

Informatik Seminar  
Sommersemester 2008  
Thema : Brute Force

Robert Schilling

li5656

# Brute Force

- Sehr einfach zu implementieren
- Basiert direkt auf dem Problem
- Sehr lange Laufzeiten (alle Input-Elemente müssen miteinander verglichen werden)
- Meistens ineffizient

# String Matching

- Algorithmus zum Auffinden von Mustern in Texten
- Text und Muster müssen zum Starten angegeben werden
- Text muss länger als das Muster sein
- Algorithmus beendet, wenn der Text durchlaufen wurde oder das Muster im Text gefunden wurde

# Pseudocode

n: Textlänge; m: Musterlänge

T: Text; P: Muster;

for i := 0 to n-m do

    j := 0

    while j < m and P[j] = T[j+i] do

        j:= j+1;

    if j==m return i

return -1

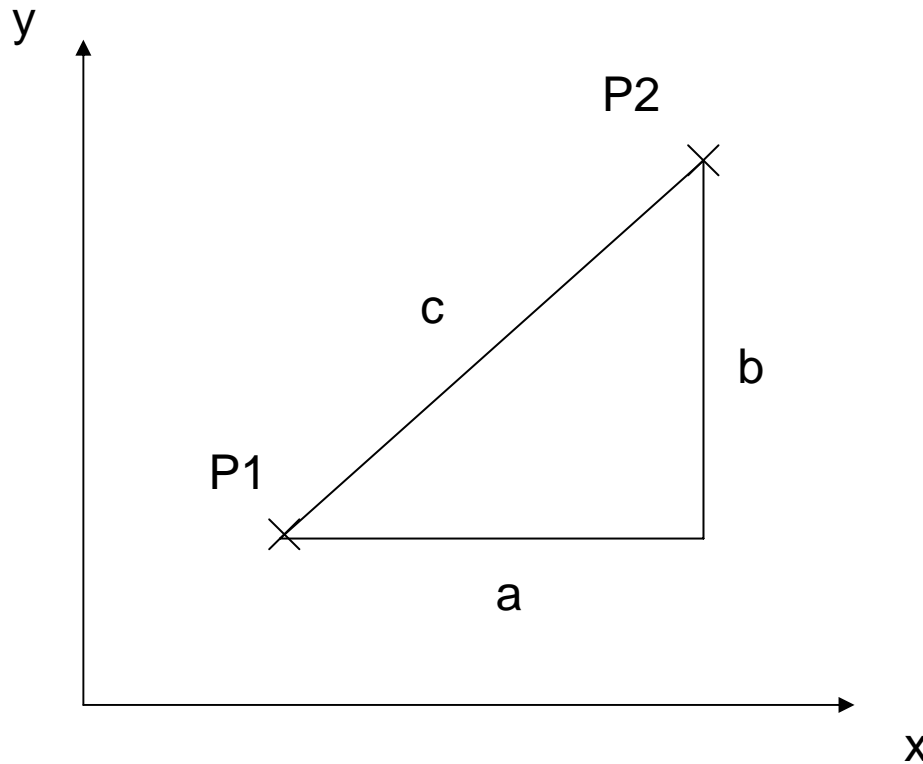
# Laufzeitkomplexität String Matching

- Worst Case :  
 $O(\text{Textlänge} * \text{Musterlänge})$
- Beispiel:  
Text = „ttttttt“  
Muster = „tte“

# Closest Pair

- Menge an Punkten von denen die geringste Distanz zueinander ermittelt wird
- Es müssen mindestens zwei Punkte angegeben werden
- Von allen möglichen Punktepaaren muss die Distanz berechnet werden

# Herleitung der Distanzberechnung



$$a^2 + b^2 = c^2$$

$$a = |X_2 - X_1|$$

$$b = |Y_2 - Y_1|$$

$$c = \sqrt{a^2 + b^2}$$

# Pseudocode

n: Punkteanzahl; min: kleinste Distanz;

d: aktuell berechnete Distanz

Index1,index2: indizes für die geringste Distanz

min = unendlich

for i := 0 to n-1 do

    for j := n+1 to n do

        d=distanz berechnen( $P_i$ , $P_j$ );

        if  $d < \text{min}$  then

            min=d; index1=i;index2=j;

return index1,index2

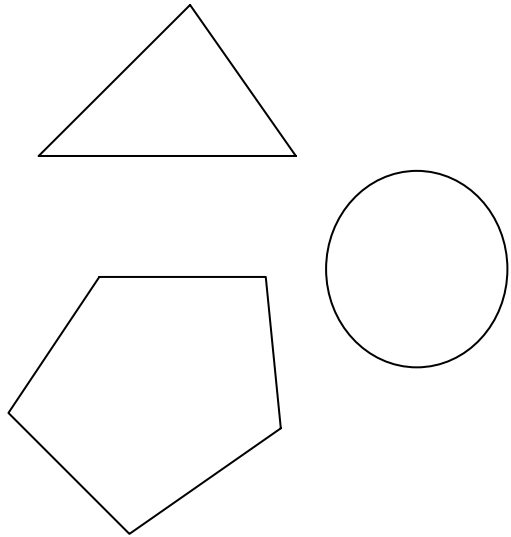
# Laufzeitkomplexität Closest Pair

- Punkte werden vollständig miteinander verglichen
- Laufzeitkomplexität= $O(n^2)$

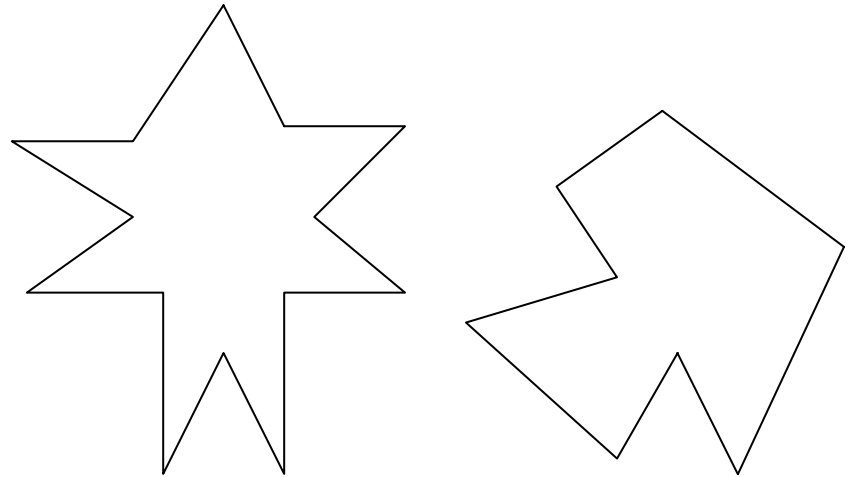
# Convex Hull

- Menge von Punkten ist Input dieses Algorithmus
- Mindestens drei Punkte müssen angegeben werden
- Aus der Punktemenge wird eine konvexe Form gebildet (Alle Punkte sind in dieser Form enthalten)

# Unterscheidung konvexe/ nicht konvexe Formen

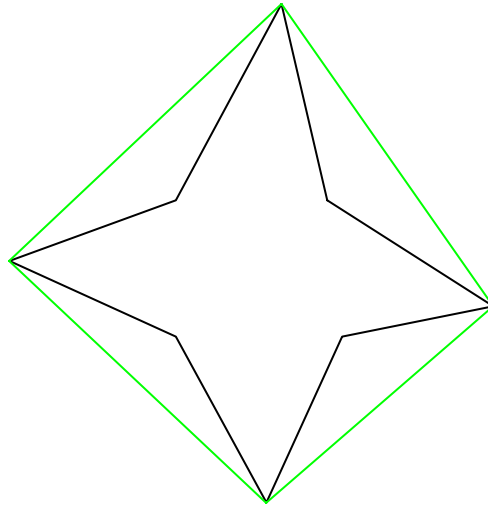


Konvexe Formen



Nicht konvexe Formen

Umwandlung nicht konvex -> konvex



# Bestimmung des Randes von einer konvexen Form

- Allgemeine Geradengleichung wird als Grundlage genommen ( $a \cdot x + b \cdot y - c = 0$ )
- Die Konstanten  $a$ ,  $b$  und  $c$  werden für jedes Punktepaar bestimmt
- Mit der allgemeinen Geradengleichung wird bestimmt ob die restlichen Punkte nur auf einer Seite der Geraden liegen, wenn dies der Fall ist, bildet das Punktepaar den Konvexrand

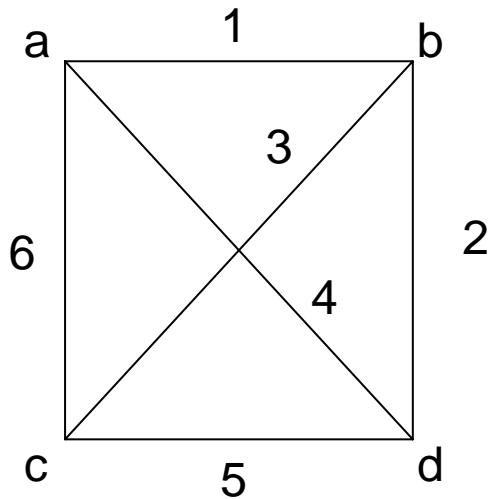
# Laufzeitkomplexität

- Der Algorithmus muss mit allen Punkten die allgemeine Geradengleichung aufstellen ( $a$ ,  $b$ ,  $c$  bestimmen). Die Anderen Punkte werden dann überprüft ob sie nur auf einer Seite liegen
- Daraus resultiert eine Laufzeitkomplexität von  $O(n^3)$

# Traveling Salesman

- Algorithmus um die kürzeste „Reisetour“ zwischen verschiedenen Orten zu bestimmen
- $N$  Orte werden angegeben mit einer Entfernung zueinander
- Jede Mögliche Tour wird zusammengezählt und die kürzeste Tour ist Ergebnis des Algorithmus

# Beispiel für Traveling Salesman



$$a - b - c - d - a = 1 + 3 + 5 + 4 = 13$$

$$a - b - d - c - a = 1 + 2 + 5 + 6 = 14$$

$$a - c - b - d - a = 6 + 3 + 2 + 4 = 15$$

$$a - c - d - b - a = 6 + 5 + 2 + 1 = 14$$

$$a - d - b - c - a = 4 + 2 + 3 + 6 = 15$$

$$a - d - c - b - a = 4 + 5 + 3 + 1 = 13$$

$$a - b - c - d - a = 1 + 3 + 5 + 4 = 13$$

$$a - d - c - b - a = 4 + 5 + 3 + 1 = 13$$

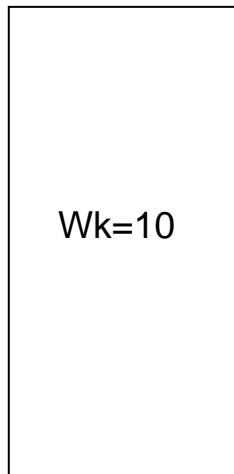
# Laufzeit Traveling Salesman

- In dem Beispiel gibt es sechs mögliche Touren
- Bei fünf Städten wären es 24 Touren
- Allgemein : Touren = (Anzahl Städte - 1)!

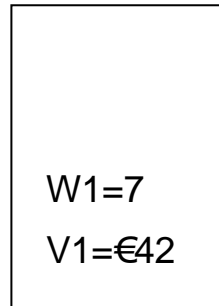
# Knapsack(Rucksack Problem)

- Angenommen man bricht in ein Museum ein und kann nur ein bestimmtes Maß an Gewicht mit sich tragen
- Die Gegenstände müssen nach abwägen des Wertes ausgewählt werden
- Algorithmus durchläuft alle Kombinationen von Gegenständen und gibt die wertvollste Kombination, die tragbar ist, zurück

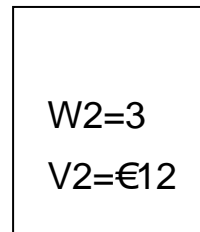
# Beispiel für Knapsack



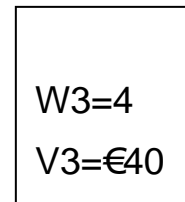
Knapsack



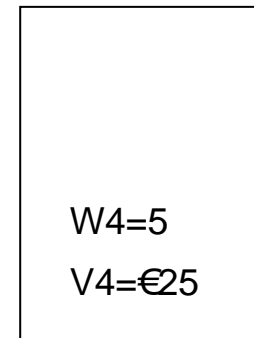
Gegenstand1



Gegenstand2



Gegenstand3



Gegenstand4

G	1	2	3	4	1,2	1,3	1,4	2,3	2,4	3,4	1,2,3	1,2,4	2,3,4	1,3,4	1,2,3,4
W	7	3	4	5	10	11	12	7	8	9	14	15	12	16	19
V	42	12	40	25	54	XX	XX	52	37	65	XX	XX	XX	XX	XX

# Kombinationen Knapsack

- Bei vier Gegenständen gibt es 16 verschiedene Möglichkeiten
- Allgemein gibt es  $2^n$  Kombinationen

# Job Assignement

- Ein Betrieb hat  $N$  Jobs mit  $N$  Arbeitern zu vergeben
- Die Arbeiter werden alle anders bezahlt für ein und denselben Job
- Job Assignement Algorithmus bestimmt für den Betrieb die wirtschaftlichste Lösung

# Beispiel Job Assignment

	Job 1	Job 2	Job 3
Person 1	5	6	7
Person 2	2	3	4
Person 3	4	2	1

$$\langle 1, 2, 3 \rangle = 5 + 3 + 1 = 9$$

$$\langle 1, 3, 2 \rangle = 5 + 4 + 2 = 11$$

$$\langle 2, 1, 3 \rangle = 6 + 2 + 1 = 9$$

$$\langle 2, 3, 1 \rangle = 6 + 4 + 4 = 14$$

$$\langle 3, 1, 2 \rangle = 7 + 2 + 2 = 11$$

$$\langle 3, 2, 1 \rangle = 7 + 3 + 4 = 14$$

$$\langle 2, 1, 3 \rangle = 6 + 2 + 1 = 9$$

$$\langle 1, 2, 3 \rangle = 5 + 3 + 1 = 9$$

# Laufzeit Job Assignment

- Kombinationen vom Beispiel : 6
- Kombinationen allgemein :  $n!$