
```
Example: Determining the max. Flow in Network

Edges indicate the capacity:
Step 1: Graph is initialized with flow = 0 edge notation: \( f/c/r \)

- \( f \): flow
- \( c \): initial capacity
- \( r \): rest capacity

![Graph diagram with flow values](image)
Iteration 1: Choosing the first path: $q \rightarrow c \rightarrow e \rightarrow s$

with $r = \min\{\text{rest}(u,v) \mid (u,v) \text{ lies in } p\} = 2$. 
• Iteration 2: make corresponding edge descriptions and choose the second path $q \rightarrow e \rightarrow d \rightarrow s$ with $r = \min\{\text{rest}(u,v) \mid (u,v) \text{ lies in } p\} = 1$

Note: rest capacity of $(c,e)$ is used up after the first iteration
• Iteration 3: make corresponding edge descriptions and choose third path \( q \rightarrow a \rightarrow b \rightarrow s \) with \( r = \min\{\text{rest}(u,v) \mid (u,v) \text{ lies in } p\} = 2 \)

Note:
- Rest capacity of \((e,d)\) is used up after the second iteration
- When making a corresponding edge description \((e,d)\), the back edge \((d,e)\) must be taken in consideration: \(f(v,u) = f(v,u) - r\) (see Pseudocode)
• Iteration 4: Make corresponding edge descriptions and choose the fourth path with $r = \min\{\text{rest}(u,v) \mid (u,v) \text{ lies in } p\} = 1$

$$q \rightarrow a \rightarrow d \rightarrow e \rightarrow s$$

Note: Rest capacities of (e,d) and (b,s) are used up after the third iteration.
• Iteration 5: Make corresponding edge descriptions and choose the fifth path with \( r = \min\{\text{rest}(u,v) \mid (u,v) \text{ lies in } p\} = 2 \)

\[ q \rightarrow c \rightarrow d \rightarrow s \]

Note:
- Rest capacities of (a,d) and (e,s) are used up after the fourth iteration
- When making a corresponding edge description for the border (d,e), the back edge must be taken into consideration
• Result:
  - No other paths are possible
  - The calculation of the maximal flow is thus ended!
  - Note:
    ▪ Which path is chosen in every iteration Step remains open!
    ▪ The number of iterations depends on the choice of paths
Given: The following network with the source node q, the target node s, the further nodes $v_1, ..., v_6$ and the inscriptions on the edges $f/c/r$ with $f = \text{flow}$, $c = \text{initial capacity}$ and $r = \text{rest capacity}$.

a.) Use the Ford-Fulkerson algorithm on the above mentioned network. List in a table all possible paths for every step through the rest capacities network, as well as their length and capacity. Choose as the next path the shortest, and - if equal length appears - the one with highest capacity.
Problem: Initial Determination of a Flow Network (determine Flow with 0)

5.5. Examples for Algorithms
5.5.4. Network Flow
Problem (Step 1)

- Define all possible flows from source to target
- Choose flow with minimal length. If there is more than one flow with minimal length, choose the one with max. flow
  - Choose q, v_1, v_4, s
  - Flow Increase = 3

<table>
<thead>
<tr>
<th>Path</th>
<th>f = 0</th>
<th>Length</th>
<th>Flow</th>
</tr>
</thead>
<tbody>
<tr>
<td>q, v_1, v_4, s</td>
<td>4</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>q, v_1, v_4, v_5, v_6, s</td>
<td>6</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>q, v_1, v_5, v_6, s</td>
<td>5</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>q, v_1, v_2, v_5, v_6, s</td>
<td>6</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>q, v_1, v_2, v_3, v_5, v_6, s</td>
<td>7</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>q, v_1, v_2, v_3, v_6, s</td>
<td>6</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>q, v_2, v_5, v_6, s</td>
<td>5</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>q, v_2, v_3, v_5, v_6, s</td>
<td>6</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>q, v_2, v_3, v_6, s</td>
<td>5</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>q, v_3, v_5, v_6, s</td>
<td>5</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>q, v_3, v_6, s</td>
<td>4</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>
Determination of the Flow Network after Step 1

5.5. Examples for Algorithms

5.5.4. Network Flow

- **f**: Flow
- **c**: Initial Cap.
- **r**: Rest Capacity

Graph with nodes **q**, **v1**, **v2**, **v3**, **v4**, **v5**, **v6**, and **s**.

- **q** to **v1**: 3/3/0
- **v1** to **v2**: 0/2/2
- **v1** to **v4**: 3/4/1
- **v2** to **v3**: 0/6/6
- **v2** to **v1**: 0/5/5
- **v2** to **v5**: 0/3/3
- **v3** to **v2**: 0/2/2
- **v3** to **v4**: 0/5/5
- **v4** to **v3**: 3/4/1
- **v4** to **v5**: 0/4/4
- **v5** to **v2**: 0/3/3
- **v5** to **v6**: 0/4/4
- **v5** to **v1**: 0/2/2
- **v6** to **v5**: 0/2/2
- **v6** to **v1**: 0/3/3
- **s** to **v6**: 0/5/5
Problem (Step 2)

- Update the flows
- Choose the flow with min. length
  - Choose q, v₃, v₆, s
  - Flow increase = 2

<table>
<thead>
<tr>
<th>Path</th>
<th>f</th>
<th>Length</th>
<th>Flow</th>
</tr>
</thead>
<tbody>
<tr>
<td>q, v₁, v₄, s</td>
<td>3</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>q, v₁, v₄, v₅, v₆, s</td>
<td>/</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>q, v₁, v₅, v₆, s</td>
<td>/</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>q, v₁, v₂, v₅, v₆, s</td>
<td>/</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>q, v₁, v₂, v₃, v₅, v₆, s</td>
<td>/</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>q, v₁, v₂, v₃, v₆, s</td>
<td>/</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>q, v₂, v₅, v₆, s</td>
<td>5</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>q, v₂, v₃, v₅, v₆, s</td>
<td>6</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>q, v₂, v₃, v₆, s</td>
<td>5</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>q, v₃, v₅, v₆, s</td>
<td>5</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>q, v₃, v₆, s</td>
<td>4</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>
Determination of the Flow Network after Step 2

- **f**: Flow
- **c**: Initial Cap.
- **r**: Rest Capacity

Graph showing flow network with vertices labeled as follows:
- **v1**: Initial capacity 0, rest capacity 2
- **v2**: Initial capacity 3, rest capacity 0
- **v3**: Initial capacity 6, rest capacity 2
- **v4**: Initial capacity 0, rest capacity 3
- **v5**: Initial capacity 0, rest capacity 4
- **v6**: Initial capacity 0, rest capacity 2
- **q**: Source with flow 3, initial capacity 3, rest capacity 0
- **s**: Sink with flow 3, initial capacity 3, rest capacity 0

Flow paths and capacities are indicated between vertices.

5.5. Examples for Algorithms
5.5.4. Network Flow
Problem (Step 3)

- Update the flows
- Choose flow with min. length
  - Choose q, v₂, v₃, v₆, s
  - The flow q, v₂, v₅, v₆, s may also be possible
  - flow increase = 2

<table>
<thead>
<tr>
<th>Path</th>
<th>f</th>
<th>Length</th>
<th>Flow</th>
</tr>
</thead>
<tbody>
<tr>
<td>q, v₁, v₄, s</td>
<td>3</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>q, v₁, v₄, v₅, v₆, s</td>
<td>/</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>q, v₁, v₅, v₆, s</td>
<td>/</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>q, v₁, v₂, v₅, v₆, s</td>
<td>/</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>q, v₁, v₂, v₃, v₅, v₆, s</td>
<td>/</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>q, v₁, v₂, v₃, v₆, s</td>
<td>/</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>q, v₂, v₅, v₆, s</td>
<td>5</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>q, v₂, v₃, v₅, v₆, s</td>
<td>6</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>q, v₂, v₃, v₆, s</td>
<td>5</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>q, v₃, v₅, v₆, s</td>
<td>/</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>q, v₃, v₆, s</td>
<td>2</td>
<td>4</td>
<td>0</td>
</tr>
</tbody>
</table>
Determination of the Flow Network after Step 3

\[ f/c/r \]

\[ f: \text{Flow} \]
\[ c: \text{Initial Cap.} \]
\[ r: \text{Rest Capacity} \]

5.5. Examples for Algorithms
5.5.4. Network Flow
Problem (Step 4)

- Update the flows
- Choose flow with min. length
  - Choose q, v2, v5, v6, s
  - Flow increase f = 1

<table>
<thead>
<tr>
<th>Path</th>
<th>f</th>
<th>Length</th>
<th>Flow</th>
</tr>
</thead>
<tbody>
<tr>
<td>q, v1, v4, s</td>
<td>3</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>q, v1, v4, v5, v6, s</td>
<td>/</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>q, v1, v5, v6, s</td>
<td>/</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>q, v1, v2, v5, v6, s</td>
<td>/</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>q, v1, v2, v3, v5, v6, s</td>
<td>/</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>q, v1, v2, v3, v6, s</td>
<td>/</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>q, v2, v5, v6, s</td>
<td>5</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>q, v2, v3, v5, v6, s</td>
<td>6</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>q, v2, v3, v6, s</td>
<td>2</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>q, v3, v5, v6, s</td>
<td>/</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>q, v3, v6, s</td>
<td>2</td>
<td>4</td>
<td>0</td>
</tr>
</tbody>
</table>
Determination of the Flow Network after Step 4

5.5.4. Network Flow

- **f**: Flow
- **c**: Initial Cap.
- **r**: Rest Capacity

Graph showing the flow network with vertices labeled as $v_1, v_2, v_3, v_4, v_5, v_6, q, s$. The edges are labeled with the flow values $(u,v)$ where $u$ is the source and $v$ is the destination.
Problem (after Step 4)

- No further flow increase is possible ★ Ford-Fulkerson algorithm terminates

<table>
<thead>
<tr>
<th>Path</th>
<th>f</th>
<th>Length</th>
<th>Flow</th>
</tr>
</thead>
<tbody>
<tr>
<td>q, v1, v4, s</td>
<td>3</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>q, v1, v4, v5, v6, s</td>
<td>/</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>q, v1, v5, v6, s</td>
<td>/</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>q, v1, v2, v5, v6, s</td>
<td>/</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>q, v1, v2, v3, v5, v6, s</td>
<td>/</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>q, v1, v2, v3, v6, s</td>
<td>/</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>q, v2, v5, v6, s</td>
<td>1</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>q, v2, v3, v5, v6, s</td>
<td>/</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>q, v2, v3, v6, s</td>
<td>2</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>q, v3, v5, v6, s</td>
<td>/</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>q, v3, v6, s</td>
<td>2</td>
<td>4</td>
<td>0</td>
</tr>
</tbody>
</table>
b) After solving the previous partial problem, any intersection of graphs can be done. The resulting maximal flow equals 8.

\[ f_{\text{max}} = 3 + 0 + 1 + 0 + 4 = 8 \]
Literature


[Cott01] Vorlesungen TU Cottbus (2001)


Java Programm mit konvexer Hülle und Voronoi Diagrammen:

http://www.pi6.fernuni-hagen.de/GeomLab/VoroGlide/


Weitere Links:

http://www-ti.informatik.tu-cottbus.de/HTML/sortieren.html

http://www.inf.fh-flensburg.de/lang/algorithmen/sortieren/

http://www.inf.fh-flensburg.de/lang/algorithmen/geo/convex.htm